

CA378-AOIS for Jetson Nano *ソフトウェアセットアップガイド*

Version 1.0.0

Dated: 2019/09/12

Home Page <https://www.centuryarks.com/products/sensor/cm>

日付	バージョン	コメント
2019/09/12	v1.0.0	初版

- 1. 概要
- 2. デモソフト環境設定
- 3. ドライバのインストール
- 4. デモソフトウェアインストール
- 5. デモ実行方法
 - 5. 1. Focus & OIS デモ
 - 5. 2. 12M静止画撮影
 - 5. 3. 静止画撮影
 - 5. 4. 動画撮影
- Appendix
 - A. 1. ファイル構成について
 - A. 2. 設定ファイルについて

本資料は、Jetson Nano上でカーネルドライバのビルド手順とソフトウェアのインストールについて説明します。

Hardware :Jetson Nano(32G microSD)

OS :Ubuntu 18.04 LTS – JetPack 4.2(L4T 32.1)

CSI Hardware :CenturyArks CA378-AOIS(Sony IMX378)

以下の記述は、各環境で実行されるコマンドであることを示しています。

\$...ホストPCで実行されるコマンドです。

#...Jetson Nano上で実行されるコマンドです。

2. デモソフト環境設定

★CA378-AOISのドライバをインストール前に以下の環境の構築を実施してください。

前提条件：

1. Jetpack 4.2のインストール (Linux for Tegra R32.1)
2. sudo権限の設定

手順1. Jetpack 4.2のインストール(Linux for Tegra R32.1)

(1) NVIDIA DEVELOPERサイトでユーザー登録します。

<https://developer.nvidia.com>

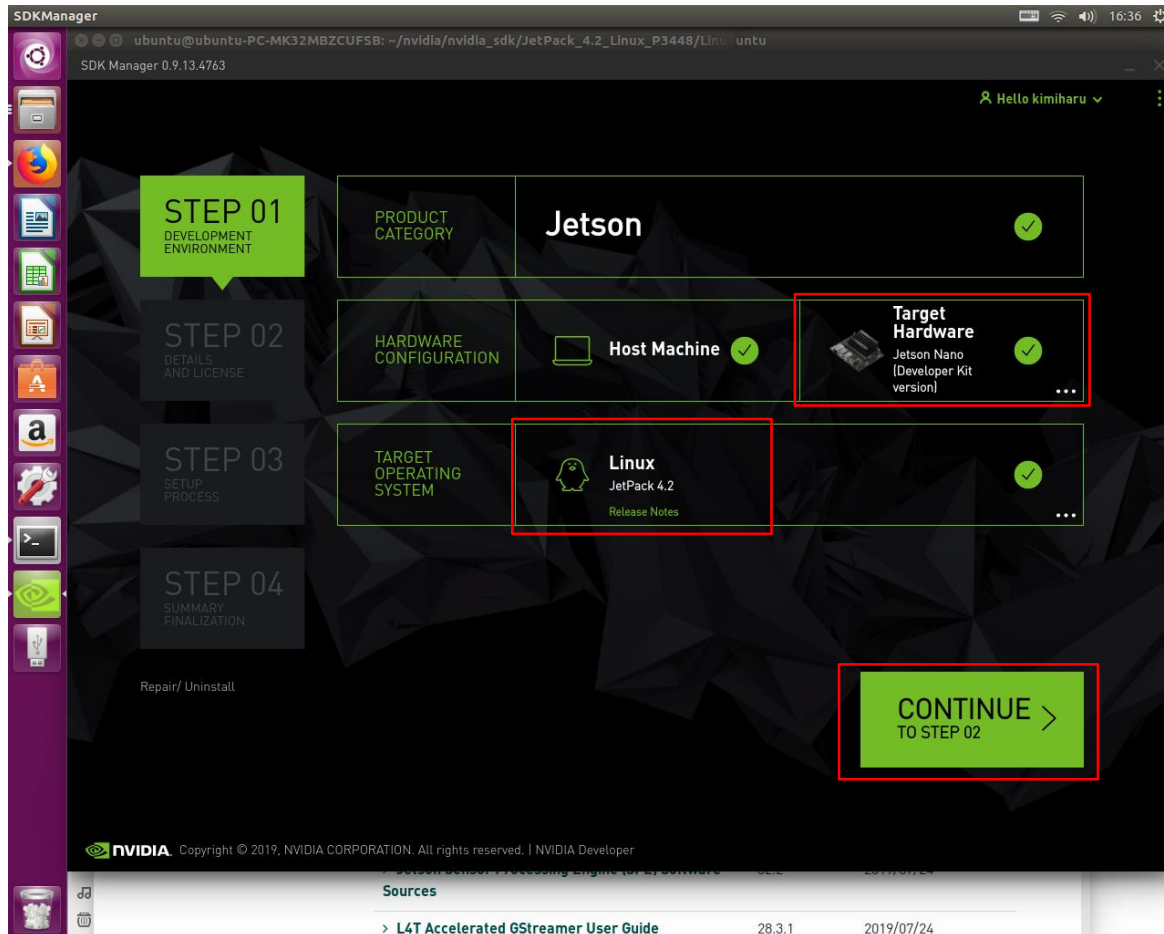
(2) ホストPCでNvidia SDK managerをダウンロードし、インストールします。

