eYs3D 鈺立微電子 an Etron company

Linux SDK EX8040S

Application Note

Revision 0.0.3
2019/12/04

Legal

Revision History

Rev	Date	Comments
0.0.1	2019/11/12	• Draft release.
0.0.2	2019/11/18	• Fixed some typo.
0.03	2019/12/4	• Added Module Sync
		•

Table of Contents

1. Introduction

This document presents the Linux SDK main functions, functional flow chart and so on, the application programmer will know how to boot up the Etron Depth Map module (Device Initial), preview the color and depth images, Device Control and generate the point cloud file.

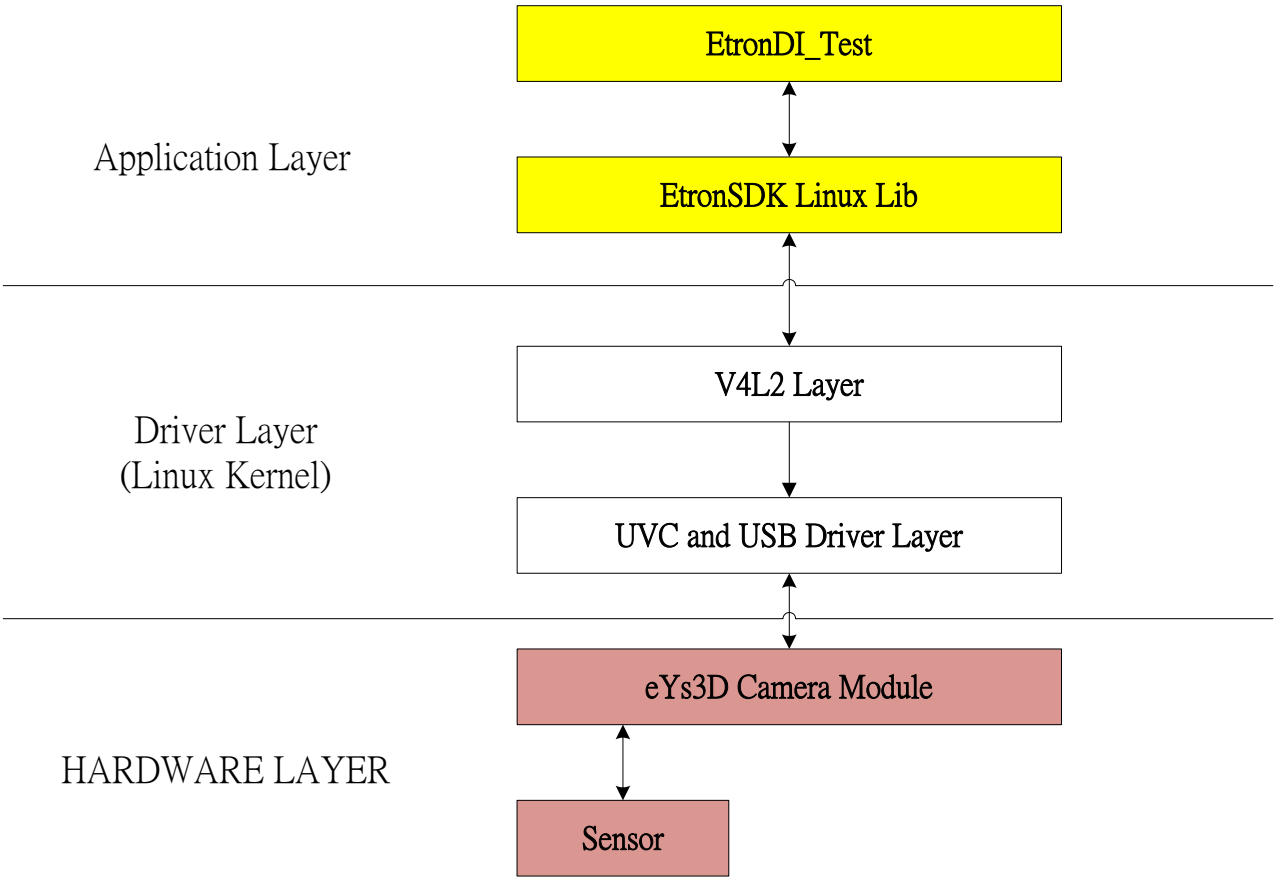
The EtronSDK_Linux is included below items.

EtronSDK_Linux/EtronDI_Test: All functional application demo code

EtronSDK_Linux/ eSPDI: EtronSDK Linux library.

EtronSDK_Linux\doc: API Spec.

Linux SW Architecture



2. Device Initial

Etron Depth Map Module plug in the Linux Platform (Eg, x86 PC, TX2, …), the /dev/videoX will be created on V4L2, the EtronSDK_Linux provide the application interface to access the /dev/videoX following UVC protocol.

This chapter descriptor how to initial the device.

2.1 Device Introduction

There are two video in EX8040S, one is C+D stream, other is K stream. The stream format table is below.

