

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 to obtain the best possible image quality. The tuning procedure provided in this report is accompanied with examples using the IMX219 sensor (RGB-only) and the OX05B1S sensor (RGB-IR) on the AM62A Starter Kit EVM.

Table of Contents

1 Introduction	3
2 Tuning Overview	3
3 Hardware Requirement	4
4 Software Requirement	5
4.1 Processor SDK Linux®	5
4.2 TI's Reference Imaging Software	5
4.3 ISP Tuning Tool	5
5 Sensor Software Integration	5
5.1 Overview of Image Pipeline Software Architecture	5
5.2 Adding Sensor Driver to SDK	6
5.3 Updating TIOVX Modules	6
5.4 Update GStreamer Plug-in for VISS	6
6 Tuning Procedure	9
6.1 Verify Functional Operation of Camera Capturing	9
6.2 Enable Camera Streaming With Initial VPAC Configuration	9
6.3 Adjust Camera Mounting	12
7 Perform Basic Tuning	13
7.1 Launch the Tuning Tool and Create a Project	13
7.2 Tuning Order	15
7.3 Black Level Subtraction	16
7.4 Hardware 3A (H3A)	17
7.5 PCID	17
7.6 Auto White Balance (AWB)	19
7.7 Color Correction	22
8 Perform Fine Tuning	25
8.1 Edge Enhancement (EE)	25
8.2 Noise Filter 4 (NSF4)	26
9 Live Tuning	28
9.1 Requirements	28
9.2 Supported Features	28
10 Summary	30
11 Revision History	30

List of Figures

Figure 3-1. Tuning Hardware and Equipment Setup	4
Figure 5-1. AM6xA Image Processing Pipeline	6
Figure 7-1. Sensor Raw Image Parameters for Tuning	13
Figure 7-2. Image Preview by DCC Tuning Tool	14
Figure 7-3. DCC Tuning Tool Plug-ins	15
Figure 7-4. Black Level Subtraction Tuning	16
Figure 7-5. Image Before Black Level Subtraction	17

Figure 7-6. Image After Black Level Subtraction.....	17
Figure 7-7. PCID Tuning.....	18
Figure 7-8. RAW Image Input to PCID.....	19
Figure 7-9. PCID Output Image in Bayer Pattern.....	19
Figure 7-10. Live Capture Feature of DCC Tuning Tool.....	19
Figure 7-11. Color Chart Image Captured With D50 Lighting.....	20
Figure 7-12. Color Chart Image Captured With D65 Lighting.....	20
Figure 7-13. Color Chart Image Captured With TL84 Lighting.....	20
Figure 7-14. Color Chart Image Captured With A-light.....	20
Figure 7-15. Auto White Balance Tuning.....	21
Figure 7-16. Choosing Corners of Color Checker Chart.....	21
Figure 7-17. Auto White Balance Tuning Results.....	22
Figure 7-18. Image Before Auto White Balance Tuning.....	22
Figure 7-19. Image After Auto White Balance Tuning.....	22
Figure 7-20. Color Correction Tuning.....	23
Figure 7-21. Color Correction Tuning Output.....	23
Figure 7-22. Image Before Color Correction Tuning.....	24
Figure 7-23. Image After Color Correction Tuning.....	24
Figure 8-1. EE Semi-Automatic Tuning GUI.....	25
Figure 8-2. YUV Image Input Before EE Tuning.....	26
Figure 8-3. YUV Image Output After EE Tuning.....	26
Figure 8-4. NSF4 Tuning GUI.....	27
Figure 9-1. EVM IP Address.....	28
Figure 9-2. RAW Capture.....	28
Figure 9-3. YUV Capture.....	29
Figure 9-4. DCC Update.....	29
Figure 9-5. Exposure Control for Live Tuning.....	29
Figure 9-6. White Balance Control for Live Tuning.....	30

Trademarks

Linux® is a registered trademark of Linus Torvalds.

Python® is a registered trademark of Python Software Foundation.

Microsoft® and Windows® are registered trademarks of Microsoft Corporation.

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 the best 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. To achieve that, the best 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 best 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 High-Definition Multimedia Interface (HDMI) display, using VPAC to process the raw images before sending them to the display:

```
gst-launch-1.0 v4l2src device=/dev/video-imx219-cam0 io-mode=dmabuf-import ! \
video/x-bayer, width=1920, height=1080, framerate=30/1, format=rggb10 ! \
tiovxisp sink_0::device=/dev/v4l-imx219-subdev0 \
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 ! \
kmsink driver-name=tidss sync=false
```

In the above pipeline, GStreamer element **tiovxisp** interfaces VPAC hardware and TI's reference software for imaging algorithms of AE and AWB (2A) 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/linear/dcc_viss_10b.bin for tuned ISP parameters
- dcc-2a-file=/opt/imaging/imx219/linear/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 AM6xA 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**: Make sure the camera sensor driver is properly integrated with the system and can capture raw images. Add exposure configuration needed by the auto-exposure algorithm. Add support for this sensor to GStreamer plug-ins.
4. **Generating Initial ISP configuration**: Run a Python® script to generate an initial (default, or baseline) DCC profile for VPAC configuration. This configuration can enable video streaming with image quality that is good enough to run the next step.
5. **Adjust camera mounting**: Run live video streaming using the DCC profile generated in the previous step and adjust camera module mounting to make sure images are captured with good positioning, focus, lighting, and so on.

6. **ISP basic tuning:** Capture raw images and perform basic tuning to achieve 70% to 80% of the best image quality under lab conditions.
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 with an IMX219 camera and an OX05B1S 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 the 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. In some instances, different chart sizes are needed depending on the setup.
- A light meter, a chroma meter (or both) 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.

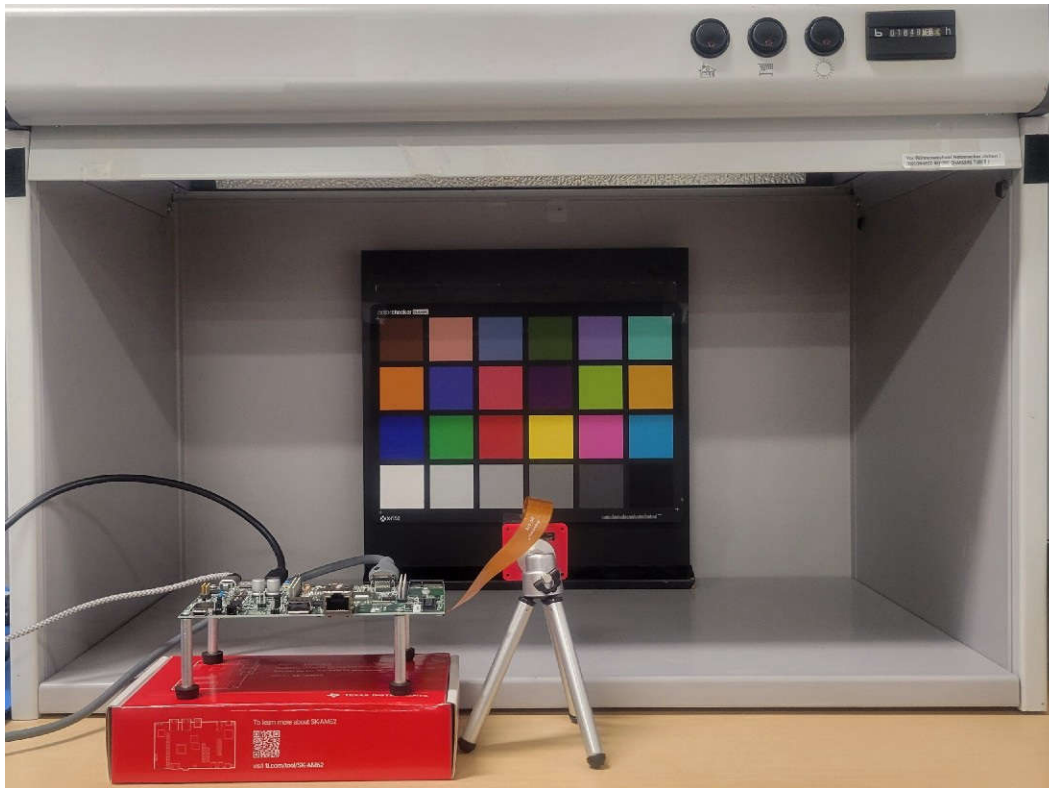


Figure 3-1. Tuning Hardware and Equipment Setup

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 report has TI's reference imaging algorithms for AE, AWB, and ISP control or DCC. This information also contains Python scripts that are needed in the tuning process.

4.3 ISP Tuning Tool

ISP tuning tools and imaging algorithms (for example, AE, AWB, and ISP control) are available from several imaging 3rd parties who may support customers for production. TI's TDA4 and AM6xA DCC tuning tool and imaging algorithms are also available for design references, and are used as an example in this report. 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 are not covered in this report.

5 Sensor Software Integration

To tune the ISP for a target sensor using the AM6xA Linux SDK, software needs to be developed to support the sensor in the SDK. Sensor driver development is beyond the scope of this report. The following sections first give an overview of the image pipeline software architecture on AM6xA, and then describe how to add the sensor driver to the SDK as well as how to modify the GStreamer plug-ins to support the sensor.

5.1 Overview of Image Pipeline Software Architecture

This section provides an overview of the ISP and the image pipeline software architecture on AM6xA, which helps understand the tuning procedure described later.

The ISP on the AM6xA devices provides common vision primitive functions for image data processing at the pixel level. There are three sub-modules in the ISP: Vision Imaging Subsystem (VISS), Lens Distortion Correction (LDC), and Multi-Scalar (MSC).