<https://developer.nvidia.com/nvidia-sdk-manager>

2. デモソフト環境設定

(3) Nvidia SDK Manager環境設定

対象機器にJetson Nano(P3448)を設定し、対象OSにJetPack4.2を設定します。



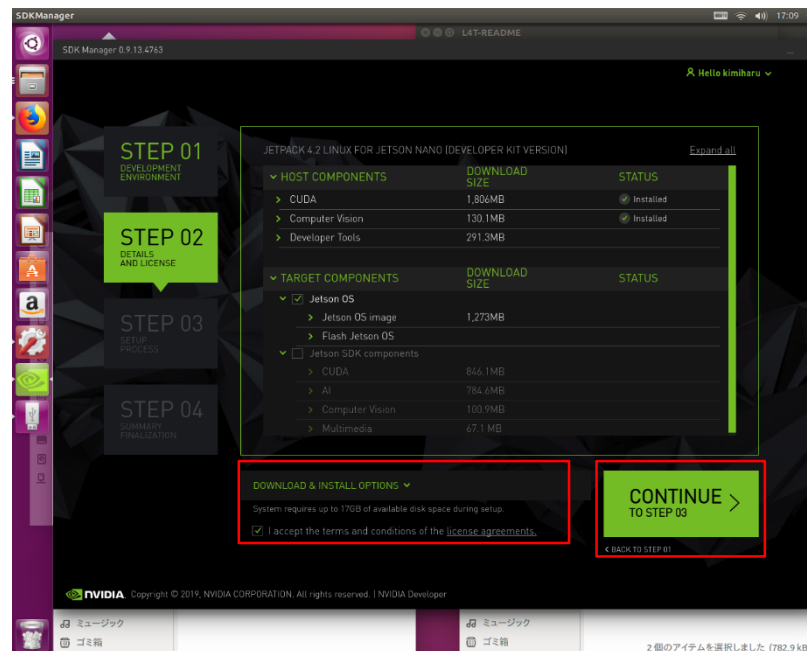
2. デモソフト環境設定

(4) Jetson NanoをUSBリカバリモードで起動します。

1. Jetson Nanoの電源をオフにします。刺さっていれば、ACアダプタをデバイスから抜きます。スタンバイやスリープ状態でなく、電源オフの状態にします。
2. フォースリカバリのピン(J40の3と4)をジャンパーでショートさせます。
3. デバイスにACアダプタを接続します。
4. 開発キットが自動的にフォースリカバリモードで起動します。
5. フォースリカバリモードで起動したら、ジャンパーピンを抜きます。

(5) SDK Managerの詳細とライセンス設定

ライセンスを承諾するにチェックし、continueボタンを押してOSをインストールします。
ただし、このOSイメージは15Gのメモリ領域しかありません。
(カーネルのビルドには15G以上のメモリ領域が必要です。)