Video		Interface	USB1-EP1 Color + Depth-		USB2 - EP1 Color (2D or 3D)		Comment	Phase
L': Rectified Left, D: Depth, R: Right			Video	FPS	Video	FPS		
Mode 1	L+R (Calibration mode)	1x USB3.1 Gen1	1856x1056x2	24	N/A	N/A	for Calibration	1st
Mode 2	L+K (Calibration mode)	1x USB3.1 Gen1	1856x1056	2	3712x2112	2	for Calibration	1st
Mode 3	L'+R' (Calibration mode)	1x USB3.1 Gen1	1088X1920x2	24	N/A	N/A	for Calibration	1st
Mode 4	L'+D	1x USB3.1 Gen1	1080x1920x2	24	N/A	N/A	D11(default), Z14	1st
Mode 5	D+K (8M: K with MJPEG)	1x USB3.1 Gen1	912x1920	30	3840x1824	10	D11(default), Z14	1st
Mode 6	D+K (8M: K with MJPEG) (Multi-Module)	1x USB3.1 Gen1	912x1920	15	3840x1824	5	D11(default), Z14	1st
Mode 7	D+K' (4M: K' with MJPEG)	1x USB3.1 Gen1	912x1920	30	2560x1216	10	D11(default), Z14	2nd
Mode 8	D+K' (2M: K' with MJPEG) (Offload SW PP)	1x USB3.1 Gen1	456x960	30	1920x912	30	D11(default), Z14	2nd
Mode 9	D+K' (4M: K' with MJPEG) (Multi-Module)	1x USB3.1 Gen1	912x1920	15	2560x1216	5	D11(default), Z14	2nd
Mode 10	L+R (Calibration)	1x USB3.1 Gen1	1920x912x2	30	N/A	N/A	for Calibration	2nd
Mode 11	L+R+K (Calibration)	1x USB3.1 Gen1	1920x912x2	5	3840x1824	5	for Calibration	2nd
Mode 12	L'+D (Calibration)	1x USB3.1 Gen1	912x1920x2	30	N/A	N/A	for Calibration	2nd
Mode 13	L+R+K (Calibration)	1x USB3.1 Gen1	1920x912x2	5	2560x1216	5	for Calibration	2nd

2.2 Initial

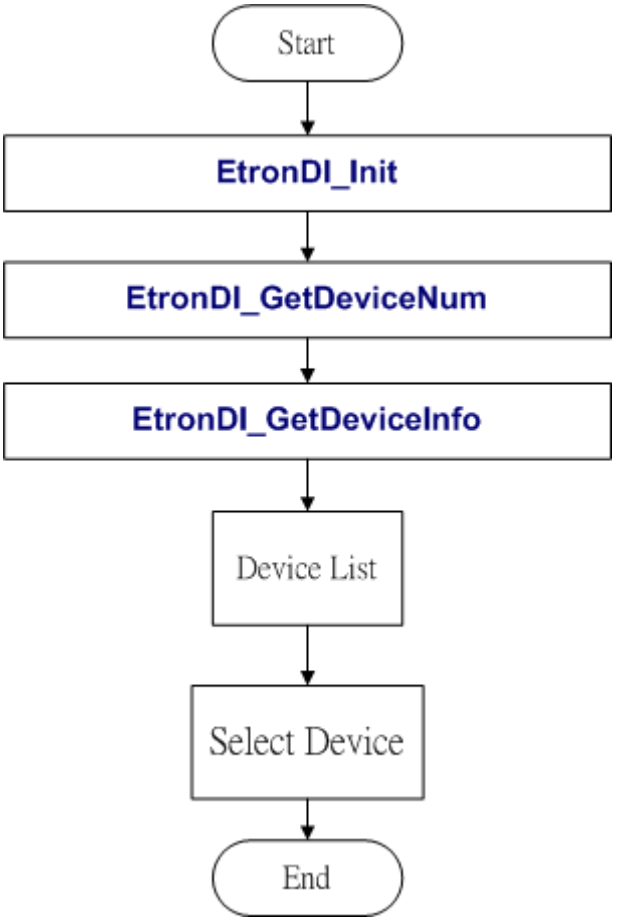
First create device handler through by EtronDI_Init API, get the Etron Device number from EtronDI_GetDeviceNumber, get the device information (EtronDI_GetDeviceInfo) to know device content and then select the devices, as below picture, 0 for /dev/video0 and 1 for /dev/video1

There are two devices in EX8040, B1335 for K stream of /dev/video0 and D777 for C+D stream of /dev/video1. We can create two Device Info classes to store device index. The m_pDevInfoEx.index = 0 for K stream, m_pDevInfo.index = 1 for C+D stream.

```
CVideoDevice: /dev/video0
CVideoDevice: /dev/video1
CVideoDevice: /dev/video2
nRet = 0
nDevCount = 2
Device Name = /dev/video0
PID = 0x0149
VID = 0x1e4e
Chip ID = 0x29
FW Version = EX8040S-C01-B1335-BL00U-004-BETA05

Device Name = /dev/video1
PID = 0x0131
VID = 0x1e4e
Chip ID = 0x15
FW Version = EX8040S-B01-D777-BL00U-004-BETA06
```

2.3 Flow chart



2.4 EtronDI_Initial

This function create device handler, the device /dev/videX will be opened, and the device information recorded in m_pEtronDI (device handler). The device information is included PID, VID, ChipID and so on, the m_pEtronDI (device handler) is a requirement of almost EtronSDK_Linux API.

2.5 EtronDI_GetDeviceNumber

We can use this function to get the number of Etron Depth Map device.

2.6 EtronDI_GetDeviceInfo

This function will provide the device information such as PID, VID, Chip name and device name.

2.7 Reference File

- EtronDI_Test/Mainwindows.cpp
- EtronDI_Test/Mainwindows.h
- eSPDI/eSPDI.h

3. Device Control

As videodevicedlg.cpp example code and according to chap 2.2, we can select m_pDevInfo.index = 1 for color + depth device, the m_pDevInfoEx.index = 0 for K device.

3.1 color + depth device

If we select m_pDevInfo.index = 1, the Color/Depth device will control below functions.