- **Vision Imaging Sub-System (VISS):** The VISS performs image processing on raw data, which includes wide dynamic range (WDR) merge, defect pixel correction (DPC), lens shading correction (LSC), global and local brightness and contrast enhancement (GLBCE), demosaicing, color conversion, and edge enhancement (EE). This block takes the raw image as input and produces YUV output.
- **Lens Distortion Correction (LDC):** The LDC engine is a YUV domain processor designed to perform perspective and geometric transforms. This can be used for creating several effects, such as lens distortion correction, epipolar rectification, and generic perspective transformation.
- **Multi-Scalar (MSC):** The MSC can generate up to 10 scaled outputs from a given input with various scaling ratios (between $1 \times$ and $0.25 \times$). Each of the 10 scaling operations can be configured to perform pyramid scale or inter-octave scale generation.

Out of these three sub-modules, VISS and LDC need to be tuned, and the VISS contains multiple processing blocks that need to be tuned individually.

Figure 5-1 shows the image processing pipeline software architecture for AM6xA, including the camera capture subsystem and ISP. The camera capture driver runs in Linux on the A53 or A72 core, and the ISP driver runs in RTOS on the R5 core. Though running on different CPU cores, camera capture and ISP can be interfaced with and integrated by the same framework of GStreamer or TIOVX.

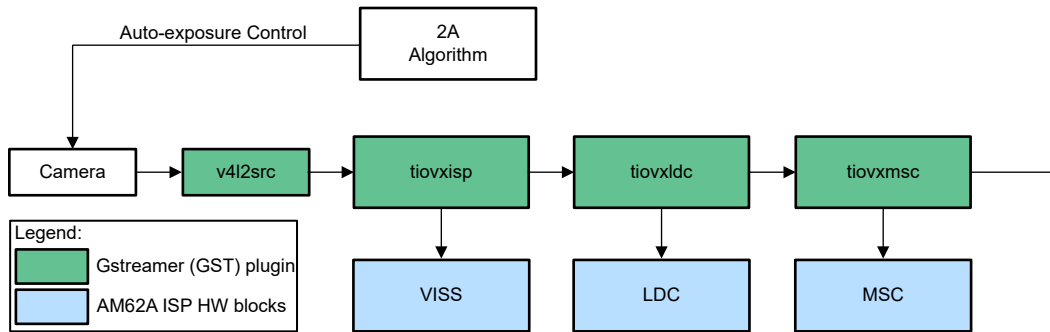


Figure 5-1. AM6xA Image Processing Pipeline

In this report, GStreamer examples are provided for integrating the image pipeline components. The camera capture subsystem is accessed through the GStreamer plug-in v4l2src. Each of the three ISP sub-modules is accessed through the corresponding GStreamer plug-in: tiovxisp, tiovxldc, and tiovxmsc.

5.2 Adding Sensor Driver to SDK

Refer to the [AM62A Academy](#) for instructions on adding a 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*.

In addition to adding the sensor driver to the SDK, the GStreamer plug-ins for the ISP need to be updated to support the sensor. This involves modifying and rebuilding two components in the target file system on the EVM, which is described in the following sections.

5.3 Updating TIOVX Modules

5.3.1 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 name of the sensor and id to the list in the `tiovx_init_sensor()` function. 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. Use the exact name string – `SENSOR_SONY_IMX219_RPI` – in the GStreamer pipeline and the DCC sensor ID (219) is used by the tuning tool in future steps.

5.3.2 Rebuild Modules

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

5.4 Update GStreamer Plug-in for VISS

To enable the ISP to support a new sensor, the GStreamer plug-in for VISS needs to be updated, including:

- Adding that sensor to the sensor list of the properties of the plug-in
- Adding the exposure setting for that sensor, which is needed by the auto exposure and auto white balance (2A) algorithm as shown in [Figure 5-1](#).

5.4.1 Update VISS Plug-in Property

In the target file system on the EVM, go to `/opt/edgeai-gst-plugins/ext/tiovx/gsttiovxisp.c`, and add the name of the sensor to the list (OX05B1S 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_OX05B1S\n"
        ...
```

5.4.2 Add Exposure Setting for 2A Algorithm

As [Figure 5-1](#) shows, the exposure of the camera sensor is automatically controlled by the 2A algorithm. The 2A algorithm adjusts the exposure based on the H3A statistics provided by the ISP. Both the exposure time and gain can be adjusted by the 2A algorithm. Every sensor has a minimum and maximum exposure time and gain operated from within. These specifications are found in the data sheet and need to be passed to the 2A algorithm through the VISS plug-in. A new function needs to be added to provide this information to the 2A algorithm. This function is usually named `get_<sensor>_ae_dyn_params()` in `gsttiovxisp.c`.

5.4.2.1 Gain

The minimum and maximum gains are usually specified in the data sheet as Nx gain and Mx gain, where N is the minimum and M is the maximum. The corresponding values in the gain register are usually also given. For example, a certain sensor may specify the following:

- Minimum gain: 1 ×, gain_register value is 16
- Maximum gain: 15.5 ×, gain_register value is 248

Since the DCC live tuning tool uses 1024 for 1 × gain to calculate and display the gain in floating point numbers, it is preferred to set the gain in this convention and map the gain between 2A and the sensor driver. Function `get_<sensor>_ae_dyn_params()` has the following setting for minimum and maximum gain:

```
p_ae_dynPrms->analogGainRange[count].min = 1024; /* 1x gain */
p_ae_dynPrms->analogGainRange[count].max = 15872; /* 15.5x gain */
```

Therefore the gain value used by the 2A algorithm is essentially 64 times the gain value used by this specific sensor. After the 2A algorithm returns the gain to be set for the sensor, that gain value needs to be divided by 64 before being sent to the sensor. This mapping is done in function `gst_tiovx_isp_map_2A_values()`:

```
*analog_gain_mapped = analog_gain / 64; /* 64 = 1024 / 16 */
```

5.4.2.2 Exposure Time

For exposure time, the minimum and maximum are usually specified as a number of row periods. For example, a certain sensor can have the following specification:

- Minimum exposure time: 6 row periods
- Maximum exposure time: (frame length – 30) row periods

Using resolution 2592 × 1944 as an example, the frame length is 1944 rows plus vertical blanking. Assuming the vertical blanking is 184, the frame length is 2128 rows. Therefore, the maximum exposure time is 2128 – 30 = 2098 row periods. Check the sensor driver and make sure the exposure time is written to the register according to the data sheet.

Since the DCC live tuning tool displays the exposure time in micro seconds, the preferred practice is to set the exposure time in micro seconds for 2A and perform a mapping between 2A and the sensor driver. The mapping depends on frame size and frame rate. For example, for 2592 × 1944 resolution and 60 fps, the minimum and maximum exposure time can be set as below in function `get_<sensor>_ae_dyn_params()`:

```
p_ae_dynPrms->exposureTimeRange[count].min = 47; /* 6*16.67/2128*1000 micro sec */
p_ae_dynPrms->exposureTimeRange[count].max = 16435; /* (2128-30)*16.67/2128*1000 micro sec */
```

Accordingly, the mapping from micro seconds to number of row periods (which is passed to the sensor driver) is provided in function `gst_tiovx_isp_map_2A_values()`:

```
*exposure_time_mapped = (int) ((double)exposure_time * 2128 * 60 / 1000000 + 0.5);
```

5.4.2.3 Other Parameters

Other parameters needed by the 2A algorithm can use the following default values:

```
/* setting brightness target and range: range is always [target-threshold, target+threshold]. -
numbers in 0~255 range
*/
p_ae_dynPrms->targetBrightnessRange.min = 40; /* lower bound of the target brightness range */
p_ae_dynPrms->targetBrightnessRange.max = 50; /* upper bound of the target brightness range */
p_ae_dynPrms->targetBrightness = 45; /* target brightness */
p_ae_dynPrms->threshold = 5; /* maximum change above or below the target brightness */
p_ae_dynPrms->enableB1c = 0; /* not used */

/* setting exposure and gains */
p_ae_dynPrms->exposureTimeStepSize = 1; /* step size of automatic adjustment for exposure time */
p_ae_dynPrms->digitalGainRange[count].min = 256; /* digital gain not used */
p_ae_dynPrms->digitalGainRange[count].max = 256; /* digital gain not used */
```

5.4.3 Rebuild Plug-ins

After changing the source code, `/opt/edgeai-gst-plugins/ext/tiovx/gsttiovxisp`, rebuild and reinstall the GStreamer plug-in by running the `/opt/edgeai-gst-apps/scripts/install_gst_plugins.sh` script.

5.4.4 Verify New Sensor in GStreamer Plug-in

After all the above changes, verify that the new sensor name is listed in the properties of GStreamer plug-in `tiovxisp`. For example, `OX05B1S` is shown below:

```
root@am62axx-evm:~# gst-inspect-1.0 tiovxisp
...
  sensor-name : TIOVX camera sensor string ID. Below are the supported sensors
                SENSOR_OX05B1S
```


6 Tuning Procedure

This section provides the detailed ISP tuning procedure. In summary, the procedure includes the following steps:

1. Verify that the camera operates properly
2. Create an initial (default, or baseline) ISP configuration profile using an automated script
3. Perform lab tuning under controlled lighting conditions. Tuning can be divided into 2 steps: basic tuning and fine tuning.
4. Perform production tuning in real-world scenes where the sensor is to be deployed. This step is not covered by this report.

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/video3
    /dev/video4
...
    /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

4-0010 in the `media-ctl` output is the I2C bus address for the sensor and this value can 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 is shown by the `media-ctl` command as above:

```

root@am62axx-evm:~# media-ctl -v '"imx219 4-0010":0 [fmt:SRGGB10_1X10/1920x1080 field:none]'
root@am62axx-evm:~# gst-launch-1.0 -v v4l2src num-buffers=5 device=/dev/video3 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 can be either overexposed or underexposed since the default exposure time and gain in the sensor does not necessarily match the lighting environment where these images are captured.

