eYs3D 鈺立微電子 an Etron company

Linux SDK

Application Note

Revision 0.0.3
2019/12/25

Legal

Revision History

Rev	Date	Comments
0.0.1	2018/10/15	• Draft release.
0.0.2	2019/12/23	• Remove Ex8038, • Added Multiple Module Sync
0.0.3	2019/12/25	• Modified Module Sync Flow Chart
0.0.4	2020/04/15	• Added 3D_Test

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 , post process, boot up the multiple baseline module (Eg, Ex 8038), preview the Fusion and Fusion select and generate the point cloud file.

The EtronSDK_Linux is included below items.

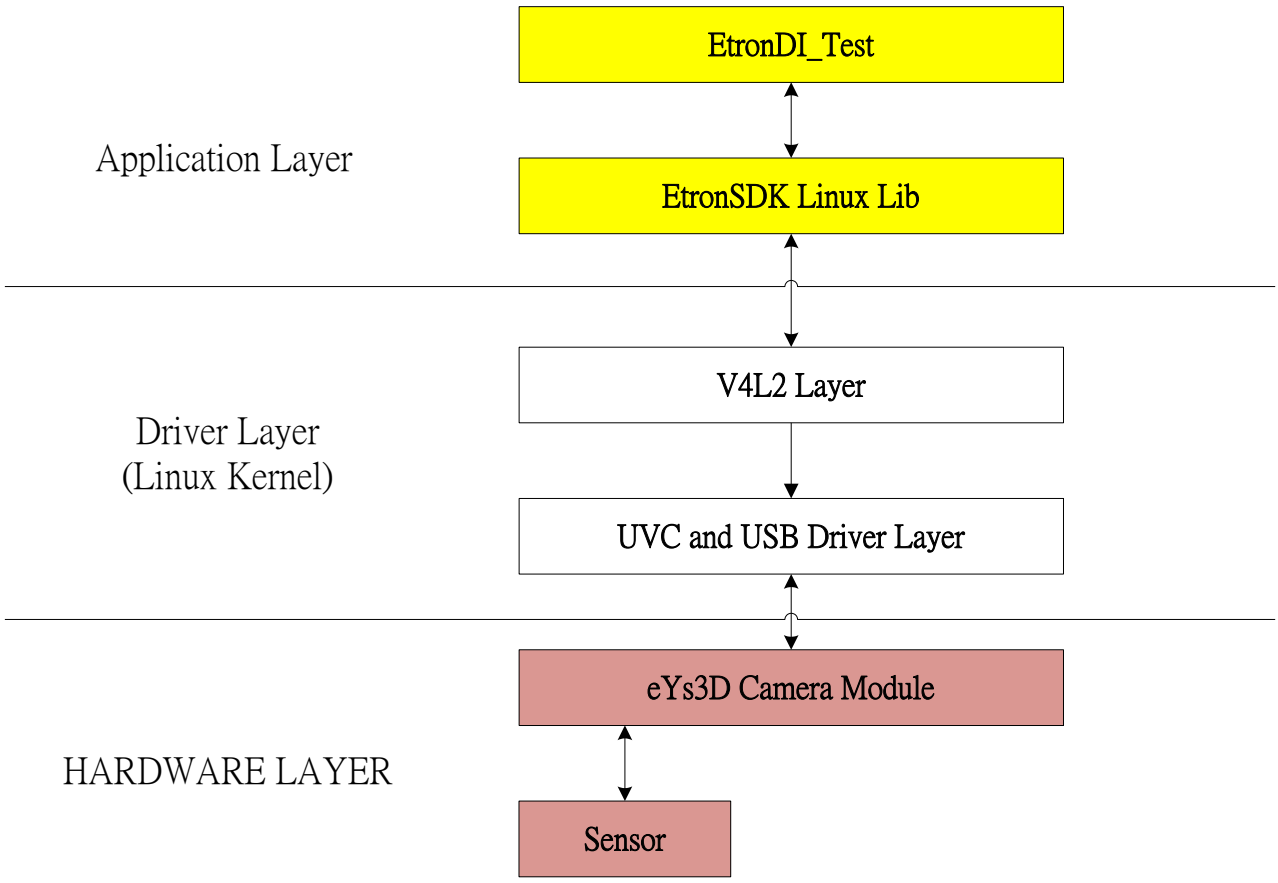
EtronSDK_Linux/EtronDI_Test: All functional application demo code

EtronSDK_Linux/ eSPDI: EtronSDK Linux library.

EtronSDK_Linux/ eSPDI: EtronSDK Linux library.

EtronSDK_Linux\doc: API Spec.