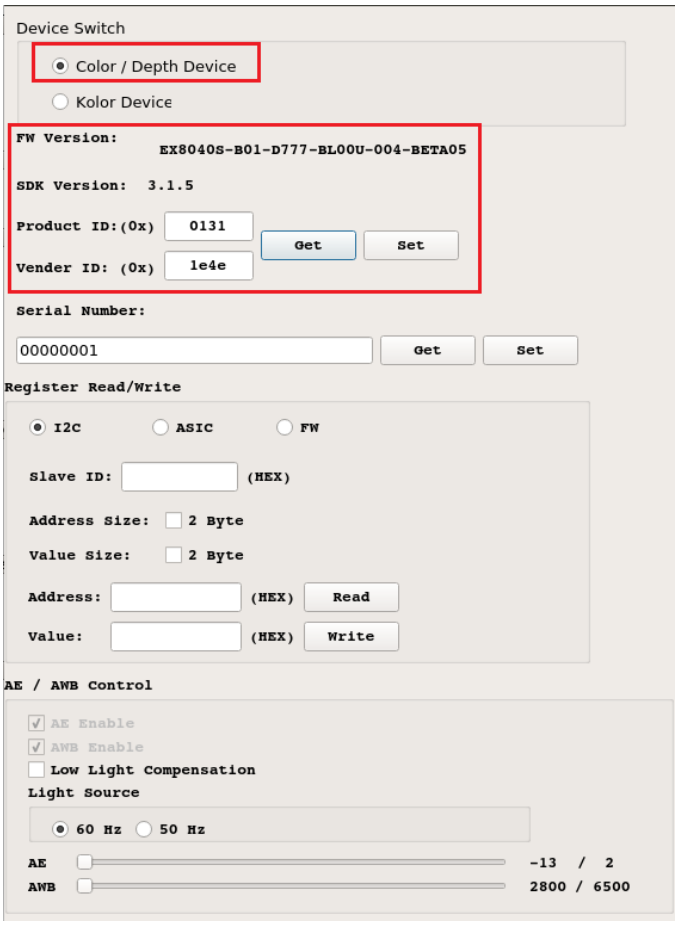
(1) Get Firmware Version

(2) Get PID / VID

(3) Get Serial Number

(4) Register Read / Write

(5) AE / AWB Control



3.2 Kolor device

If we select m_pDevInfoEx.index = 0, the Kolor device will control below functions.

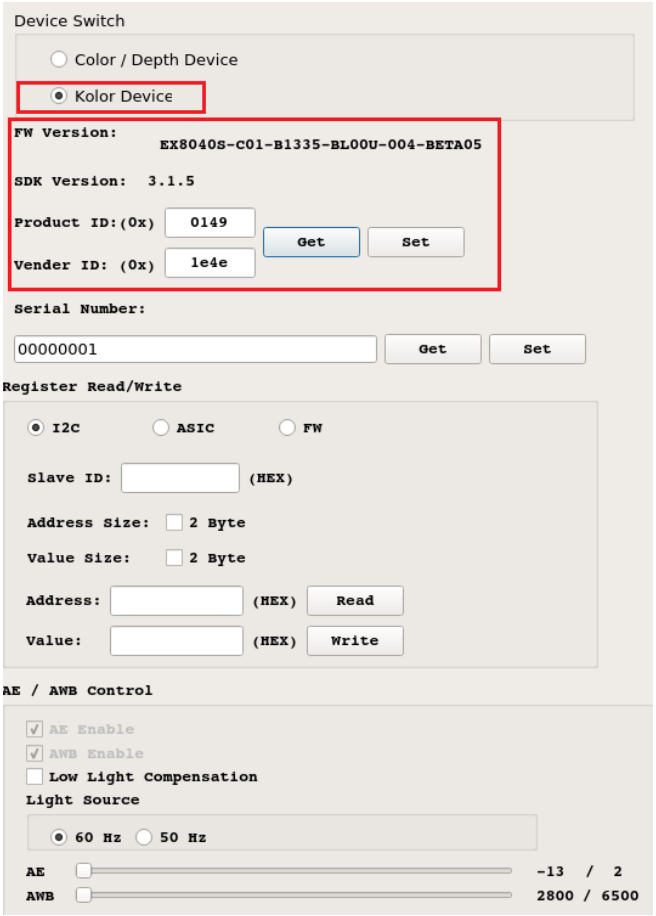
(6) Get Firmware Version

(7) Get PID / VID

(8) Get Serial Number

(9) Register Read / Write

(10)AE / AWB Control



3.2.1 Reference function

3.2.1.1 EtronDI_GetFwVersion

Get the firmware version of device.

3.2.1.2 EtronDI_GetPidVid

Get PID(product ID) and VID(vendor ID) of device

3.2.1.3 EtronDI_SetPidVid

Set PID(product ID) and VID(vendor ID) of device

3.2.1.3 EtronDI_GetSerialNumber

Get Serial Number of device

3.2.1.4 EtronDI_SetSerialNumber

Get Serial Number of device

3.2.1.5 EtronDI_GetCTPropVal

Get camera terminal(CT) property value by v4l2_control to get control value of camera terminal.

3.2.1.6 EtronDI_SetCTPropVal

Setup camera terminal(CT) property value by v4l2_control to get control value of camera terminal.

3.2.1.7 EtronDI_GetPUPropVal

Get processing unit property value by v4l2_control to get processing unit(PU) property value.

3.2.1.8 EtronDI_SetPUPropVal

Set processing unit property value by v4l2_control to get processing unit(PU) property value.