6.2 Enable Camera Streaming With Initial VPAC Configuration

After verifying that the camera and the driver work functionally and the GStreamer plug-ins 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 plug-ins for VPAC work properly. The live streaming also makes the 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 [Section 6.2.1](#) through [Section 6.2.3](#).

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 referenced.

Specify the following parameters in this configuration file:

- **Camera sensor information**
 - `SENSOR_ID`: DCC ID of the sensor (**this must match the hard coded "dccid" value in `tiiovx_sensor_module.c`**, as described in [update GStreamer plug-in for VISS](#))
 - `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`, for example, `../../../../sensor_drv/src/<sensor name>`.
- **Raw image format information**
 - `SENSOR_WIDTH`: sensor image width
 - `SENSOR_HEIGHT`: sensor image height
 - `COLOR_PATTERN`: Bayer pattern, where 0=RGGB; 1=GRBG; 2=GBRG; 3=BGGR, 4=MONO, 10=RGGI, 11=GRIG, 12=BGGI, 13=GBIG, 14=GIRG, 15=IGGR, 16=GIBG, 17=IGGB
 - `WDR_MODE`: sensor mode. 0=linear, 1=wdr
 - `BIT_DEPTH`: sensor pixel bit depth
 - `WDR_BIT_DEPTH`: WDR raw sensor image bit depth after decomanding, typically 20 or 24
 - `WDR_KNEE_X` and `WDR_KNEE_Y`: WDR decomanding knee points (must be separated by commas and have no spaces in between)
 - `BLACK_PRE`: Sensor black level to subtract before decomanding
 - `BLACK_POST`: Sensor black level to subtract after decomanding
 - `GAMMA_PRE`: GAMMA value for compressing 20-bit and 24-bit WDR raw to 16-bit ISP internal bit width. Typically around 50 (0.5) for 24-bit WDR sensors and 70 (0.7) for 20-bit sensors
 - `H3A_INPUT_LSB`: LSB location for H3A input bit range (from bit-`H3A_INPUT_LSB` to bit-`H3A_INPUT_LSB+9`)

Note

The above configuration information must match the actual sensor format.

Following is a configuration example for IMX219. The WDR related parameters are not used but must be included for the script to run.

```

SENSOR_ID 219
SENSOR_NAME IMX219
SENSOR_DCC_NAME SENSOR_IMX219_RPI
PRJ_DIR ../../../../sensor_drv/src/imx219_output
SENSOR_WIDTH 1920
SENSOR_HEIGHT 1080

COLOR_PATTERN 0
WDR_MODE 0

BIT_DEPTH 10

WDR_BIT_DEPTH 20

WDR_KNEE_X 0,512
WDR_KNEE_Y 0,2048

BLACK_PRE 0
BLACK_POST 0

GAMMA_PRE 70

H3A_INPUT_LSB 0

```

Once the configuration file is created, run the `ctt_def_xml_gen.py` Python script 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 are in the folder `$PRJ_DIR/dcc_xml`. A script file `generate_dcc.sh` is also generated in the xml folder. For example, below is the content of the xml folder for IMX219 (linear mode):

```
$PRJ_DIR/dcc_xmls/linear$ 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_b1c.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 the `generate_dcc.sh` script 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`.

```
$PRJ_DIR/dcc_bins$ls-l
dcc_2a.bin
dcc_ldc.bin
dcc_viss.bin
```

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 the `/opt/imaging/<camera>` folder. For example, IMX219 has the following pre-built binary files in the SDK:

```
root@am62axx-evm:~# ls /opt/imaging/imx219/linear
dcc_2a_10b_1920x1080.bin dcc_viss_10b_1920x1080.bin
```

Then use the `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:SRGGB10_1X10/1920x1080 field:none]'
root@am62axx-evm:~# gst-launch-1.0 v4l2src device=/dev/video-imx219-cam0 io-mode=dmabuf-import ! \
video/x-bayer, width=1920, height=1080, framerate=30/1, format=rggb10 ! \
tiovxisp sink_0::device=/dev/v4l-imx219-subdev0 \
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 ! \
kmsink driver-name=tidss sync=false
```

Even though the image quality is not fully tuned when using the initial configuration, this can validate that the GStreamer plug-ins for the ISP are working properly for the new sensor. The output video gives the 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 the camera position to make sure the color checker is fit exactly into the field of view (FOV) of the camera.

For a certain resolution, the sensor driver can crop the image before sending the image to the ISP. For example, at resolution 1920×1080 , the IMX219 driver crops the image and the image looks zoomed in on the display. For this case, a color checker of a smaller size can work better.

7 Perform Basic Tuning

In general, ISP tuning requires users to first capture a set of raw images or YUV images (or both types of images) under controlled lighting conditions and use a tuning tool to adjust ISP parameters and AWB calibration. In this section, the TDA4 and AM6xA 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.

7.1 Launch the Tuning Tool and Create a Project

Locate and run the installed tuning tool executable in Microsoft® 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 7-1](#) shows an example for IMX219 in 1080p streaming mode.

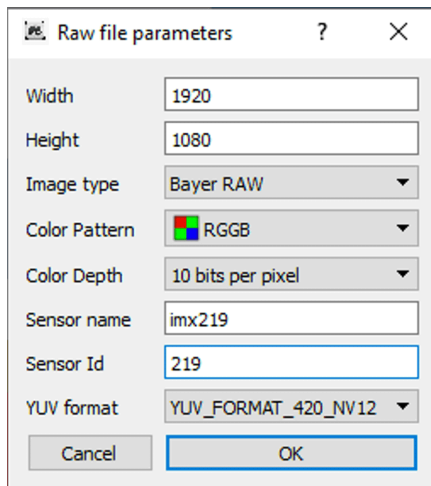


Figure 7-1. Sensor Raw Image Parameters for Tuning

Note

The image properties must match with what is used for capturing raw images, and the sensor ID must match with the hard coded value in the GStreamer plug-ins.

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 conditions using the command for [camera capturing verification](#). Select this image file in the *Browse* window of the tuning tool and the image ought to display in the *Preview window*. For example, below is an IMX219 raw image at 1920 × 1080 with 10 bits per pixel, displayed in the tuning tool.

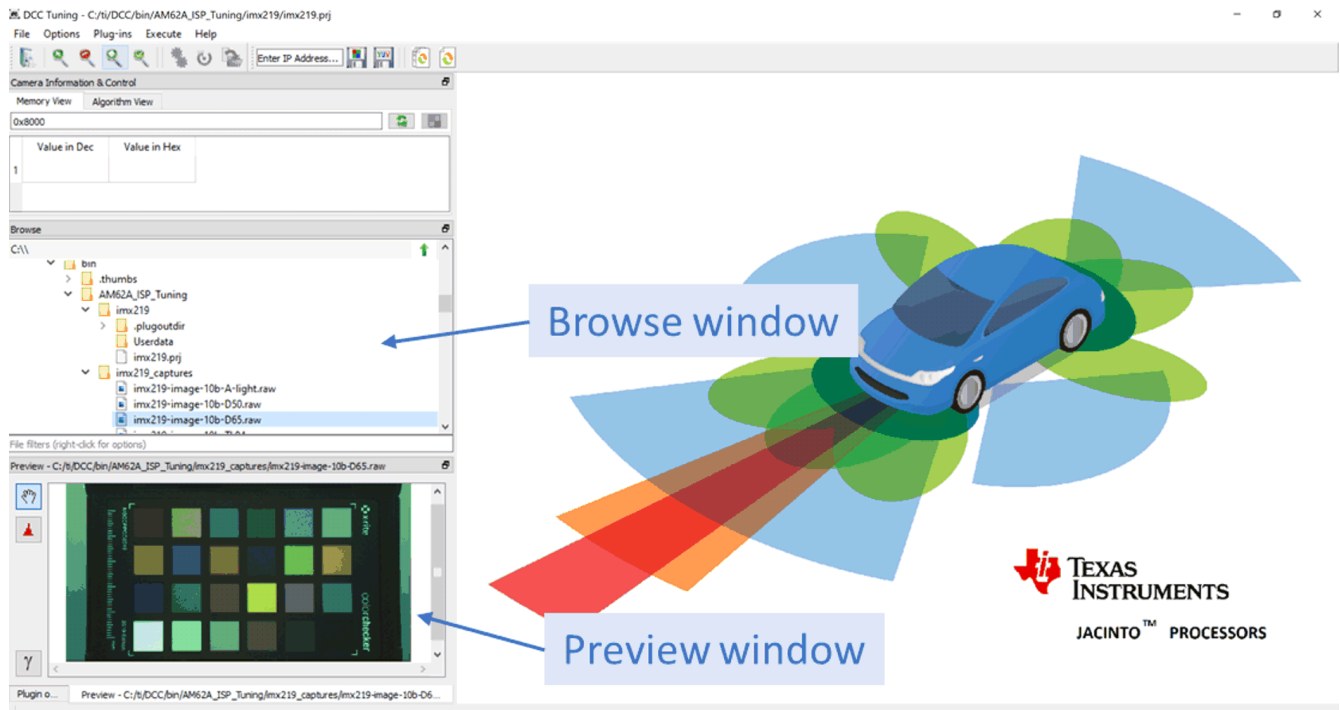


Figure 7-2. Image Preview by DCC Tuning Tool

Note

Only raw images captured with 10-, 12-, or 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 bits per pixel first and then previewed and used in the tuning tool.