Linux SW Architecture



2. Device Initial

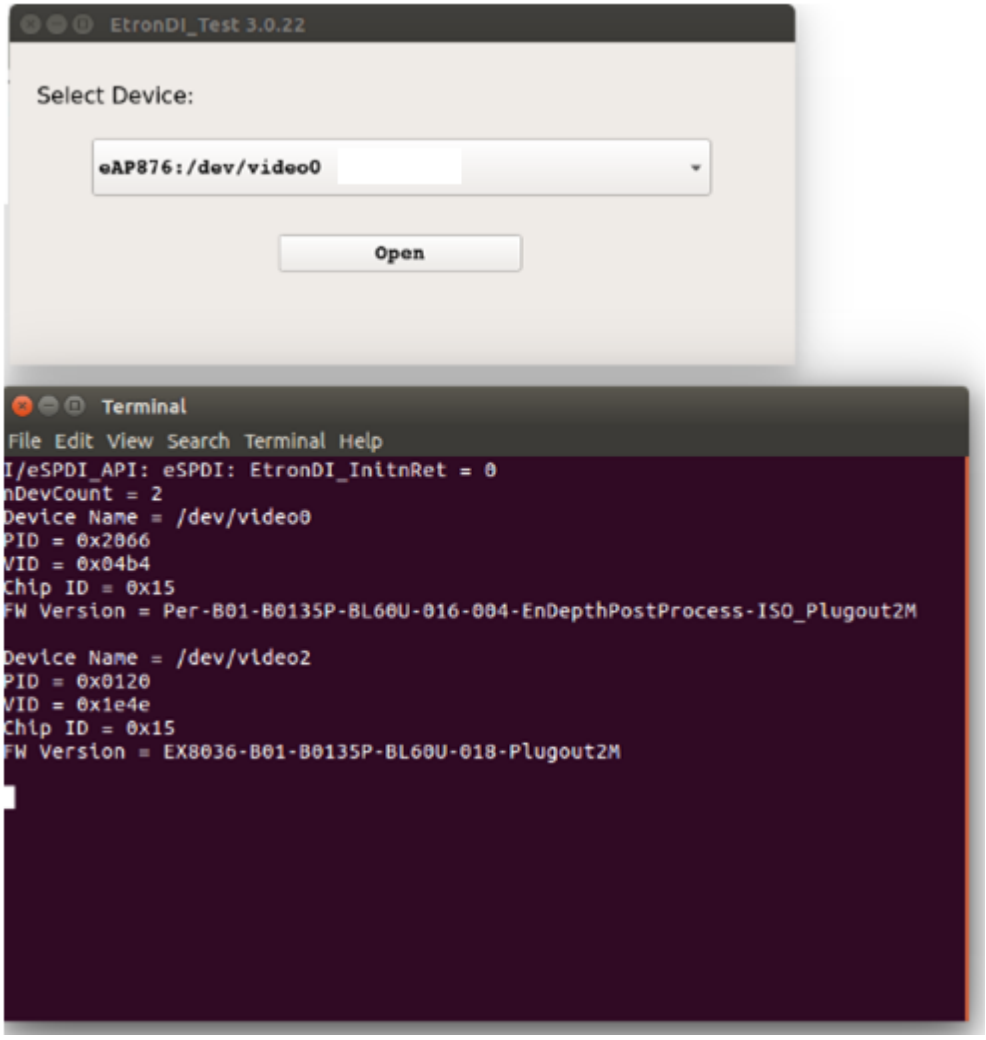
Etron Depth Map Module plug in the Linux Platform (Eg, x86 PC, TX2, …), the /dev/videoX will be created on V4L2, the EtronSDK_Linux provide the application interface to access the /dev/videoX following UVC protocol.

This chapter descriptor how to initial the device.

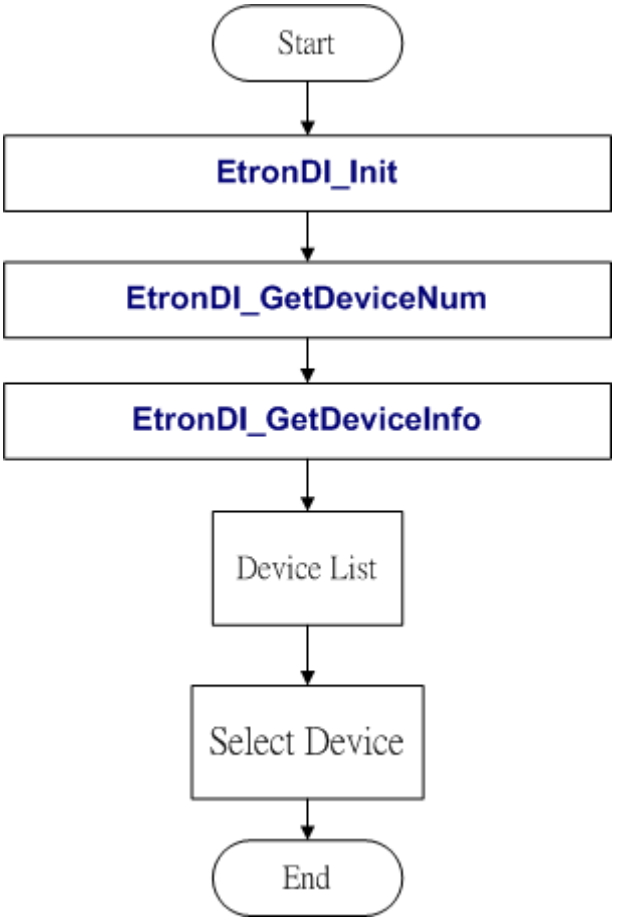
2.1 Initial

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 desired device.

If we click Open of the mainwidows dialog, the device /dev/video will be selected and opened, the device information will be recorded in m_pEtronDI (device handler). The device information is included PID, VID, ChipID and so on.



1.1 Flow chart



1.2 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.