3.2.1.9 Parameters

CT_PROPERTY_ID_AUTO_EXPOSURE_MODE_CTRL

CT_PROPERTY_ID_AUTO_EXPOSURE_PRIORITY_CTRL

CT_PROPERTY_ID_EXPOSURE_TIME_ABSOLUTE_CTRL

CT_PROPERTY_ID_EXPOSURE_TIME_RELATIVE_CTRL

CT_PROPERTY_ID_FOCUS_ABSOLUTE_CTRL

CT_PROPERTY_ID_FOCUS_RELATIVE_CTRL

CT_PROPERTY_ID_FOCUS_AUTO_CTRL

CT_PROPERTY_ID_IRIS_ABSOLUTE_CTRL

CT_PROPERTY_ID_IRIS_RELATIVE_CTRL

CT_PROPERTY_ID_ZOOM_ABSOLUTE_CTRL

CT_PROPERTY_ID_ZOOM_RELATIVE_CTRL

CT_PROPERTY_ID_PAN_ABSOLUTE_CTRL

CT_PROPERTY_ID_PAN_RELATIVE_CTRL

CT_PROPERTY_ID_TILT_ABSOLUTE_CTRL

CT_PROPERTY_ID_TILT_RELATIVE_CTRL

CT_PROPERTY_ID_PRIVACY_CTRL

PU_PROPERTY_ID_BACKLIGHT_COMPENSATION_CTRL

PU_PROPERTY_ID_BRIGHTNESS_CTRL

PU_PROPERTY_ID_CONTRAST_CTRL

PU_PROPERTY_ID_GAIN_CTRL

PU_PROPERTY_ID_POWER_LINE_FREQUENCY_CTRL

PU_PROPERTY_ID_HUE_CTRL

PU_PROPERTY_ID_HUE_AUTO_CTRL

PU_PROPERTY_ID_SATURATION_CTRL

PU_PROPERTY_ID_SHARPNESS_CTRL

PU_PROPERTY_ID_GAMMA_CTRL

PU_PROPERTY_ID_WHITE_BALANCE_CTRL

PU_PROPERTY_ID_WHITE_BALANCE_AUTO_CTRL

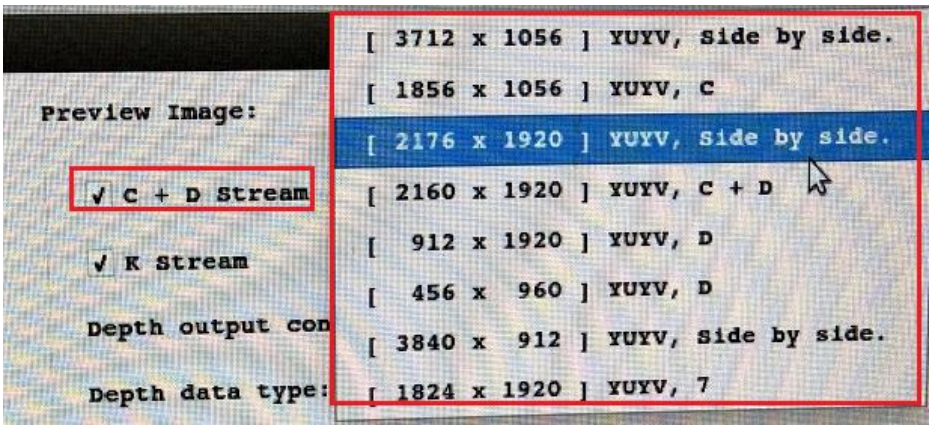
3.3 Reference File

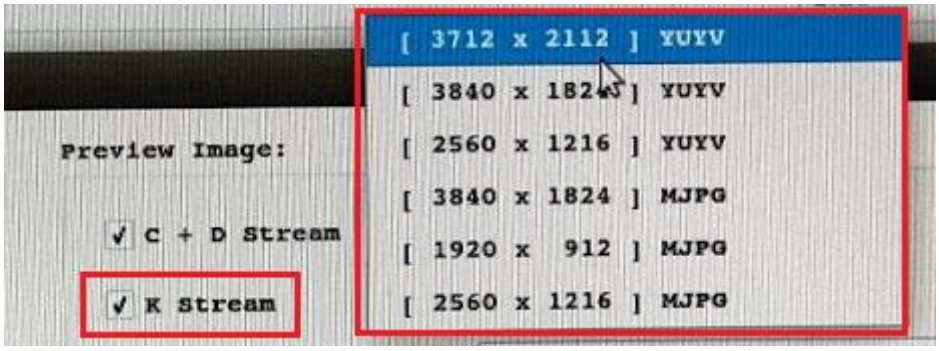
- EtronDI_Test/videodevicedlg.cpp
- EtronDI_Test/ videodevicedlg.h
- eSPDI/eSPDI.h

4 Preview

The Etron Depth module provided the depth and color video stream. This chapter descriptor previewed the color video and depth video through by EtronSDK_Linux.

Fisrt we need initial the device (please refer the Device Initial chapter), setup the Depth Data Type, and doing EtronDI_GetDeviceResolutionList to get the all resolution from C+D Device and Kolor device, and then we will know which resolution and MJPEG supported.