2. デモソフト環境設定

(6) 以下に示すコマンドで、32GのOSイメージを生成します。

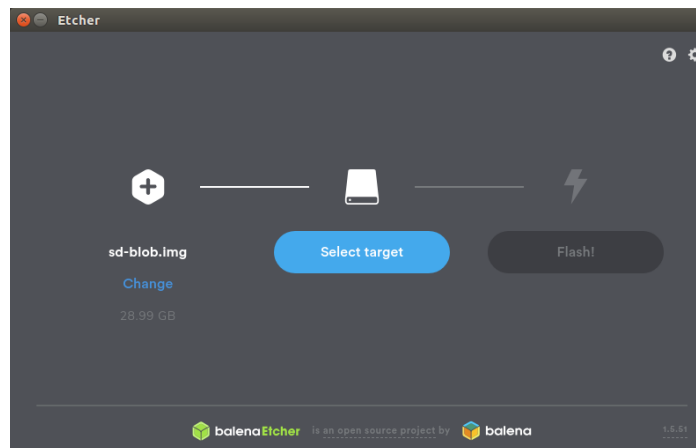
```
$ cd /home/nvidia/nvidia/nvidia_sdk/JetPack-4.2_Linux_GA_P3448/Linux_for_Tegra  
$ sudo ./create-jetson-nano-sd-card-image.sh -o sd-blob.img -s 27G -r 100
```

参照

<https://devtalk.nvidia.com/default/topic/1050105/jetson-nano/jetson-nano-sd-card-partitions-can-not-extend-/2>

(7) 先ほどOSを書き込んだSD cardをフォーマットします。(SD card formatter等のツールで)

(8) Etcher toolでOSイメージを書き込みます。



(9) 作成したSDカードでJetson Nanoを起動します。

ID:nvidia

PASSWORD:nvidia

手順2. sudo権限の設定

(1)以下のコマンドを実行します。

```
# sudo visudo
```

(2) 以下の赤字の行を追記します。

```
# User privilege specification
root  ALL=(ALL:ALL) ALL
nvidia ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL
%nvidia ALL=(ALL:ALL) NOPASSWD: ALL
```

(3) Jetsonを再起動します。

```
# sudo reboot
```

3. ドライバのインストール

・手順概要

1. カーネルソースの準備
2. カーネルのビルドとインストール
3. 新しいデバイスツリー(DTB)の書き込み

手順1. カーネルソースの準備

以下のサイトから「CA378_v1.0.0_Jetpack4.2_Nano_src_build.tar.gz」をJetson Nanoのホームディレクトリへダウンロードし、以下のコマンドを実行します。

[https://github.com/centuryarks/CA378-AOIS/releases/download/JSNano_v1.0.0_L4T32.1\(Jetpack4.2\)/CA378_v1.0.0_Jetpack4.2_Nano_src_build.tar.gz](https://github.com/centuryarks/CA378-AOIS/releases/download/JSNano_v1.0.0_L4T32.1(Jetpack4.2)/CA378_v1.0.0_Jetpack4.2_Nano_src_build.tar.gz)

```
# tar -zxvf CA378_v1.0.0_Jetpack4.2_Nano_src_build.tar.gz  
# cd CA378_v1.0.0_Jetpack4.2_Nano_src_build  
# ./PrepareKernelSources.sh
```

手順2. カーネルのビルドとインストール

以下のコマンドでカーネルをビルド、インストールします。

```
# ./BuildKernelSources.sh
```

3. ドライバのインストール

手順3. 新しいデバイスツリー(DTB)の書き込み

(1)コンパイルしたdtbファイルをホストPCへコピーします。

```
# cd ~/nvidia/nvidia_sdk/JetPack_4.2_Linux_P3448/Linux_for_Tegra/  
# sudo sshpass -p 'nvidia' scp -o StrictHostKeyChecking=no nvidia@192.168.xxx.xxx:/boot/*.dtb ./kernel/dtb/  
# nvidia@192.168.xxx.xxx is IP address on JetsonNano.
```

(2)JetsonNanoをUSBリカバリモードで起動します。

1. Jetson Nanoの電源をオフにします。刺さっていれば、ACアダプタをデバイスから抜きます。スタンバイやスリープ状態でなく、電源オフの状態にします。
2. フォースリカバリのピン(J40の3と4)をジャンパーでショートさせます。
3. デバイスにACアダプタを接続します。
4. 開発キットが自動的にフォースリカバリモードで起動します。
5. フォースリカバリモードで起動したら、ジャンパーピンを抜きます。

(3)Dtbファイルを書き込みます。

```
# cd ~/nvidia/nvidia_sdk/JetPack_4.2_CA_Linux_P3448/Linux_for_Tegra/  
# sudo ./flash.sh -r -k DTB jetson-nano-qspi-sd mmcblk0p1
```