1.3 EtronDI_GetDeviceNumber

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

1.4 EtronDI_GetDeviceInfo

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

1.5 Reference File

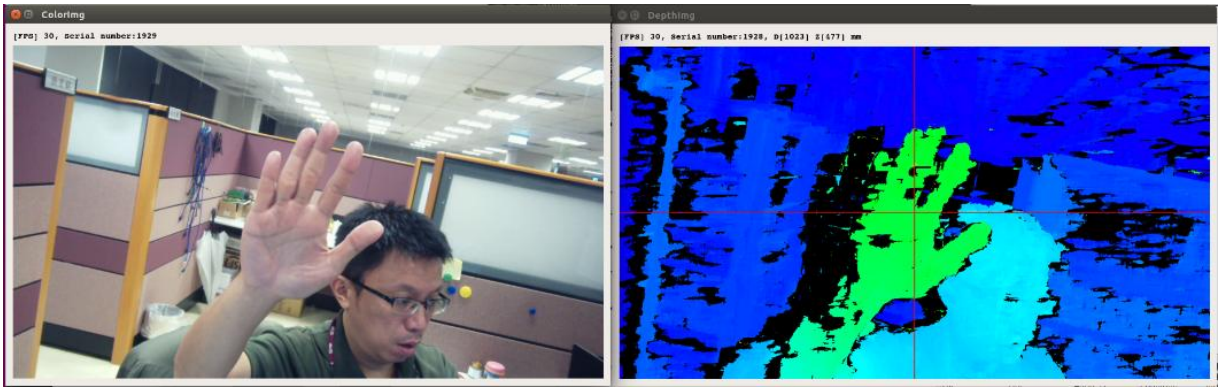
- EtronDI_Test/Mainwindows.cpp
- EtronDI_Test/Mainwindows.h
- eSPDI/eSPDI.h

2. Preview

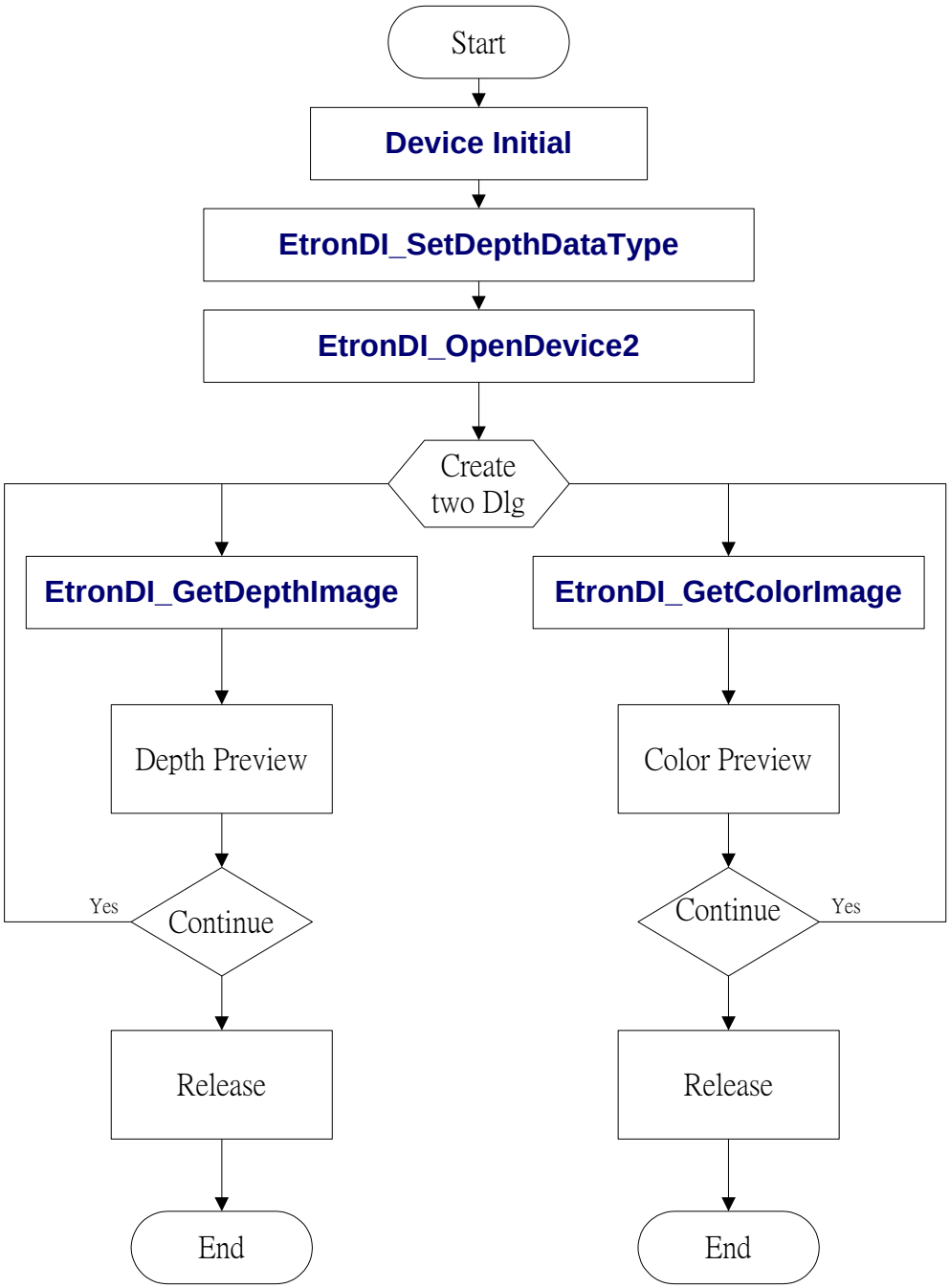
The Etron Depth module provided the depth and color video stream. This chapter descriptor previewed the color video and depth video through by EtronSDK_Linux.

Fisrt we need initial the device (please refer the Device Initial chapter), setup the Depth Data Type, and doing EtronDI_OpenDevice2 to open video and depth device.

We need create the two dialog on for color video image and one for depth image. The depth image is from the EtronDI_GetDepthImage and the color image is from the EtronDI_GetColorImage.



2.1 Flow Chart



2.2 EtronDI_SetDepthDataType

Brief set depth data type.