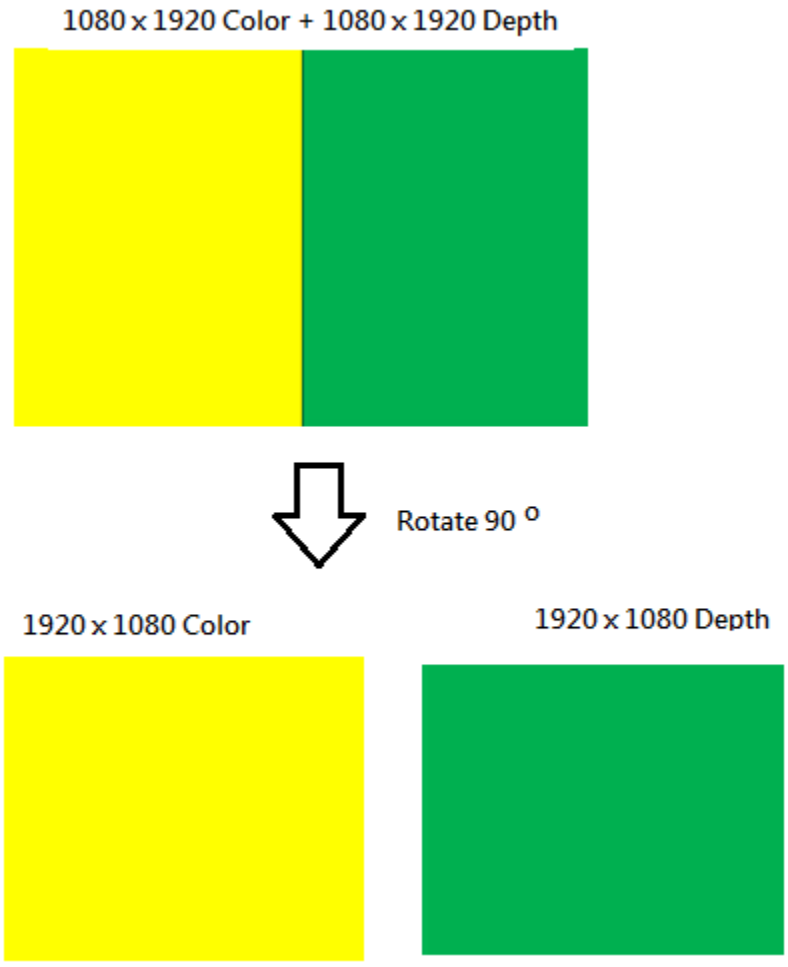


And do EtronDI_OpenDevice2 to open C+D and Kolor device.

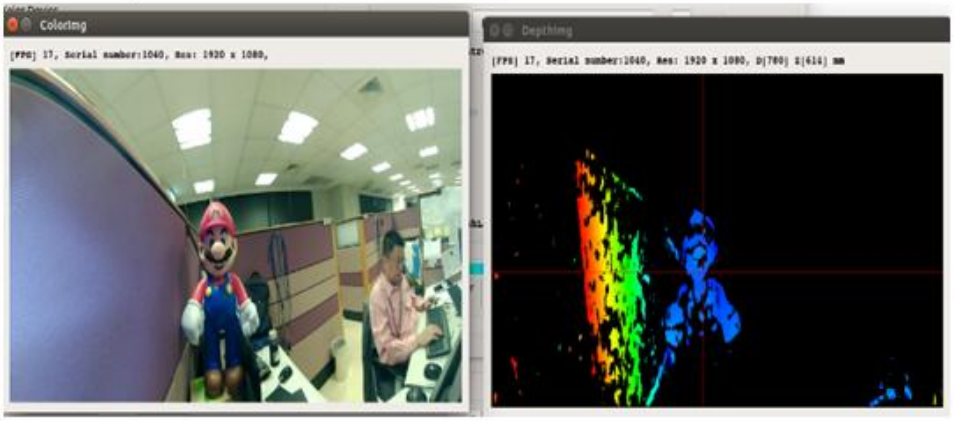
We need create the two dialog on for color video image and one for depth image. The C+D image is from the EtronDI_GetColorImage and the Kolor image is from the EtronDI_GetColorImage.

The C + D image should be rotated 90, K image do not need rotate 90.

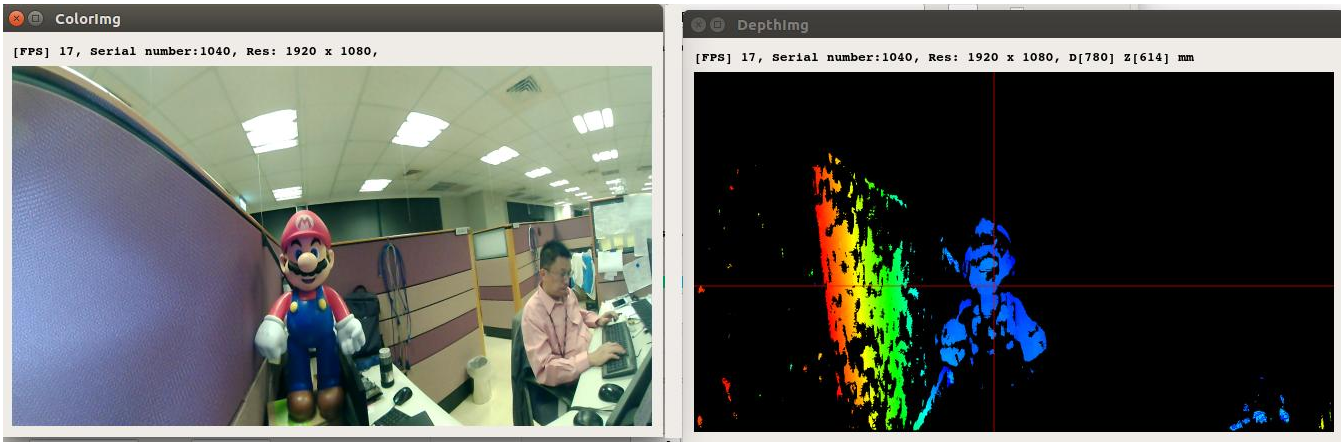
For example 2160 x 1920 YUYV C+ D mode, we need separate the Color Image and Depth Image, and then rotate both color and Depth image, the resolution are 1920 x 1080.