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

7.2 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 tuning of the ISP blocks in independent groups, with each group containing one or more ISP blocks. The tuning groups are referred as plug-ins, as shown in the tuning tool menu *Plug-ins*:

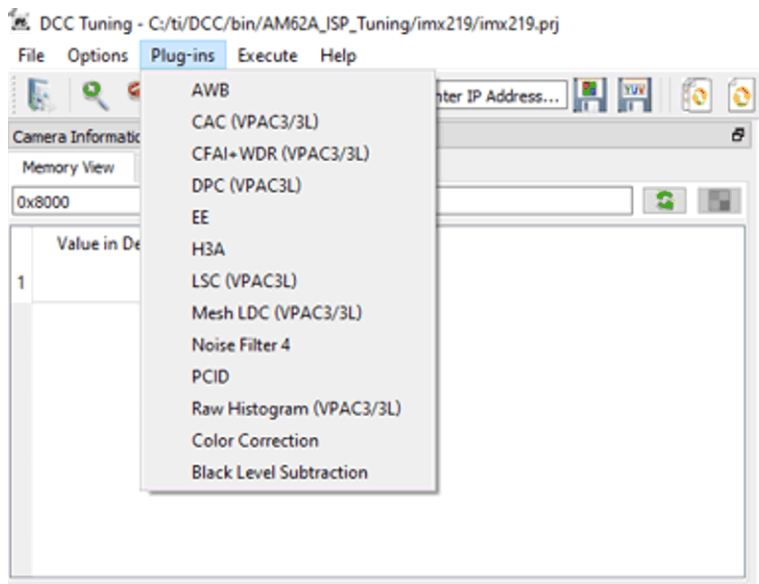


Figure 7-3. DCC Tuning Tool Plug-ins

Note

Each plug-in has a tuning guide, available from the Help → Documentation drop-down menu. Reference these guides during the tuning.

In general, tune these plug-ins in the same order of ISP blocks when processing the raw image. Below is a recommended tuning order:

1. Black Level Subtraction (BLC): 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

There is a slight difference in tuning RGB-only sensors and RGB-IR sensors:

- For RGB-only sensors: BLC, H3A, AWB, CCM, NSF4, EE, and so forth.
- For RGB-IR sensors: H3A, PCID, AWB, CCM, NSF4, EE, and so forth.

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

In this application note, the OX05B1S sensor is used to illustrate the tuning of PCID, and the IMX219 sensor is used for illustrating the tuning of the rest plug-ins including black level, AWB, color correction, and so forth. 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.

7.3 Black Level Subtraction

The Black Level Subtraction (BLC) plug-in tuning is only needed for RGB-only sensors. For RGB-IR sensors, this plug-in needs not to be tuned because of the IR subtraction performed by PCID.

For linear sensors including the IMX219, subtract the black level or pedestal 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, measure the 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 ought to 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 is displayed in the *Advanced params* tab of the upper right window.

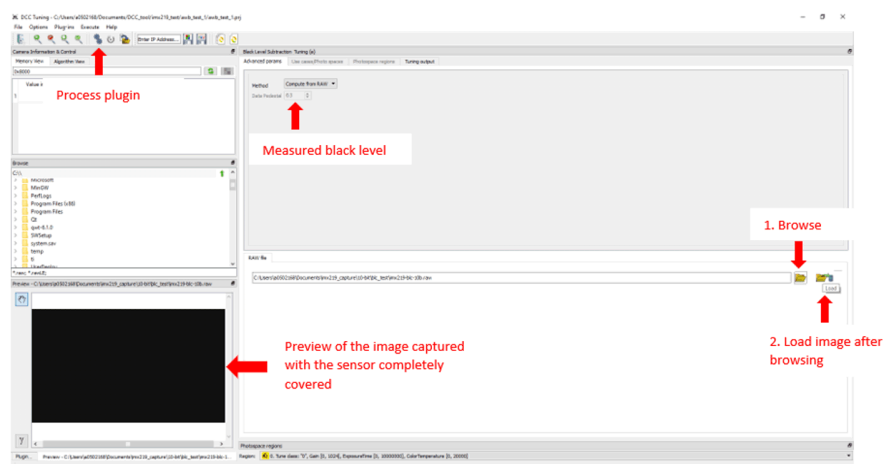


Figure 7-4. Black Level Subtraction Tuning

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* plug-in in the tuning tool. See the user guide of the plug-in 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 plug-in. The XML files are located in the `.plugoutdir\XML` folder under the project folder created in [Section 7.1](#). For Black Level Subtraction, there is only one output XML file: `imx219_viss_blc.xml`. Replace the same XML file generated from [initial configuration](#). Then rerun the Python script to generate new DCC binary files as described in [Generate DCC Binary Files](#). Use the newly-generated DCC binaries to improve streaming quality in the next step. Complete this action 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/video3 io-mode=dmauf-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 7-5 shows an example of the captured images before Black Level Subtraction tuning (using the initial configuration) and Figure 7-6 shows the image after Black Level Subtraction tuning (compare the appearance of black).

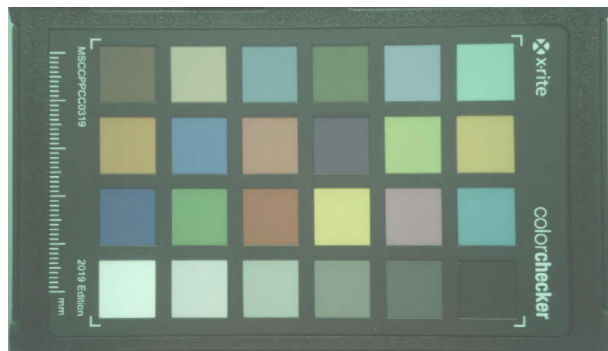


Figure 7-5. Image Before Black Level Subtraction



Figure 7-6. Image After Black Level Subtraction

7.4 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 and OX05B1S, only AE and AWB are relevant. The `ctt_def_xm1_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 such that H3A, AE, and AWB can be programmed properly. Therefore, this parameter must be set correctly.

The Python script configures the sensor or 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 cases when fine adjustment of H3A settings is required, please follow the TRM of the SoC and the user guide in tuning tool.

7.5 PCID

For RGB-IR sensors including the OX05B1S, tune the PCID plug-in first to generate the output 16-bit Bayer raw images that are used to calibrate AWB, CCM, NSF4, and so forth.

To tune PCID, raw images need to be captured at different lighting conditions. The specifics of these lighting conditions are outlined in [Capture Raw Images for Different Lighting Conditions](#). Below is a summary of tuning this plug-in. For more details, refer to the PCID plug-in guide from the help menu.

- Load an input 16-bit RAW image
- Enter the proper parameter values
 - When choosing the Input and Output Format, verify that the Output Format matches with the configured pipeline. For example, OX05B1S has the color format of BGGR. The Input Format is set to "2:BGGR", Output Format is set to "1: Interpolate R at IR positions and B at R locations" and the GStreamer pipeline must be configured as BGGR.
- *Process* the plug-in to generate the output Bayer pattern with IR subtraction in 16 bit
- Repeat this process for all lighting conditions
- *Export DCC profile binary* to generate XML and binary files

Note

When tuning RGB-IR sensors, set the black level to 0 while tuning the plug-ins that require the black level value since the black level has already been removed in IR subtraction.