4. デモソフトインストール

以下の手順でインストールを実行してください。

・インストール手順

1. パッケージインストール

```
# sudo apt-get install v4l-utils ufwraw -y  
# sudo apt-get install libgstreamer1.0-0 gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad  
gstreamer1.0-plugins-ugly gstreamer1.0-libav gstreamer1.0-doc gstreamer1.0-tools
```

2. 以下のサイトから「demo_v1.0.0_nano.tar.gz」をダウンロードします。

[https://github.com/centuryarks/Sample/releases/download/JSNano_v1.0.0_L4T32.1\(Jetpack4.2\)/demo_v1.0.0_nano.tar.gz](https://github.com/centuryarks/Sample/releases/download/JSNano_v1.0.0_L4T32.1(Jetpack4.2)/demo_v1.0.0_nano.tar.gz)

```
# wget --no-check-certificate ¥  
https://github.com/centuryarks/Sample/releases/download/JSNano_v1.0.0_L4T32.1(Jetpack4.2)/demo_v1.0.0_nano.tar.gz
```

3. 「demo_v1.0.0_nano.tar.gz」ファイルを解凍してください。

```
# tar -zxvf demo_v1.0.0_nano.tar.gz
```

4. 解凍できたフォルダ内にある「Install.sh」を実行してください。

```
# cd demo  
# ./Install.sh
```

5. デスクトップにショートカットが作成されます。(一度アプリを実行すると、アイコンが表示されます。)

DEMO



JetsonはISPでストリーミング時に120fpsまでしか出せないリミッターがかかっています。
120fps以上でストリーミングを行うには、リミッターを解除する必要があります。
以下の手順でFPSのリミッターを解除してください。

参照

<https://devtalk.nvidia.com/default/topic/1056210/jetson-nano/nvarguscamerasrc-frame-rate-limit/>

1. 以下のサイトからライブラリファイルをダウンロードします。

<https://devtalk.nvidia.com/cmd/default/download-comment-attachment/79559/>

2. ファイルを解凍し、ファイルをコピーします。

```
# tar -zxvf devtalk1056210_Jun26_v2.tar.gz  
# cd devtalk1056210_Jun26_v2  
# sudo cp libgstnvarguscamerasrc.so /usr/lib/aarch64-linux-gnu/gstreamer-1.0/libgstnvarguscamerasrc.so
```

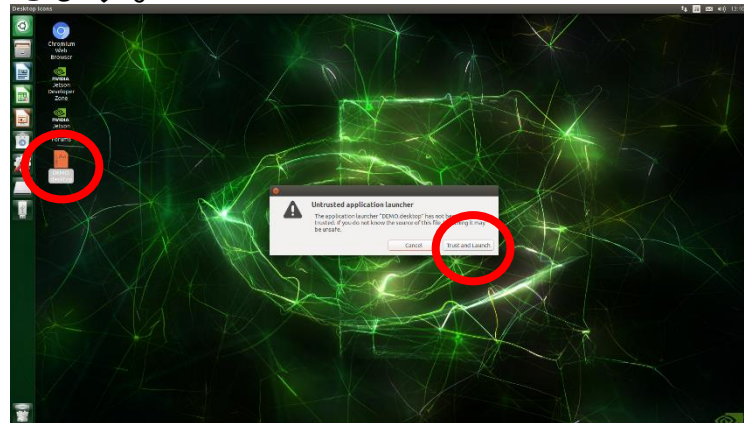
5. デモ実行方法

- 5. 1. Focus & OIS デモ
- 5. 2. 12M静止画撮影
- 5. 3. 静止画撮影
- 5. 4. 動画撮影

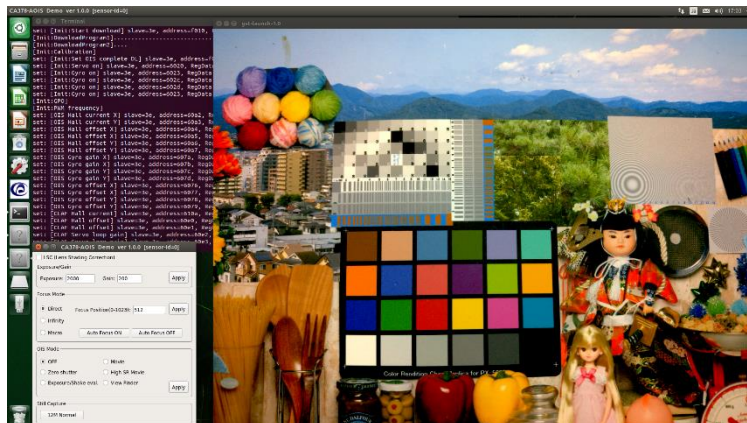
5. 1. Focus & OIS デモ

Focus & OIS デモの起動:

1. デスクトップの「DEMO.desktop」アイコンをクリックし、表示されたダイアログの「Trust and Launch」をクリックします。



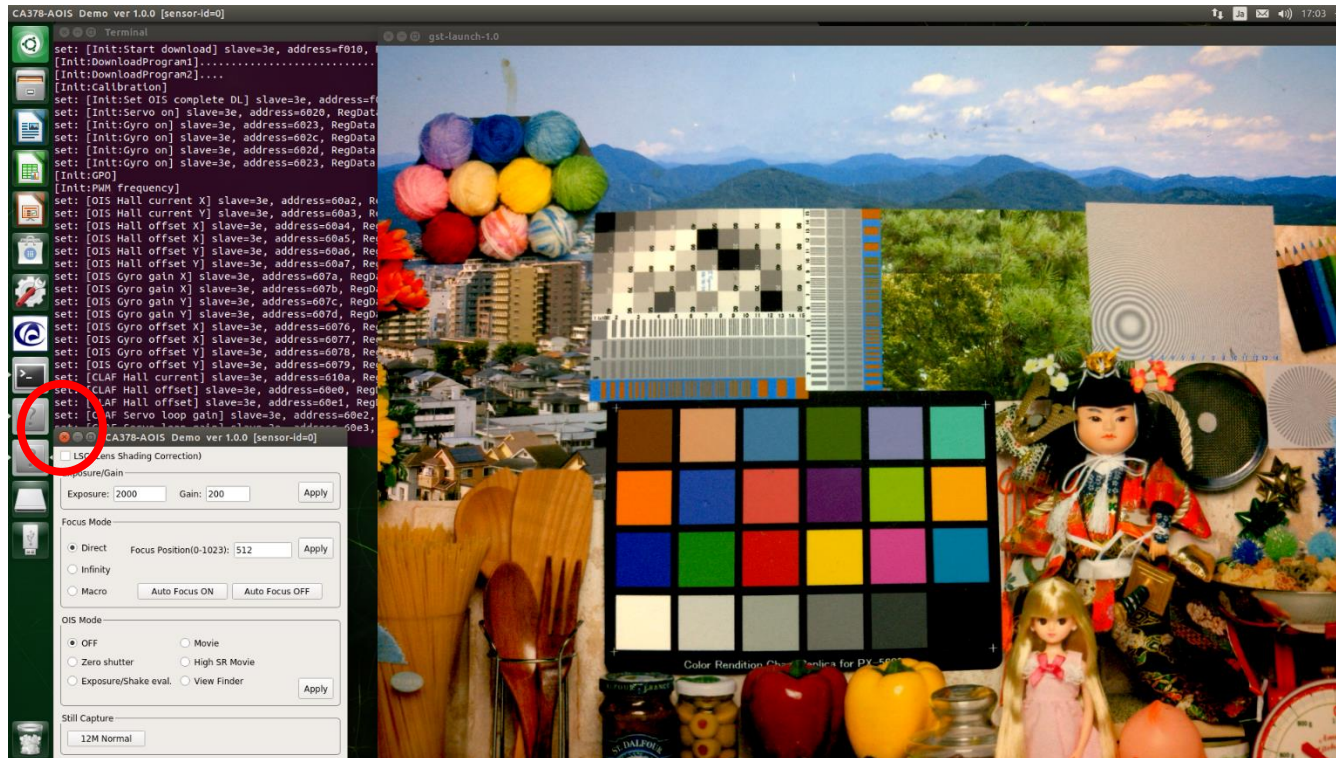
2. しばらくするとGUI画面が表示され、アプリアイコンが変わります。
 3. 被写体の距離を変更したり、カメラを動かしたりして、機能を確認してください。
- ※機能詳細は、P17～P18を参照ください。