Data Type	Value	Description
ETronDI_DEPTH_DATA_OFF_RAW	0	raw (depth off, only raw color)
ETronDI_DEPTH_DATA_DEFAULT	0	raw (depth off, only raw color)
ETronDI_DEPTH_DATA_8_BITS	1	rectify, 1 byte per pixel
ETronDI_DEPTH_DATA_14_BITS	2	rectify, 2 byte per pixel
ETronDI_DEPTH_DATA_8_BITS_x80	3	rectify, 2 byte per pixel but using 1 byte only
ETronDI_DEPTH_DATA_11_BITS	4	rectify, 2 byte per pixel but using 11 bit only
ETronDI_DEPTH_DATA_OFF_RECTIFY	5	rectify (depth off, only rectify color)
ETronDI_DEPTH_DATA_8_BITS_RAW	6	Raw
ETronDI_DEPTH_DATA_14_BITS_RAW	7	Raw
ETronDI_DEPTH_DATA_8_BITS_x80_RAW	8	Raw
ETronDI_DEPTH_DATA_11_BITS_RAW	9	Raw
ETronDI_DEPTH_DATA_11_BITS_COMBINED_RECTIFY	13	multi-baseline

2.3 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

2.4 EtronDI_GetDepthImage

Get Depth image only by this function.

2.5 EtronDI_GetColorImage

Get Color image only by this function.

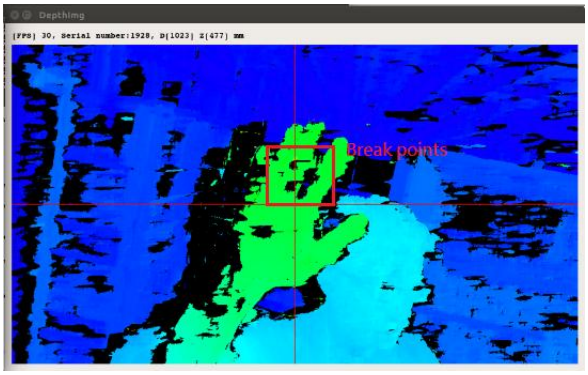
2.6 Reference File

- 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

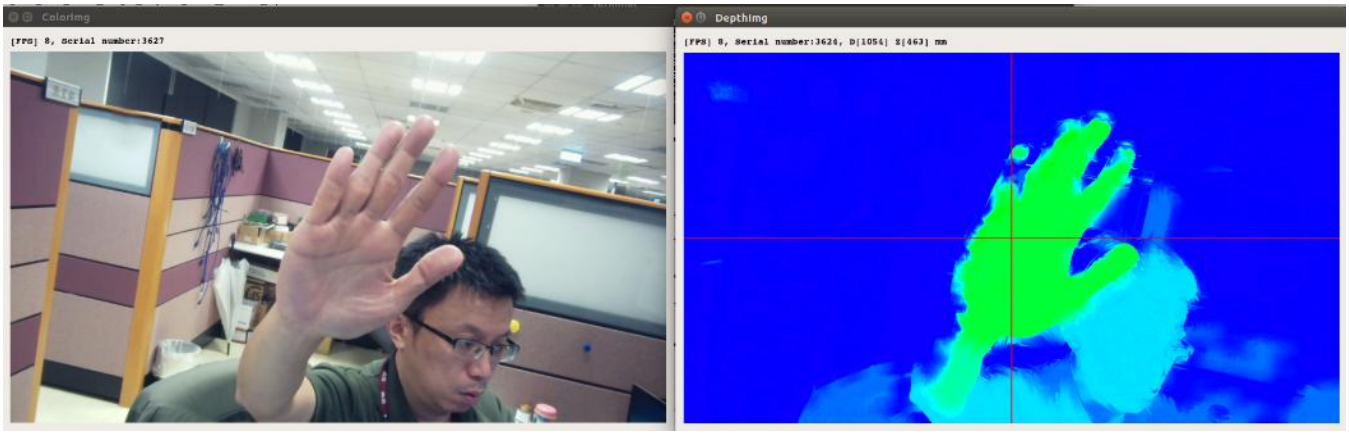
3 Post Process

Some holes are included in depth map, we can clean up the holes by post process. EtronSDK_Linux provided EtronDI_CreateSwPostProc and EtronDI_DoSwPostProc to generate the post prcess.