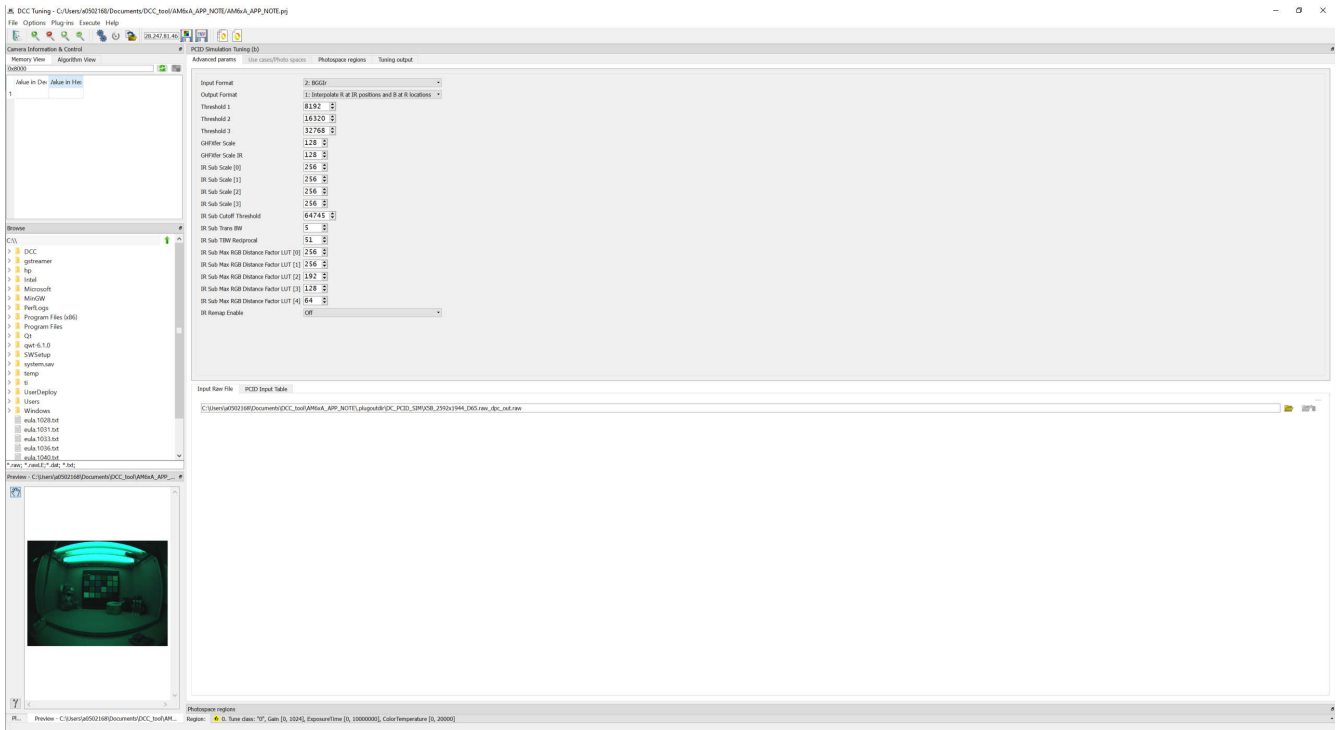


Figure 7-7. PCID Tuning

An example of GStreamer pipeline using the OX05B1S follows:

```
gst-launch-1.0 -v v4l2src device=/dev/video4 io-mode=dmabuf-import ! \ video/x-bayer,
width=2592, height=1944, framerate=30/1, format=bggi10 ! queue ! \ tiovxisp sink_0::pool-size=4
sink_0::device=/dev/v4l-subdev2 sensor-name="SENSOR_OX05B1S" \ dcc-isp-file=/opt/imaging/ox05b1s/
linear/dcc_viss.bin sink_0::dcc-2a-file=/opt/imaging/ox05b1s/linear/dcc_2a.bin format-msb=9 !
queue ! \ tiovxldc dcc-file=/opt/imaging/ox05b1s/linear/dcc_ldc.bin sensor-name=SENSOR_OX05B1S !
\ video/x-raw, format=NV12, width=2592, height=1944 ! queue ! \ tiovxmultiscaler src_0::pool-
size=4 target=1 ! video/x-raw, format=NV12, width=1920, height=1080 ! queue ! \ tiperfoverlay
location=perf_logs num-dumps=10 ! queue ! kmssink driver-name=tidss force-modesetting=true
sync=false
```

Figure 7-8 and Figure 7-9 show an example of the captured RAW image completely unprocessed by the ISP and the output image of PCID tuning in Bayer pattern format.



Figure 7-8. RAW Image Input to PCID

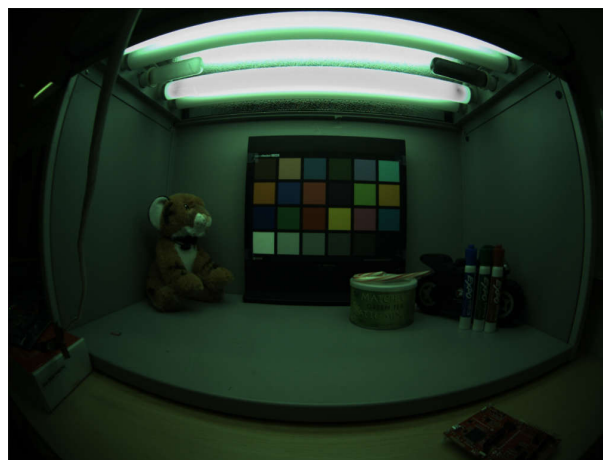


Figure 7-9. PCID Output Image in Bayer Pattern

7.6 Auto White Balance (AWB)

7.6.1 Capture Raw Images for Different Lighting Conditions

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

- Set up lighting conditions to: D65, D50, TL84, and A Light (one at a time)
- Place the 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 the monitor is bright enough without apparent saturation or clipping on the white patch. This process is usually done in a few seconds.)
 - Run live video streaming with the initial configuration
 - Raw images can 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 the AM62A EVM and PC are connected to the same Ethernet, the tuning tool can also capture raw images from EVM directly over Ethernet. From the tool bar of the DCC tuning tool, enter the IP address of the EVM and capture raw images as shown below. For more details, follow the tuning tool user's guide.

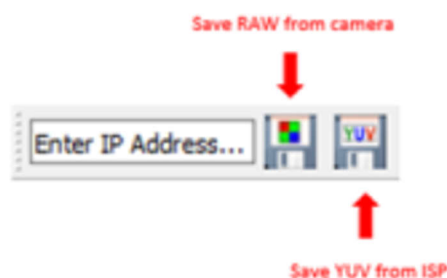


Figure 7-10. Live Capture Feature of DCC Tuning Tool

Figure 7-11 through Figure 7-14 are examples of well-exposed captures for all of the above-mentioned lighting conditions (note the color differences caused by the color of lighting):

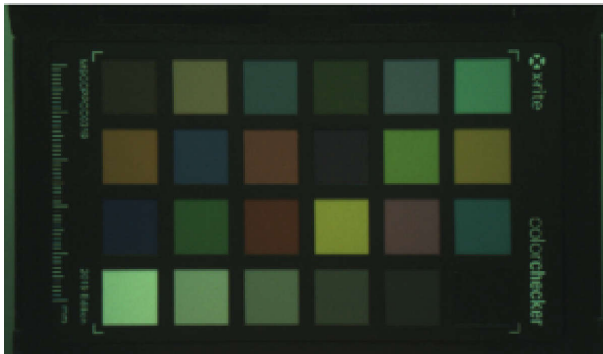


Figure 7-11. Color Chart Image Captured With D50 Lighting

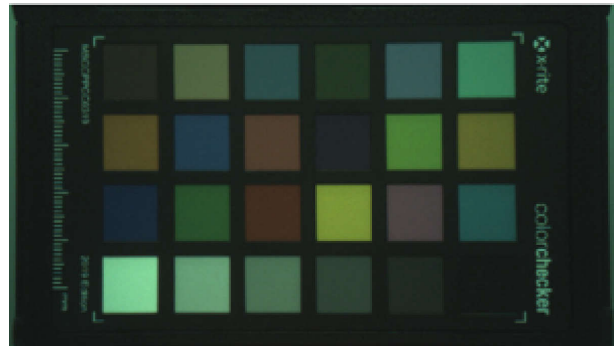


Figure 7-12. Color Chart Image Captured With D65 Lighting

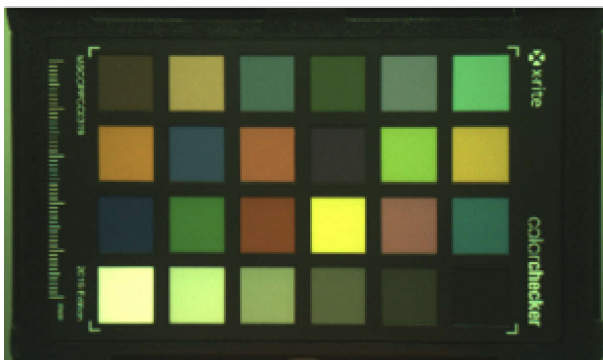


Figure 7-13. Color Chart Image Captured With TL84 Lighting

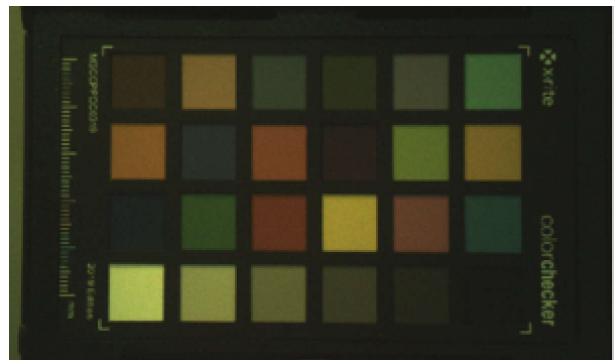


Figure 7-14. Color Chart Image Captured With A-light

7.6.2 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 plug-in 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 is 6500 for the D65 lighting condition.
- Exposure, Gain, and Aperture: these values are not used for AWB; therefore, these values can be ignored.
- Black Level: this must be the pedestal value that was measured in the Black Level Subtraction plug-in. In this example, IMX219 has a measured black level of 63 in 10-bit mode.

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 clockwise order.
- After the four corners are selected, the tool automatically identifies the 24 patches and displays the selection of each patch as shown below.

Figure 7-15 through Figure 7-16 show an example of importing one raw image for AWB tuning.