5. 1. Focus & OIS デモ

Focus & OIS デモの終了:

1. ×ボタンをクリックします。



5. 1. Focus & OIS デモ

Focus & OIS の各機能について説明します。

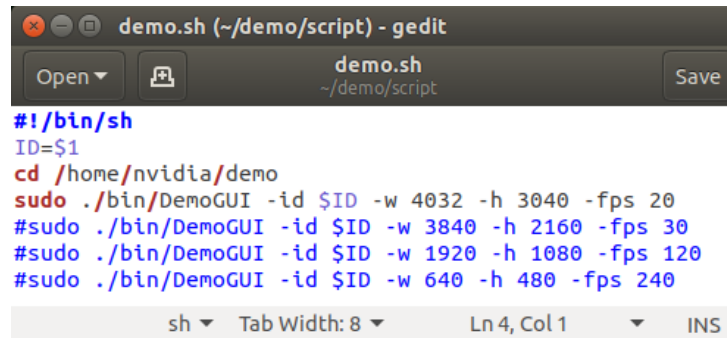


機能	説明
LSC	チェックするとシェーディング補正を有効にします ※理論値を設定してあります
Exposure/Gain	Exposure: 露光時間を設定します(1-65515) Gain: ゲインを設定します(100-2200) Apply: 設定を適用します
Focus Mode	Direct: フォーカス位置を直接指定します Infinity: フォーカス位置を無限遠に設定します Macro: フォーカス位置を近距離に設定します Focus Position : フォーカス位置 Apply: 設定を適用します Auto Focus ON: オートフォーカスを有効にします Auto Focus OFF: オートフォーカスを無効にします ※現在デモ用にデバッグ制御されています。
OIS Mode	OFF: OISを無効にします。 各OISモードに対応します。 Zero Shutter Exposure/Shake eval. Movie High SR Movie View Finder Apply: 設定を適用します
Still Capture	12M Normal: 12M静止画撮影を行います。

5. 1. Focus & OIS デモ

ストリーミングのサイズを変更するには、以下のファイルのパラメータを変更します。

~/demo/script/demo.sh



```
demo.sh (~/.demo/script) - gedit
demo.sh
~/demo/script

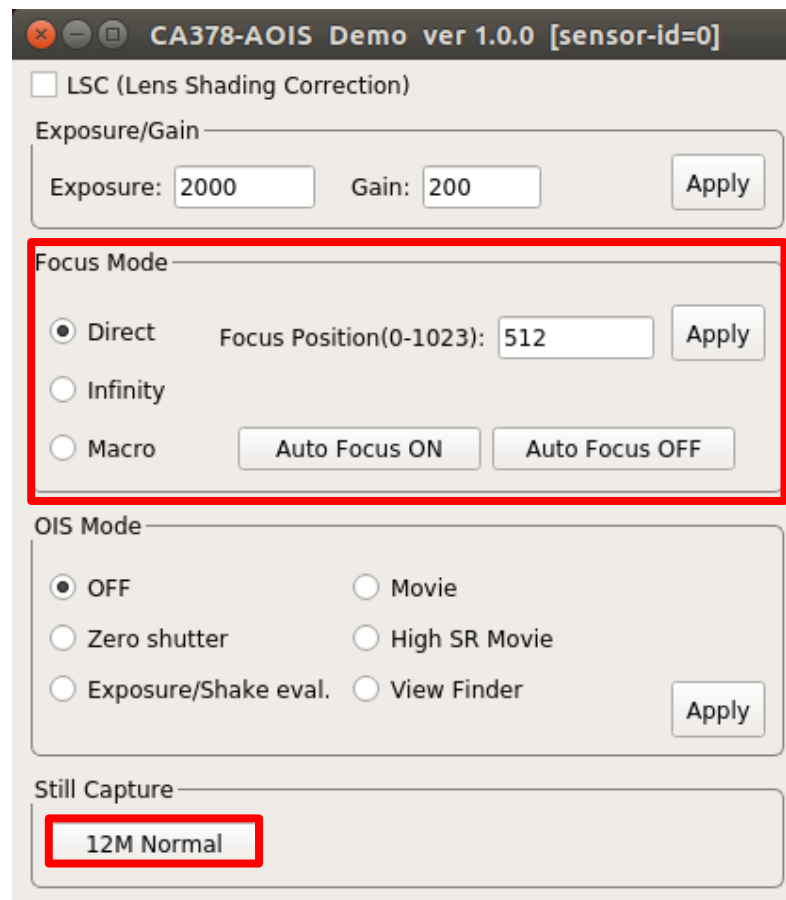
#!/bin/sh
ID=$1
cd /home/nvidia/demo
sudo ./bin/DemoGUI -id $ID -w 4032 -h 3040 -fps 20
#sudo ./bin/DemoGUI -id $ID -w 3840 -h 2160 -fps 30
#sudo ./bin/DemoGUI -id $ID -w 1920 -h 1080 -fps 120
#sudo ./bin/DemoGUI -id $ID -w 640 -h 480 -fps 240
```