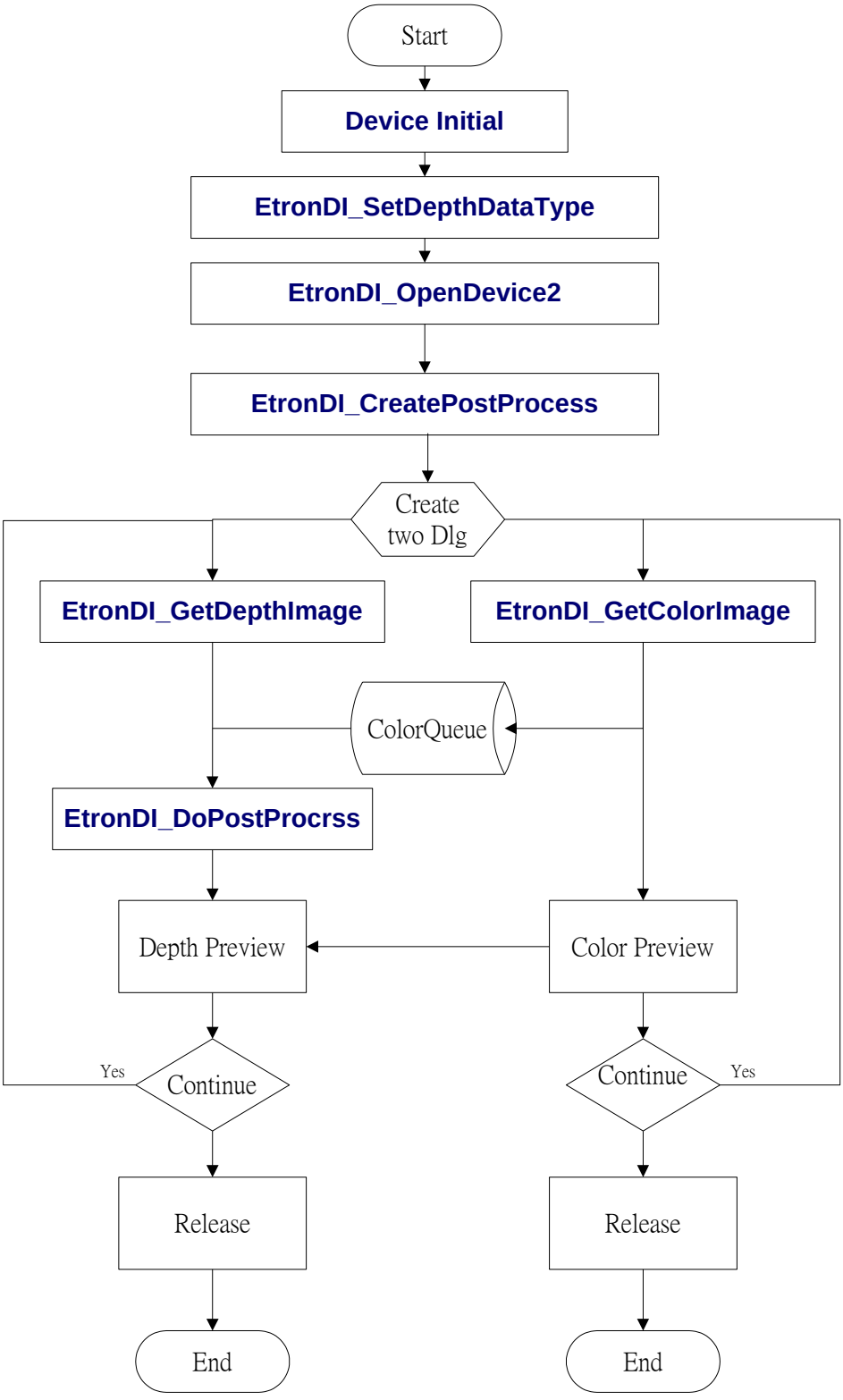
Before Post Process



After Post precess.



3.1 Flow chart



3.2 EtronDI_CreateSwPostProc

This function will initial the post process.

3.3 EtronDI_DoSwPostProc

If the Color Image and Depth image are ready, and then input them to this function, we will get the Post Process image.

3.4 Limitation

EtronSDK_Linux provide the Post Process APIs which based on openCL frame work, in the other word we need install the opencl on our platform (x86 or ARM) before doing post process.

+

3.5 Reference File

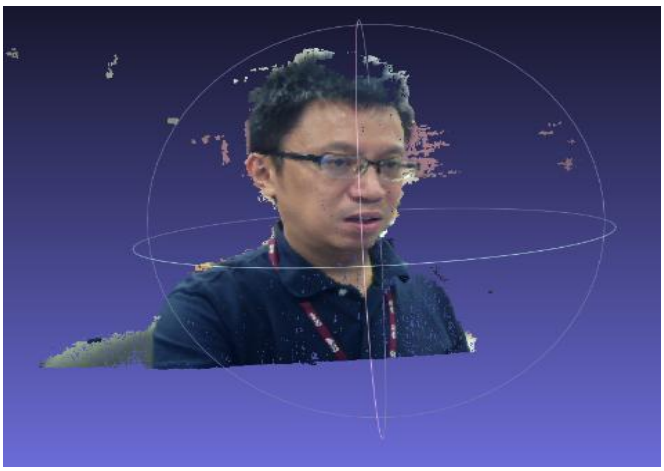
- EtronDI_Test/Mainwindows.cpp
- EtronDI_Test/Mainwindows.h
- eSPDI/eSPDI.h
- EtronDI_Test/videodevicedlg.cpp
- EtronDI_Test/videodevicedlg.h
- EtronDI_Test/colordlg.cpp
- EtronDI_Test/colordlg.h
- EtronDI_Test/depthdlg.cpp