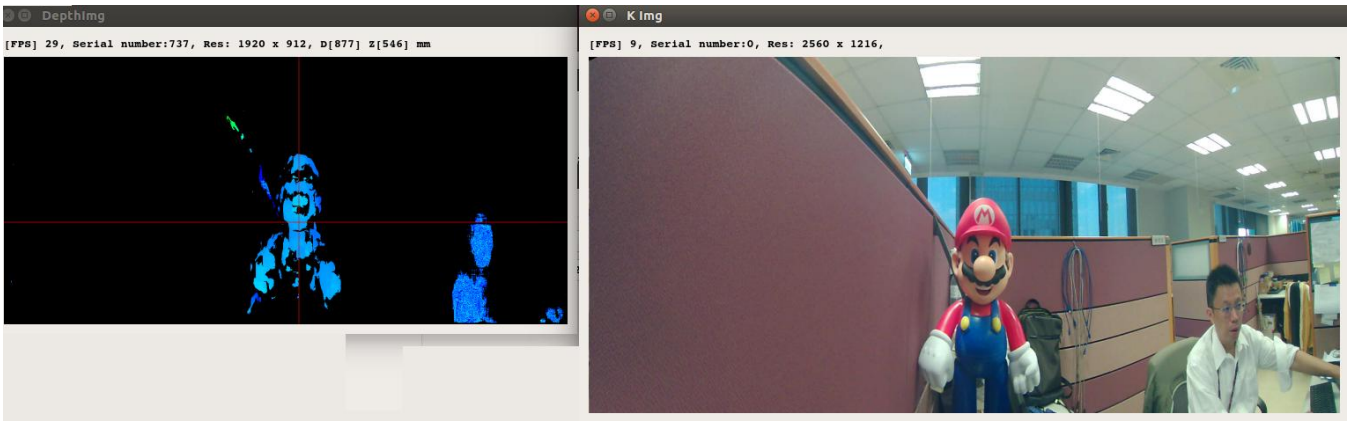
Rotate 90 °
and YUYV2RGB



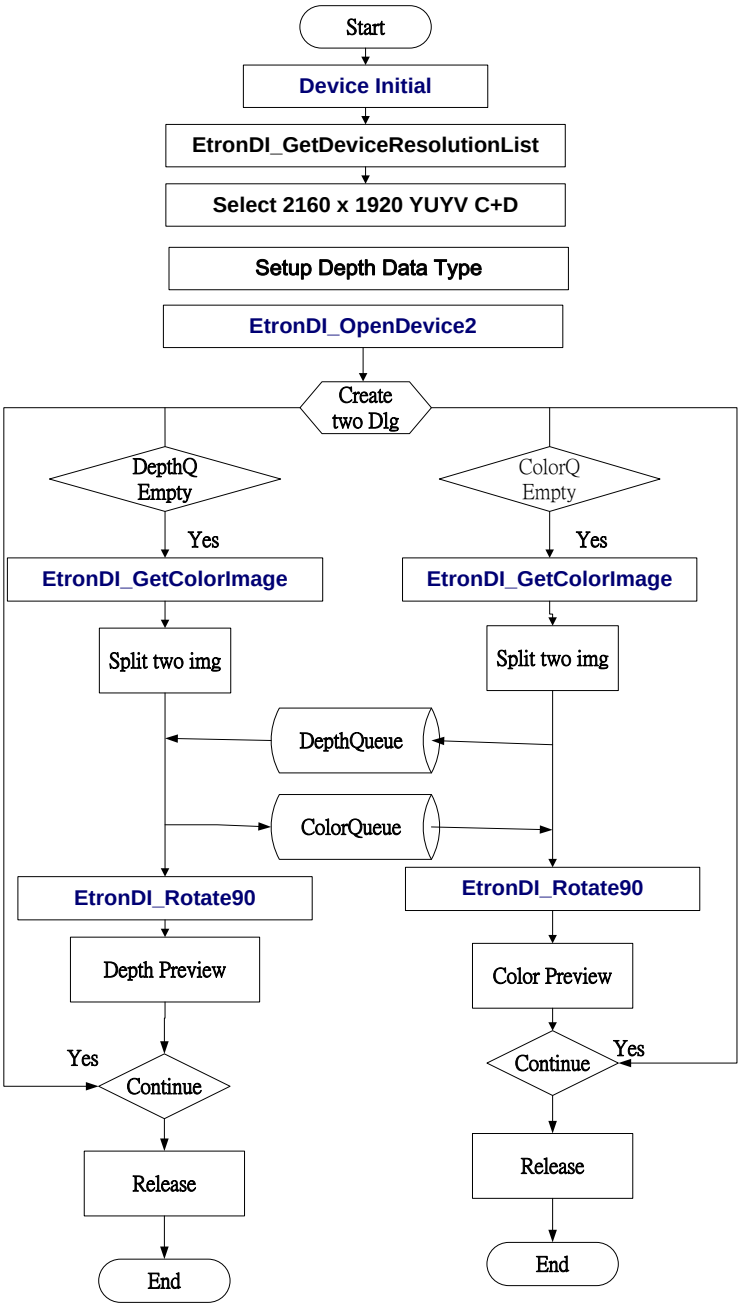
2160 x 1080 YUYV C+D



1920 x 912 YUYV Depth and 2560 x 1216 Kolor MJPEG



4.2 Flow Chart



4.3 EtronDI_GetDeviceResolutionList

Get the device resolution List

4.4 EtronDI_OpenDevice2

This function provided opening both depth and color device.

The implement layer to open Etron camera device by V4L2(https://en.wikipedia.org/wiki/Video4Linux),

It open color and depth at one time call, do functions as below,

1. Initialize the USB device by V4L2 protocol
   - Query device v4l2 capability, e.g. video capability, streaming capability
   - Setup the resolution mode to UVC driver and check result
   - Initialize memory buffer mapping from kernel to user mode
2. Enumerate frame interval to set frame rate
3. Start video capture processes

4.5 EtronDI_GetColorImage

Get Color image only by this function.

4.6 EtronDI_Rotate90

Rotate the image 90 degree.