Setting	Parameter
4032x3040	Width=4032, Height=3040, fps=20
3840x2160	Width=3840, Height=2160, fps=30
1920x1080	Width=1920, Height=1080, fps=120
640x480	Width=640, Height=480, fps=240

5. 2. 12M静止画撮影

12M静止画撮影の実行:

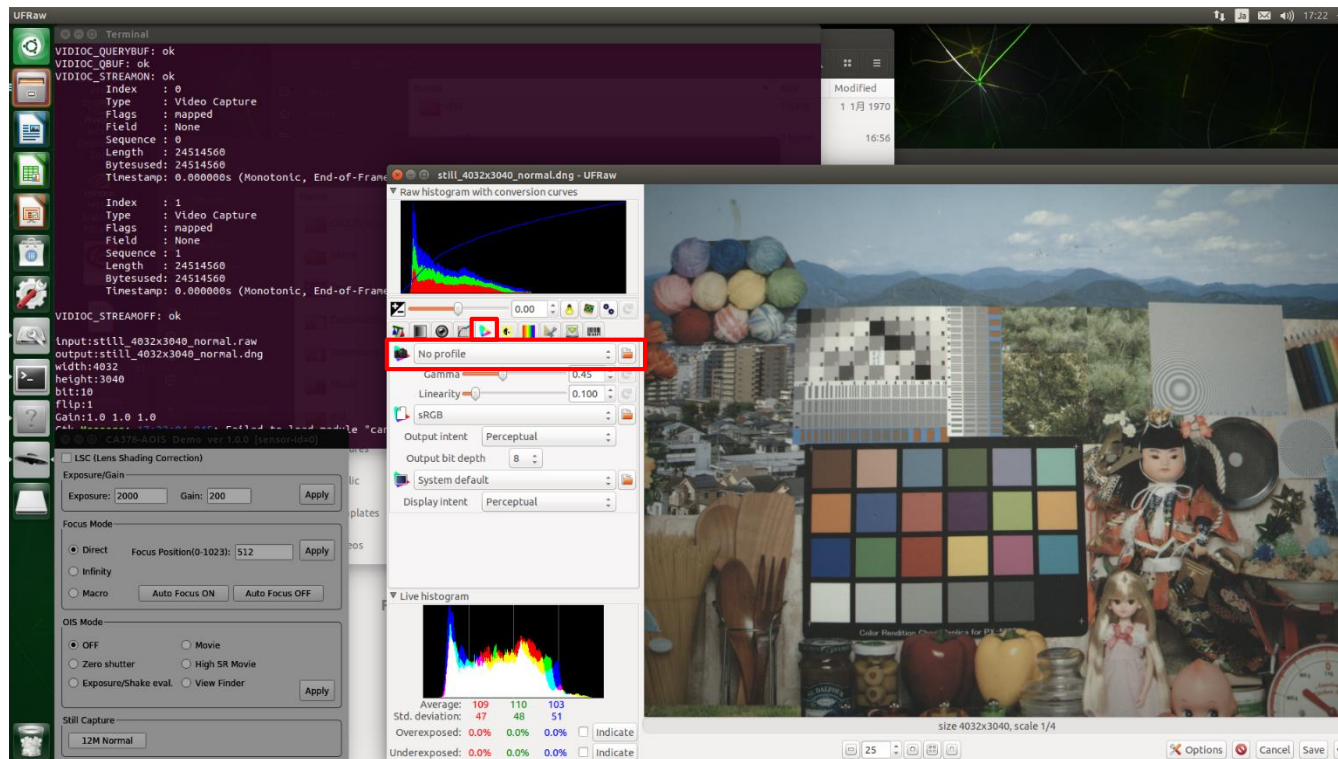
1. フォーカスの調整をしておきます。
(Auto Focus ONにし、焦点が合ったらAuto Focus OFFにすると便利です)
2. [12M Normal] ボタンをクリックします。