EtronDI_Test/depthdlg.h

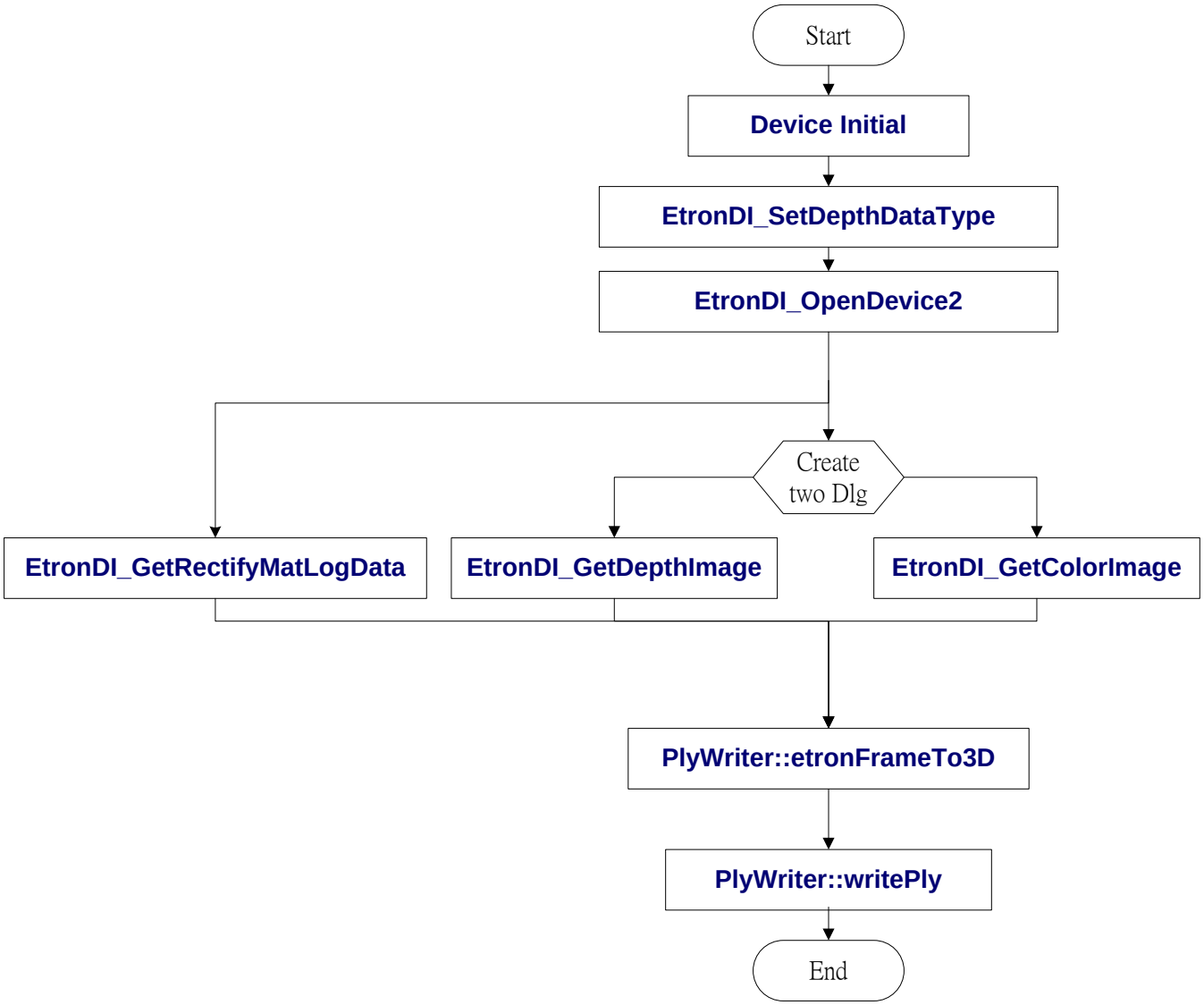
4 Point Cloud

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::etronFrameTo3D

Generated the 3D image from etronFrameTo3D 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

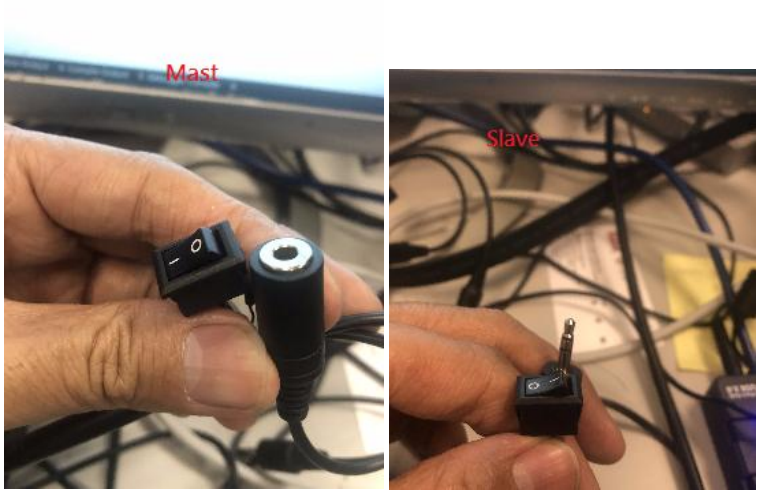
5. Multiple Module Sync

YX8053/YX8059 Multiple Module Sync supported.