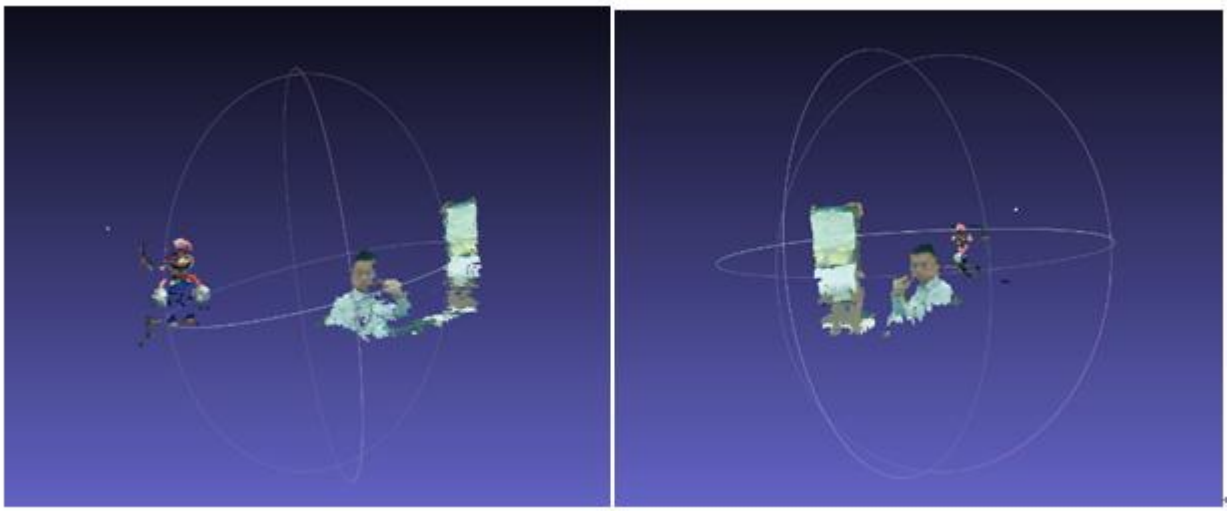
4.7 Reference File

- EtronDI_Test/Mainwindows.cpp
- EtronDI_Test/Mainwindows.h
- eSPDI/eSPDI.h
- EtronDI_Test/videodevicedlg.cpp
- EtronDI_Test/videodevicedlg.h
- EtronDI_Test/colordlg.cpp
- EtronDI_Test/colordlg.h
- EtronDI_Test/depthdlg.cpp
- EtronDI_Test/depthdlg.h

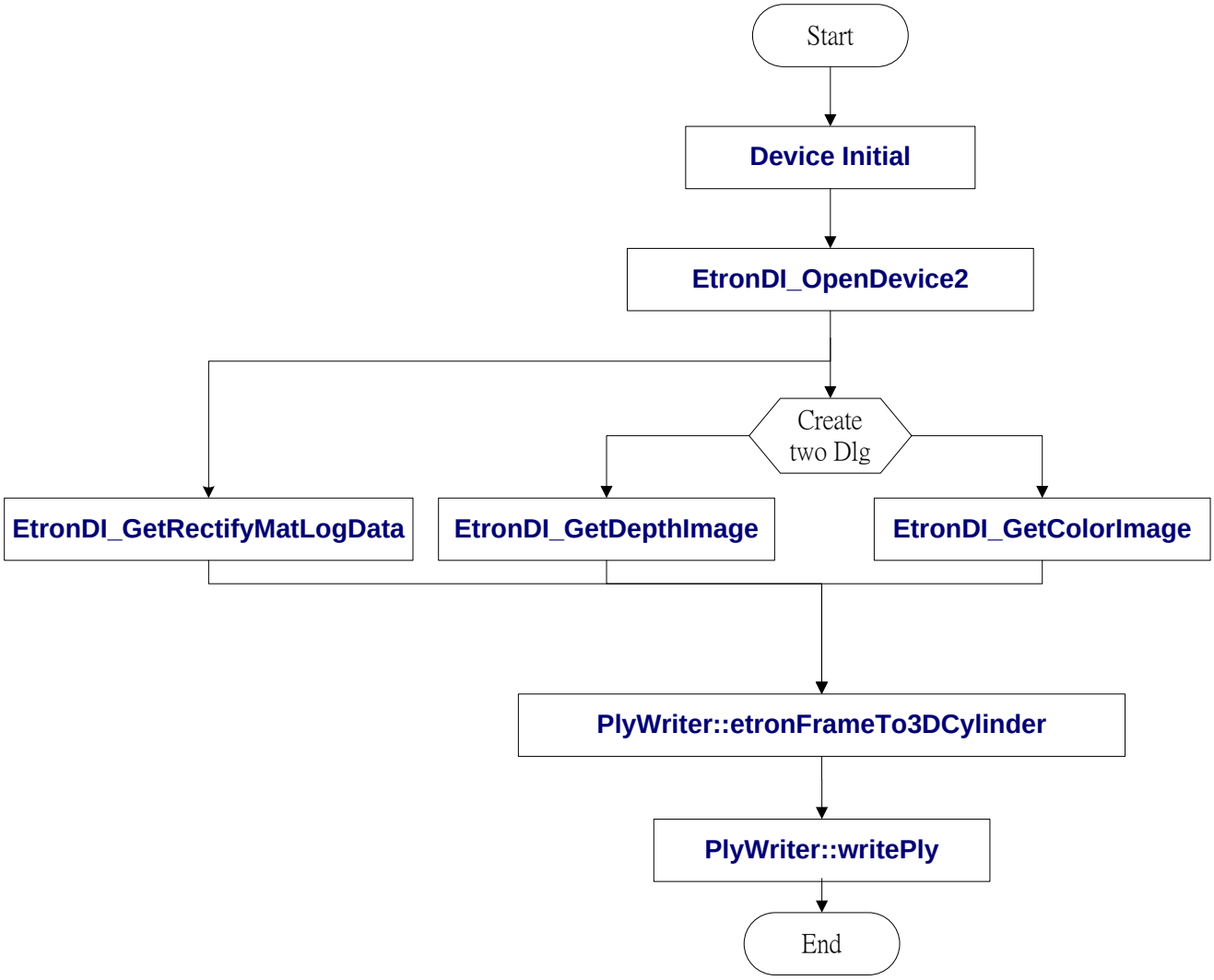
5 Point Cloud

EtronSDK_Linux support 3D image point clued, we can generate the ply (**Polygon File Format**) image, we can read ply file through by MeshLab tools as below pictures.