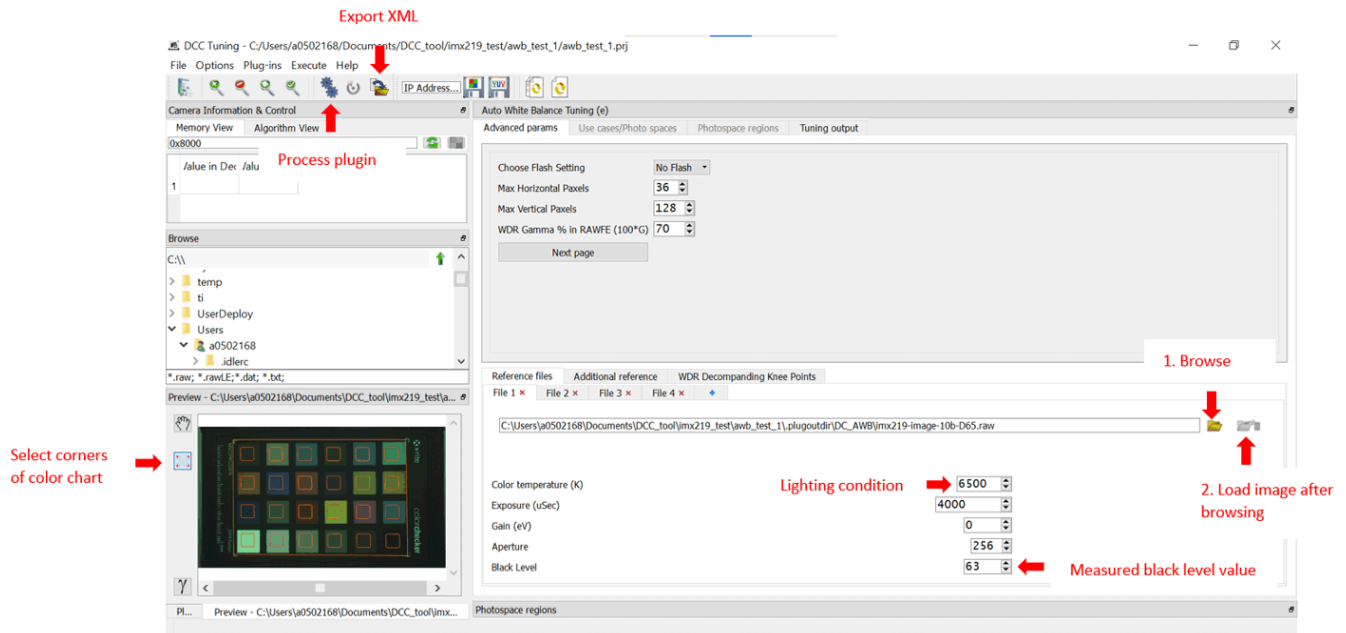


Figure 7-15. Auto White Balance Tuning

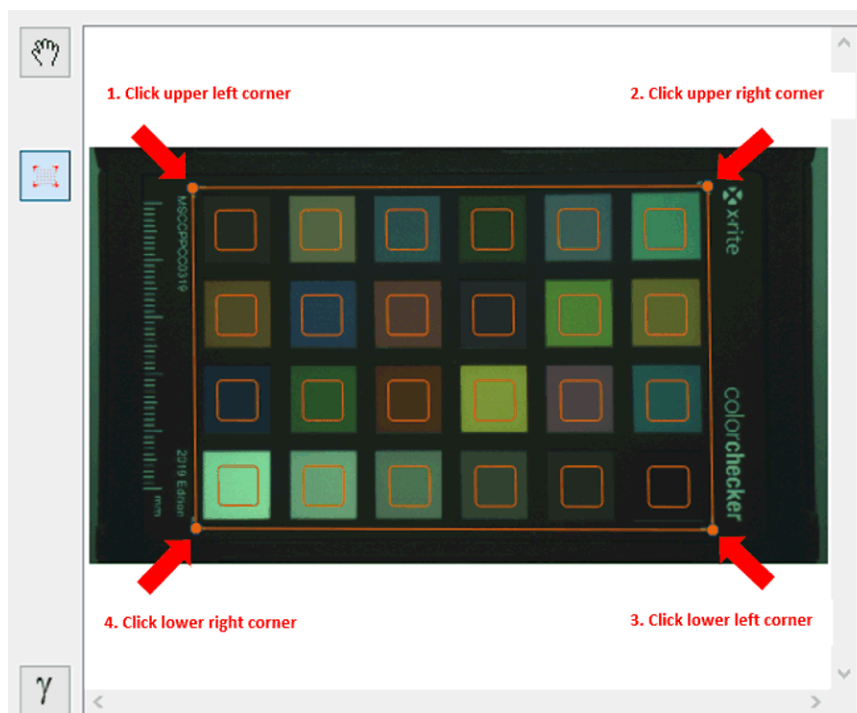


Figure 7-16. Choosing Corners of Color Checker Chart

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

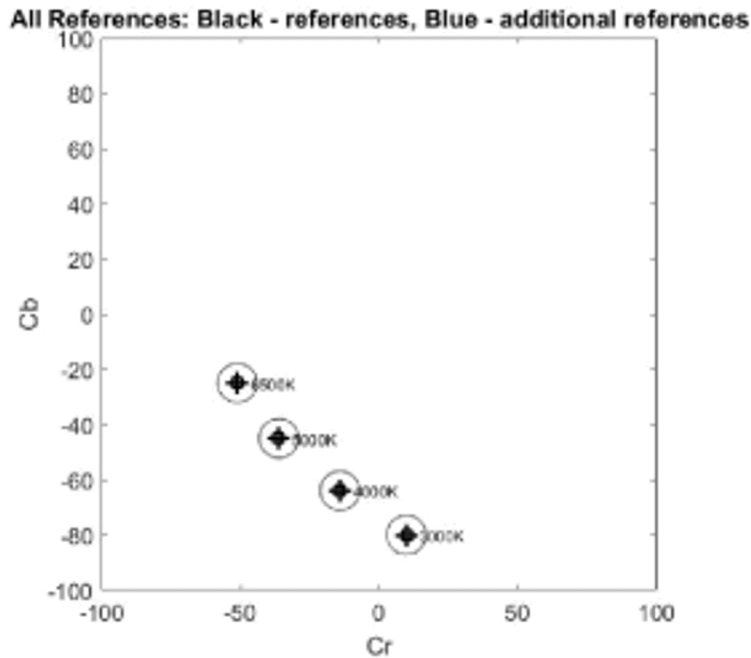


Figure 7-17. Auto White Balance Tuning Results

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 7-18](#) and [Figure 7-19](#) show images captured before AWB tuning and after AWB tuning (all gray patches appear neutral in the image after AWB tuning).



Figure 7-18. Image Before Auto White Balance Tuning



Figure 7-19. Image After Auto White Balance Tuning

7.7 Color Correction

To get the correct color output, tune the Color Correction plug-in. The raw images captured for AWB tuning can be used for tuning this plug-in. Below is a summary of this tuning step. For more details, see the Color Correction plug-in 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 can use any reference image.
- 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 plug-in and *Export DCC profile binary* to generate XML and binary files, as shown in [Figure 7-20](#) and [Figure 7-21](#).