Two module sync testing steps following,

(1) Connect two module on one or two PC

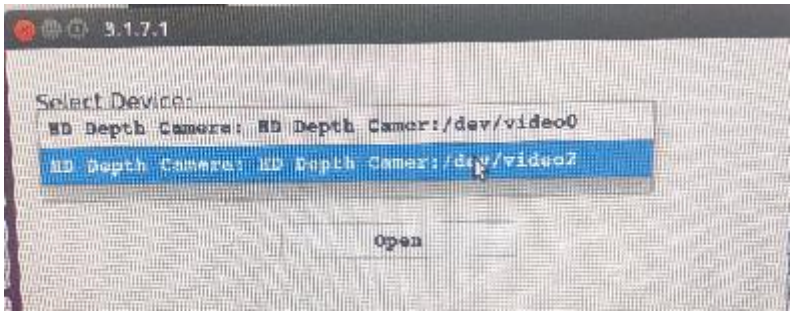
(2) Sync Line config



(3) Open device

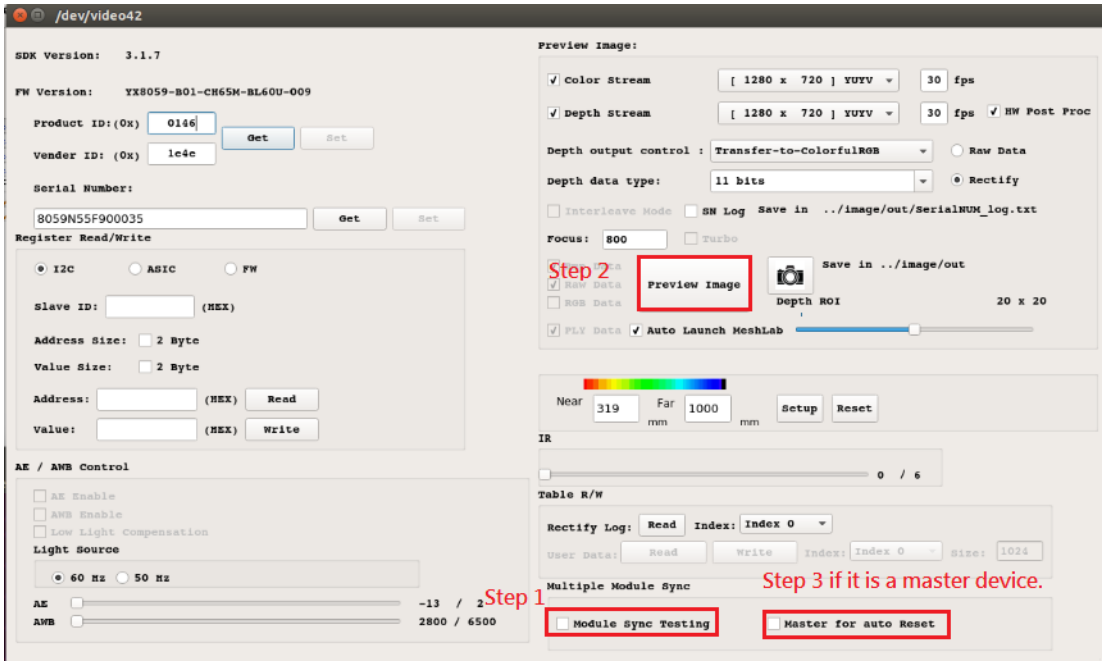
If two module connect one PC, we need do run_qt/run_x86_EtronDI_Test.sh two times,

Open /dev/video0 and /dev/video2



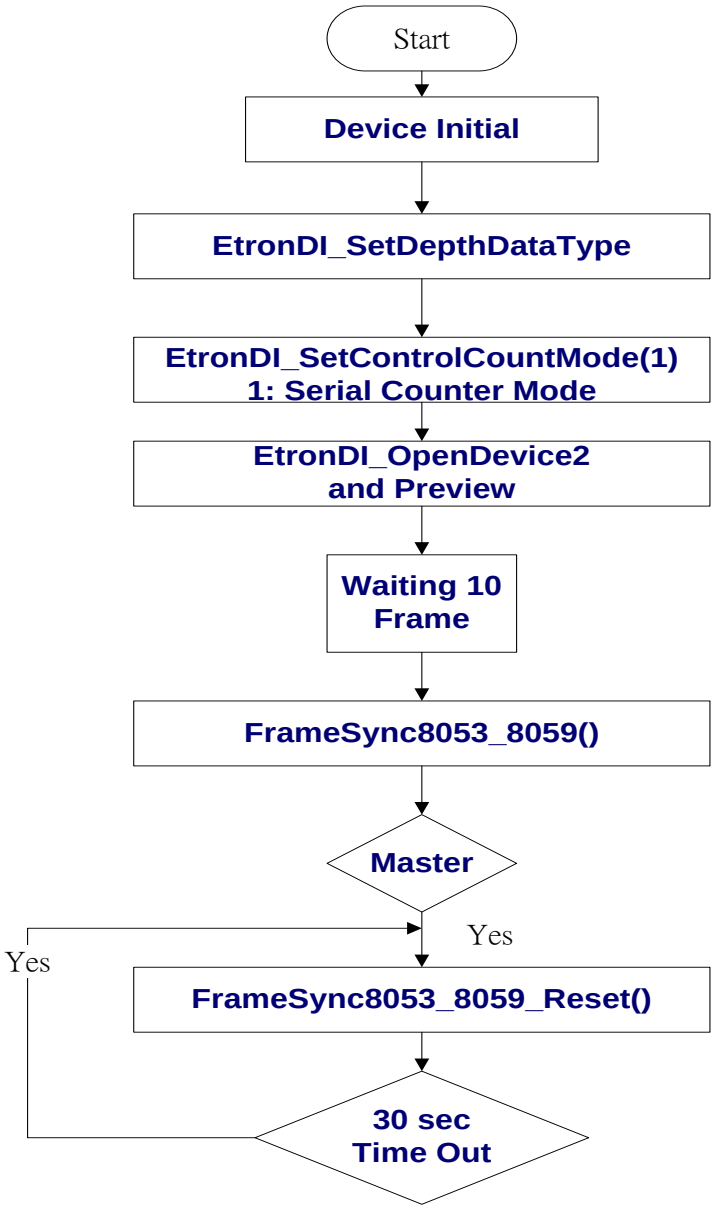
(4) Do preview for each module

(5) Checked Module Sync Testing Checked Box



(6) Checked Master for auto Reset Checked Box (Mater Module need be checked)