The ply file need Recitify data, Depth Map data and ColorImage, they are from EtronDI_GetRectifyMatLogData, EtronDI_GetDepthImage and EtronDI_GetColorImage, put Recitify data, Depth Map data and ColorImage to PlyWriter::etronFrameTo3D to get 3D image, and put the 3D image to PlyWriter::writePly to get ply image.



4.1 Flow Chart



4.2 EtronDI_GetRectifyMatLogData

This function can get the Rectify Math Data from Etron Depth Map module.

4.3 PlyWriter::etronFrameTo3DCylinder

Generated the 3D image from etronFrameTo3DCylinder function, we shell input depth map data, color RGB image and Rectify Math Data.

4.4 PlyWriter::writePly

Input the 3D image in this function to generate ply file.

4.5 Reference File

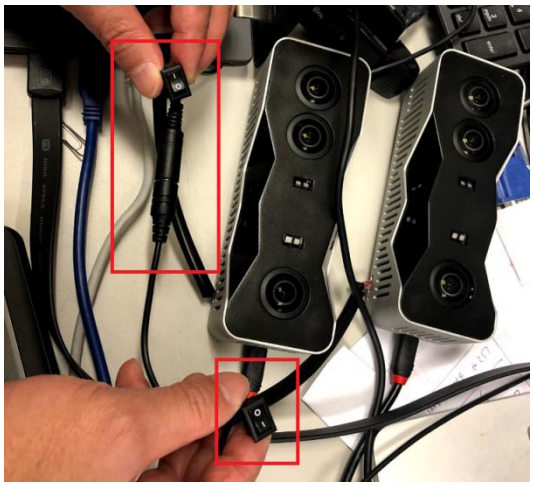
- EtronDI_Test/Mainwindows.cpp
- EtronDI_Test/Mainwindows.h
- eSPDI/eSPDI.h
- EtronDI_Test/videodevicedlg.cpp
- EtronDI_Test/videodevicedlg.h
- EtronDI_Test/colordlg.cpp
- EtronDI_Test/colordlg.h
- EtronDI_Test/depthdlg.cpp
- EtronDI_Test/depthdlg.h
- EtronDI_Test/plywriter.cpp
- EtronDI_Test/ plywriter.h

6 Multiple Module Sync

Two module sync testing steps following,

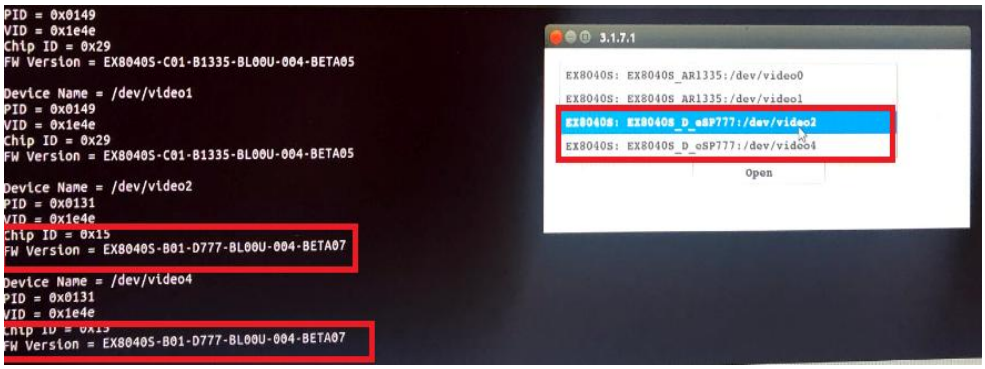
(1) Connect two module on one or two PC

(2) Sync Line config



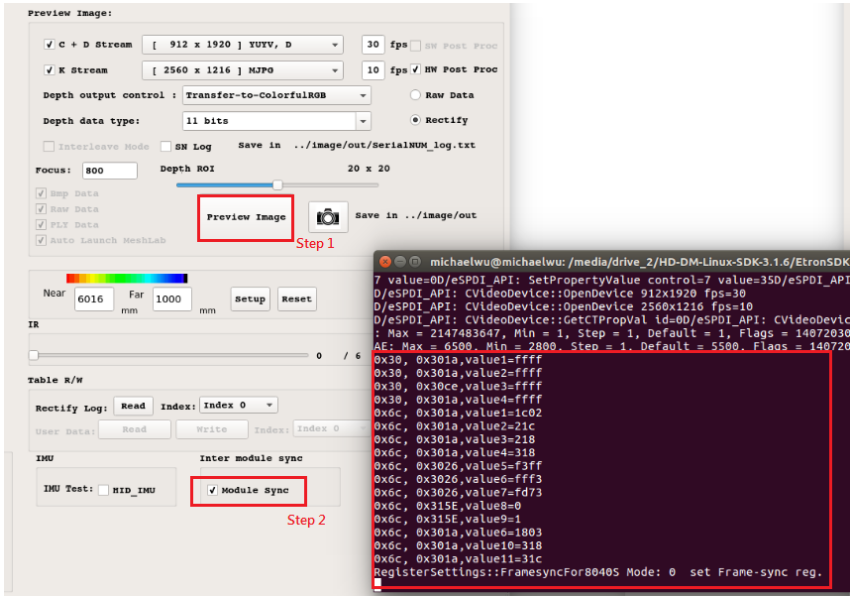
(3) Open device

If two module connect one PC, we need do run_qt/run_x86_EtronDI_Test_EX8040.sh two times, and open eSP777 device for each module.



(4) Do preview for each module

(5) Checked Module Sync Checked Box



(6) The Module Sync function please reference to RegisterSetting.cpp Fraamsync() function