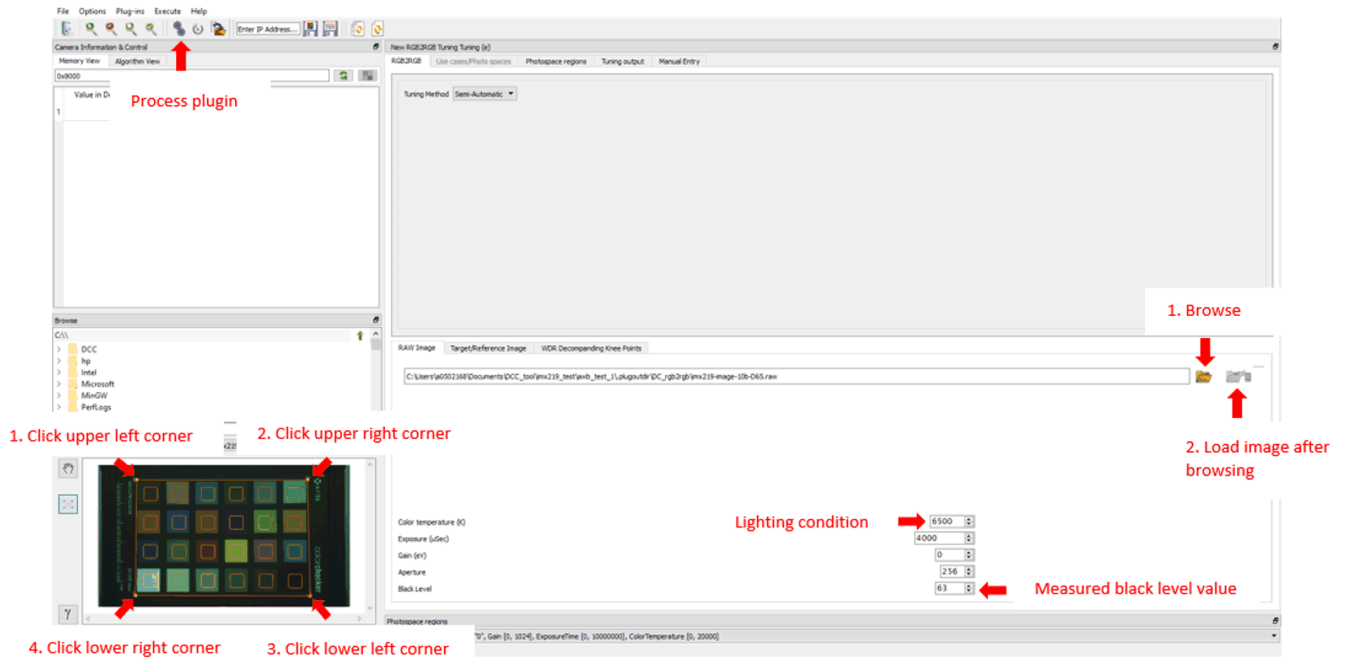


Figure 7-20. Color Correction Tuning

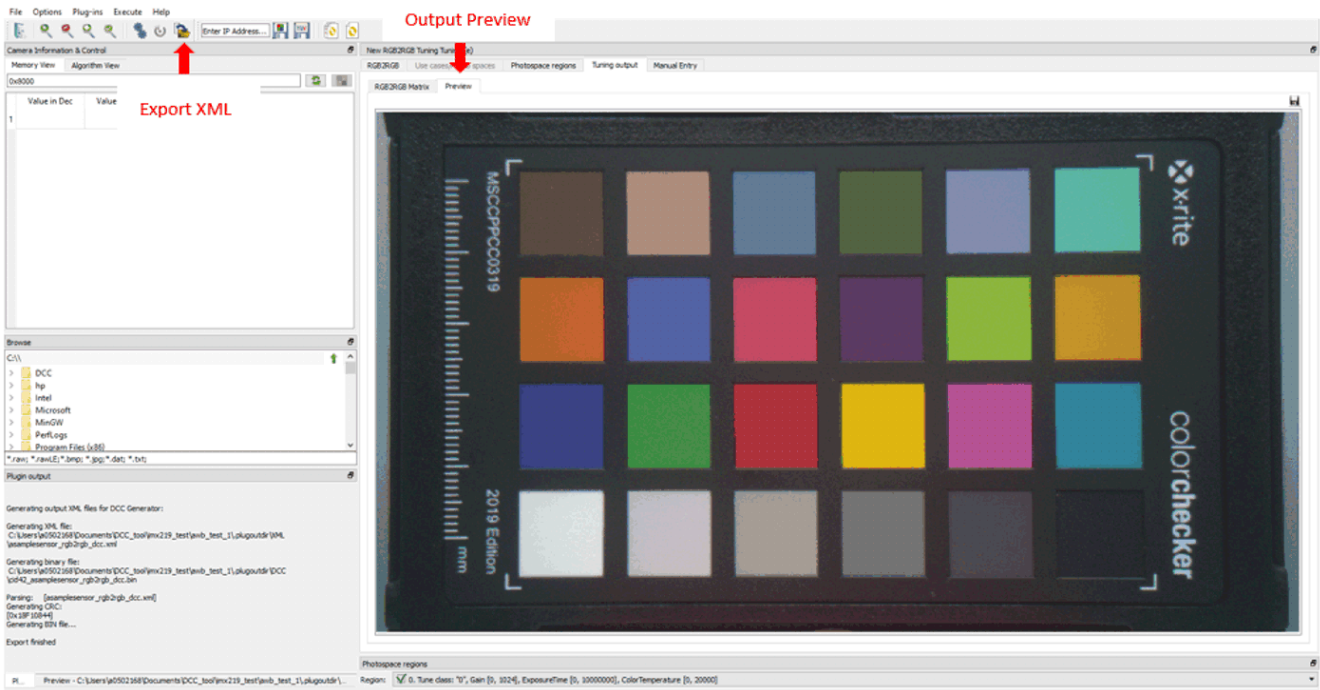


Figure 7-21. Color Correction Tuning Output

As done in AWB tuning, capture images using the newly generated DCC binaries. [Figure 7-22](#) and [Figure 7-23](#) show images before and after Color Correction tuning (note the corrected color in the image after Color Correction tuning).



Figure 7-22. Image Before Color Correction Tuning



Figure 7-23. Image After Color Correction Tuning

8 Perform Fine Tuning

Once the Black Level Subtraction, Hardware 3A, PCID (for RGB-IR sensors), Auto White Balance, and Color Correction are tuned properly, the ISP can achieve about 70% to 80% of the best image quality in light box lighting conditions. Other plug-ins can be tuned additionally to further improve the image quality, including:

- Edge Enhancement (EE) for better sharpness of the output image
- Noise Filter 4 (NSF4) for the suppression of noise in dark conditions
- Mesh Lens Distortion Correction (LDC) for removing lens distortions
- Color Filter Array Interpolation + Wide Dynamic Range (CFAI + WDR) for demosaicing and WDR sensors
- Lens Shading Correction (LSC) for removing lens shading

8.1 Edge Enhancement (EE)

The EE module parameters and LUT are automatically generated by the plug-in based on user inputs for threshold and slope.

Note

For EE modules to take effect on AM6xA, enable `ee_mode` as shown in the following:

- Open `/opt/edgeai-tiovx-modules/src/tiovx_viss_module.c` and find `ee_mode`
- Enable edge enhancement by setting `ee_mode` as below:
 - `obj->params.fcp[0].ee_mode = TIVX_VPAC_VISS_EE_MODE_Y8;`

- Capture a YUV image after tuning color correction. Load this image to EE plug-in of the tuning tool.
- Set the Tuning Method to Semi-Automatic
- Process the plug-in to switch to the EE tuning GUI. [Figure 8-1](#) shows the EE tuning GUI and details of each control
- Enable or disable the *Edge Enhancer* using the *On* or *Off* radio button in the *Enable* section
- Disable the *Edge Sharpener* by setting the *Gain* to '0'
- To see EE take effect, select one of the *Image Sources* in the *Semi-Automatic* tuning GUI and select the *Apply* button.

[Figure 8-2](#) and [Figure 8-3](#) show examples of the tuning tool output of *before* and *after* EE tuning, respectively.

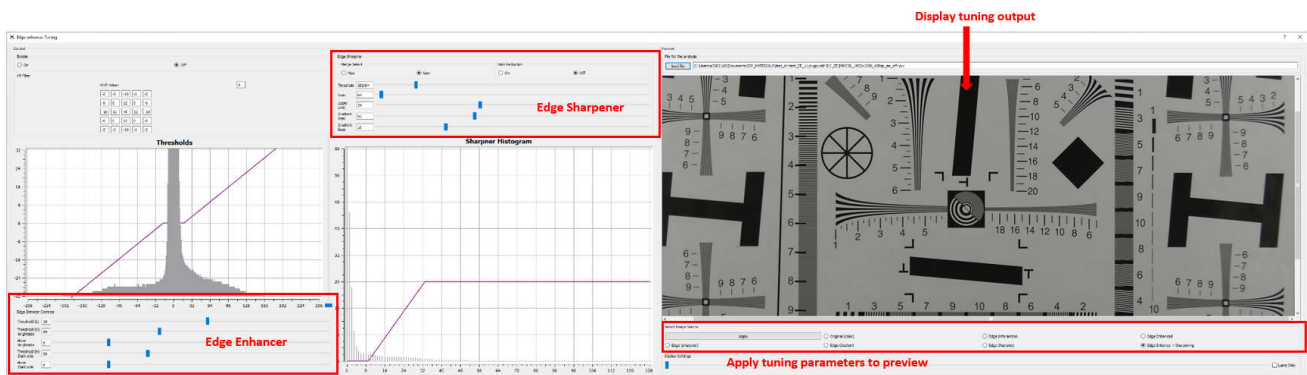


Figure 8-1. EE Semi-Automatic Tuning GUI

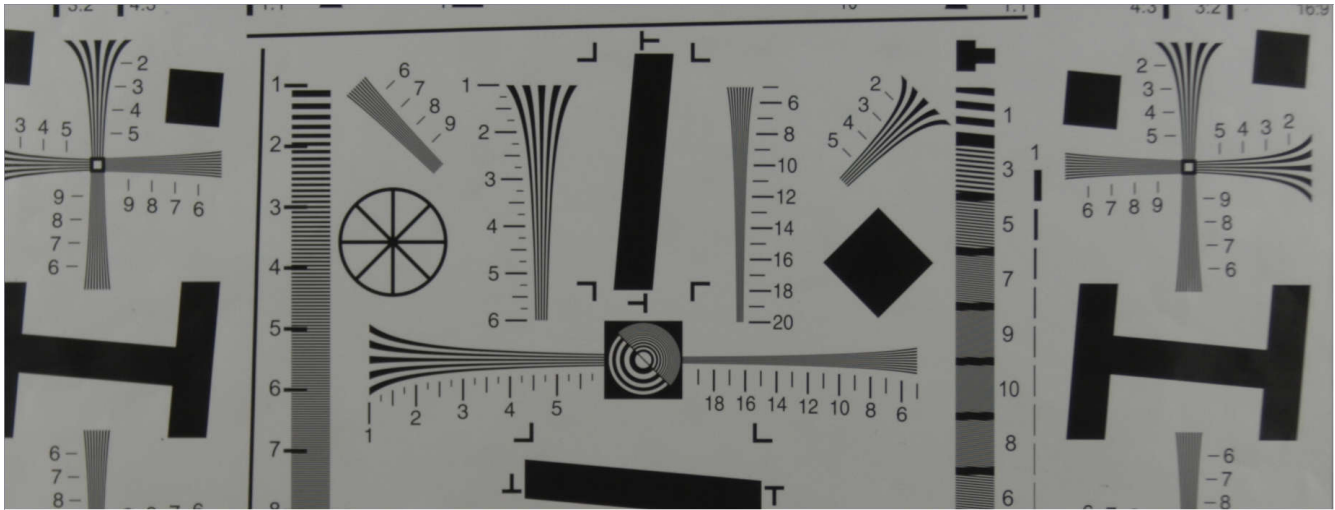


Figure 8-2. YUV Image Input Before EE Tuning

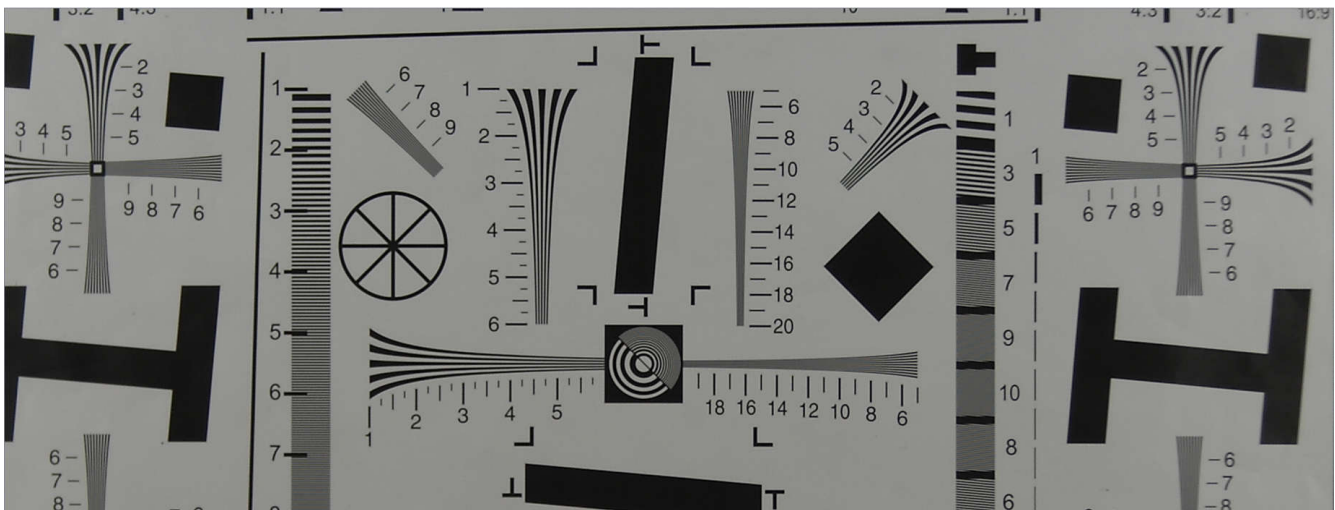


Figure 8-3. YUV Image Output After EE Tuning

8.2 Noise Filter 4 (NSF4)

Noise filtering reduces or removes noise in images without blurring details or other important image features. Below is a summary of tuning this plug-in. For more details, see the NSF4 plug-in guide from the *Help* menu.