5.1 Flow chart



6. 3D Tester

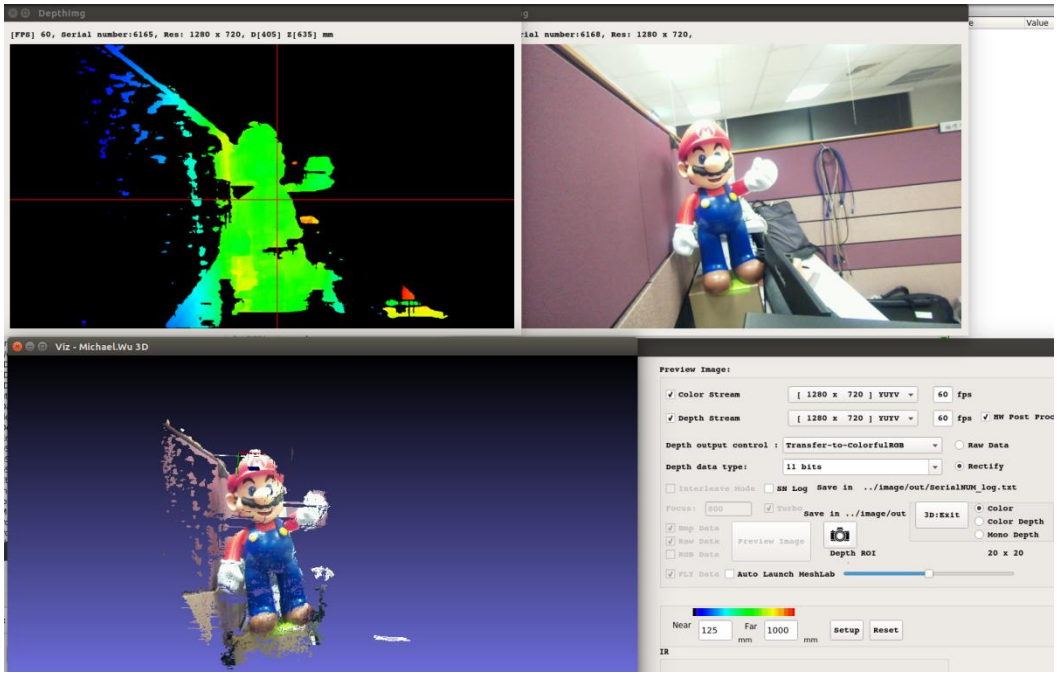
3D Tester is a dynamic point cloud tester, it is a demean tester.

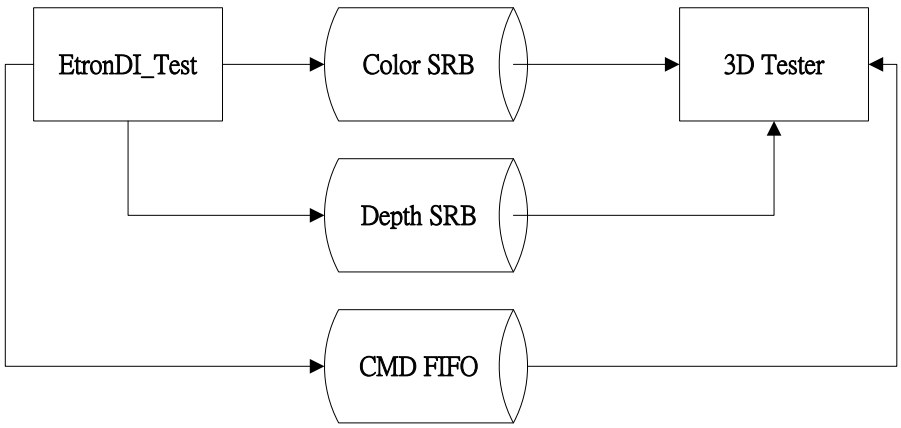
Which is in util folder. Ex util/3D_test

The EtronDI_Test get the color and depth Images from sensor device, put the images to Share Ring Buffer(SRB), the 3D Tester get these images from SRB.

EtronDI_Test can send the control cmd through by cmd FIFO,

3D Tester get the control cmd from cmd FIFO





6.1 Input Parameters

```
Set stream format & resolution (format = 0: YUYV, 1: MJPEG)
ex> 0 1280 720 4 1 0 30 165 1000 2 (PUMA)<-- YUYV  foramt, 1280 x 720 11 bits zdTableIndex color 30(FPS) near far 1
ex> 1 1280 720 4 1 1 30 165 1000 1 (AXES)<-- MJPEG foramt, 1280 x 720 11 bits  zdTableIndex mono depth  30(FPS) near far 0
ex> 1 1280 720 1 1 2 30 165 1000 2 (PUMA)<-- MJPEG foramt, 1280 x 720 8 bits  zdTableIndex  color depth 30(FPS) near far 1
ex> 1 1280 720 2 1 2 30 165 1000 2 (PUMA)<-- MJPEG foramt, 1280 x 720 14 bits  zdTableIndex  30(FPS) near far 1
```

+

6.2 3D Tester Command

1. 3D 0E ,➔ EXIT

   3D for 3D Tester, Entering the command mode.

   0E for Exit

2. 3D ‘/F’ 1000 ‘/N 100 ‘/0’ ,➔ Setup ZD Near and Far

   ‘/F’ for Far

   1000 for Far Max 1000 mm

   100 for Near min 100 mm

   ‘\0’ Ending the 3D Tester Cmd mode.

.3. 3D ‘\S’ value ‘\0’ ➔ Switch Color / Color Depth / Depth(Mono)

‘\S’ : Switch Point Cloud Type Command

Value 0: Color Point Cloud

Value 1: Color Depth Point Cloud

Value 2: Depth (Mono) Point Cloud

4.3D ‘\T’ ‘\0’ ➔ Stop Command

‘\T’ for Stop command

5.3D ‘\Y’ ‘\0’ ➔ Start Command

‘\Y’ for Start command

6.3 EtronDI_InitialCmdFiFo

6.4 EtronDI_CloseCmdFiFo

6.5 EtronDI_WriteCmdFiFo

6.6 EtronDI_ReadCmdFiFo

6.7 EtronDI_InitSRB

6.8 EtronDI_PutSRB

6.9 EtronDI_GetSRB