- Load a RAW image (for example, one of the raw images used for AWB tuning)
- Set the Tuning Method to Semi-Automatic
- Select the four corners of the color checker chart (starting with the top left corner and continuing clockwise) as done in AWB tuning
- Load RAWFE decompanding LUT. The `lut_rawfe_pw1_vshort.txt` is found in the XML output folder when using the python script to [generate configuration files](#)
- Set RAWFE decompanding LUT to On with 12-bit LUT
- Process the plug-in to switch to the NSF4 tuning GUI. [Figure 8-4](#) shows the NSF4 tuning GUI and details of each control
- Adjust the parameters in *Display Control*
 - *X/Y range*: zooms into the plot area
- The noise chart *NF Tuning Plot* displays the noise samples as small circles. The blue line curve is plotted according to the current X and Y THR values set in *THR Tuning*. Manually adjust the THR (Y) values by moving the sliding bars or by typing in numbers such that the noise sample circles and blue curve match.

- Do not allow the *Overall Strength* to be greater 1.00. If the lines become blurred, lower the overall strength as the noise filtering is too strong.
- To see NFS4 take effect, select *Run NF* and results are shown once *Output Image* option is enabled.
- Repeat previous steps until output is tuned to your preference.
- When satisfied, click the *Finish* button to save tuning *Export DCC profile binary* to generate the output XML and binary files

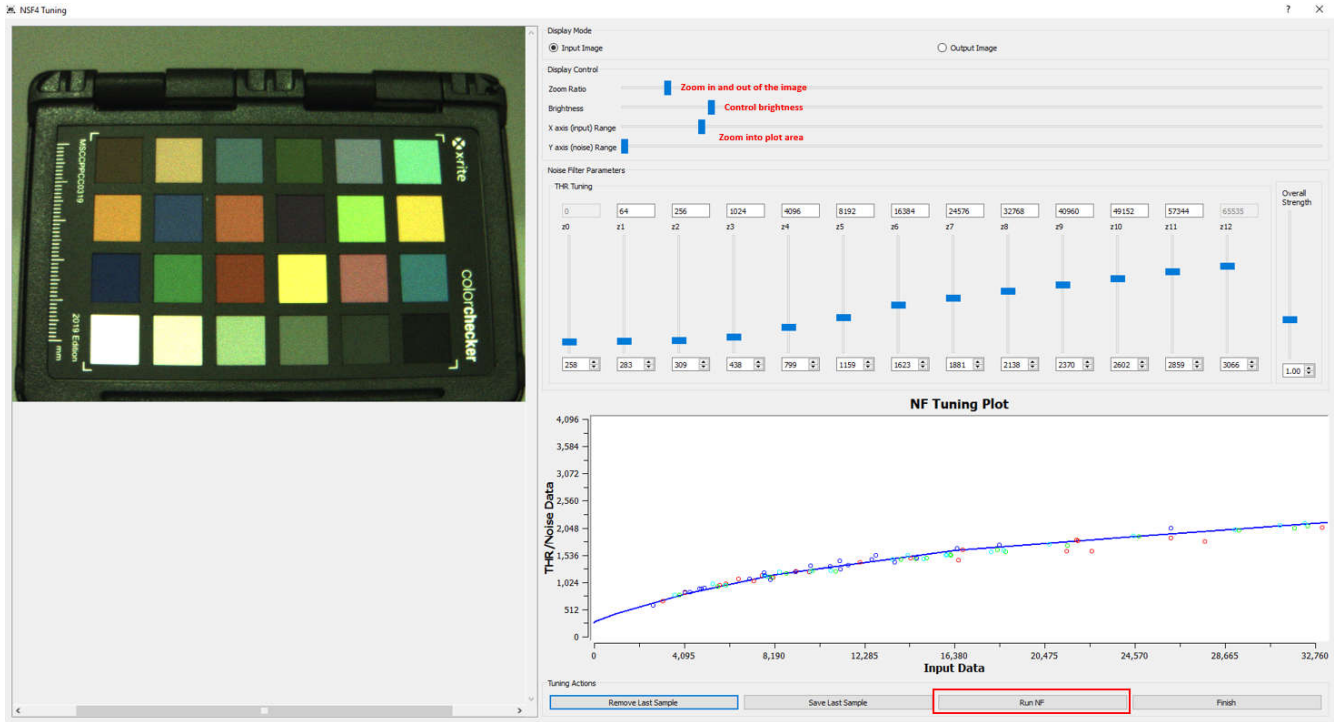


Figure 8-4. NSF4 Tuning GUI

Additional tuning that can be performed but not covered in this revision:

- Mesh LDC: see this [FAQ](#)
- CFAI + WDR
- LSC

9 Live Tuning

Live tuning features allow the tuning tool to interact with the target system at run time over the Ethernet interface.

Complete the following to enable live tuning on AM6xA:

1. In the target file system on the EVM, navigate to `/opt/edgeai-tiovx-modules/CMakeLists.txt`
2. Modify **ENABLE_DCC_TOOL** *Enable DCC Tuning Tool Support* OFF → ON
3. Rebuild and reinstall the modules by running the `/opt/edgeai-gst-apps/scripts/install_tiovx_modules.sh` script

9.1 Requirements

- AM6xA EVM must be connected to network via Ethernet.
- The PC and EVM must be connected to the same network. The PC must be able to ping the EVM.
 - A VPN connection can interfere with this. Please disconnect the PC from VPN before trying live tuning.
 - Find the EVM IP address using `ifconfig` on the EVM Linux console and enter it in the box as shown below.



Figure 9-1. EVM IP Address

- Run a GStreamer pipeline so that (1) sensor streams a raw image to AM6xA and (2) AE, AWB, and tuning server in GStreamer are also running.

9.2 Supported Features

Supported features are listed and explained in the following sections.

9.2.1 RAW Capture

RAW capture saves a raw image from the output of the capture node. This image is the output of image sensor and is completely unprocessed by the J7, AM6xA ISP. The width, height, and bit-depth are determined by the sensor driver.



Figure 9-2. RAW Capture

9.2.2 YUV Capture

YUV capture saves a YUV image from the output of VISS node. This image is the output of ISP, before LDC or scalar. The width, height, and bit-depth are determined by VISS node.



Figure 9-3. YUV Capture

9.2.3 Live DCC Update

Live *DCC Update* includes the following features:

- Sends the DCC binary of the plug-in currently open in the GUI tool to the EVM. Instant impact on the image quality is seen on the display connected to the EVM.
- Immediate effect of new parameters without the need for any code changes or restarting the application.



Figure 9-4. DCC Update

9.2.4 Exposure Control

- In the "Algorithm View" tab, read the exposure and gain computed by the AE algorithm
- Bypass AE by setting AE Mode = Manual (1)
 - Set manual exposure and gain values, subject to sensor driver limits (parameters are set [in this section](#)).

Name	Value	Note
▼ AE Parameters		
AE Mode	0	Auto (0) or Manual (1)
Digital Gain	1	Digital Gain
Analog Gain	15.85	Analog Gain
Exposure Time	10000000	Exposure time (us)
▼ AWB Parameters		
AWB Mode	0	Auto (0) or Manual (1)
R Gain	1.246	Red Color Gain
G Gain	1	Green Color Gain
B Gain	1.602	Blue Color Gain
Color Temperature	4087	Color Temperature (K)

Figure 9-5. Exposure Control for Live Tuning

9.2.5 White Balance Control

- Read the gain and color temperature computed by the AWB algorithm
- Bypass AWB by setting AWB Mode = Manual (1)
- Set manual gains and color temperature, subject to sensor and VISS driver limits

Name	Value	Note
▼ AE Parameters		
AE Mode	0	Auto (0) or Manual (1)
Digital Gain	1	Digital Gain
Analog Gain	15.85	Analog Gain
Exposure Time	10000000	Exposure time (us)
▼ AWB Parameters		
AWB Mode	0	Auto (0) or Manual (1)
R Gain	1.246	Red Color Gain
G Gain	1	Green Color Gain
B Gain	1.602	Blue Color Gain
Color Temperature	4087	Color Temperature (K)

Figure 9-6. White Balance Control for Live Tuning

9.2.6 Sensor Register Read/Write

The sensor register Read/Write function can read from or write to image sensor registers.

Note

This feature is only supported on IMX219.

10 Summary

This application note described the work flow of the AM6xA ISP tuning, using TI's AM62A Processor SDK, imaging software, and the DCC tuning tool. The RGB-only tuning procedure is described using an IMX219 camera, and RGB-IR specific tuning is described using an OX05B1S camera.

11 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (March 2023) to Revision A (May 2024)	Page
• Added exposure setting for 2A algorithm.....	7
• Added tuning guide for RGB-IR sensors	15
• Added tuning guides for Edge Enhancement	25
• Added tuning guides for Noise Filter	26
• Added guide for live tuning feature.....	28

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](https://www.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 © 2024, Texas Instruments Incorporated