5. 2. 12M静止画撮影

3. RAWとDNGフォーマットで撮像が可能です。

4. カラーマネジメントのカメラプロファイル設定を「No profile」に設定してください。

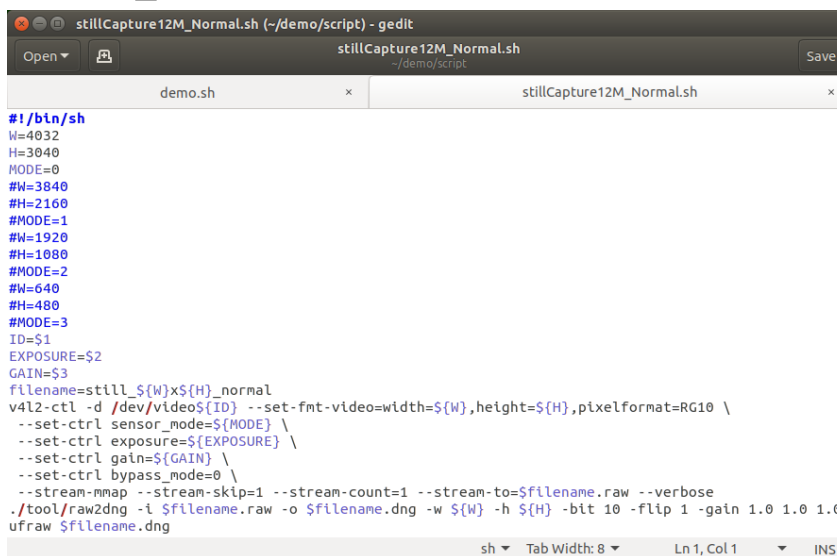


5. 3. 静止画撮影

指定画像サイズで撮影するには、Stillスクリプトを変更してアプリの撮影ボタンを押すか、キャプチャコマンドを実行してください。

Stillスクリプト

~/demo/script/stillCapture12M_Normal.sh



```
#!/bin/sh
W=4032
H=3040
MODE=0
#W=3840
#H=2160
#MODE=1
#W=1920
#H=1080
#MODE=2
#W=640
#H=480
#MODE=3
ID=$1
EXPOSURE=$2
GAIN=$3
filename=still_${W}x${H}_normal
v4l2-ctl -d /dev/video${ID} --set-fmt-video=width=${W},height=${H},pixelformat=RG10 \
--set-ctrl sensor_mode=${MODE} \
--set-ctrl exposure=${EXPOSURE} \
--set-ctrl gain=${GAIN} \
--set-ctrl bypass_mode=0 \
--stream-mmap --stream-skip=1 --stream-count=1 --stream-to=$filename.raw --verbose
./tool/raw2dng -i $filename.raw -o $filename.dng -w ${W} -h ${H} -bit 10 -flip 1 -gain 1.0 1.0 1.0
ufrw $filename.dng
```

キャプチャコマンド

```
#v4l2-ctl -d /dev/video0 --set-fmt-video=width=[Width],height=[Height],pixelformat=RG10 ¥
--set-ctrl sensor_mode=[Mode] ¥
--set-ctrl exposure=1000 ¥
--set-ctrl gain=200 ¥
--set-ctrl bypass_mode=0 ¥
--stream-mmap --stream-skip=1 --stream-count=1 --stream-to=image.raw --verbose
```

Setting	Parameter
4032x3040	Width=4032, Height=3040, Mode=0
3840x2160	Width=3840, Height=2160, Mode=1
1920x1080	Width=1920, Height=1080, Mode=2
640x480	Width=640, Height=480, Mode=3

録画メモリ領域確保

```
#cd ~/demo/script/  
#./ramdisk.sh
```

動画撮影(yuv)

```
#cd ~/demo/script/  
#./yuv_capture.sh /mnt/ram/test 0 4032 3040 20 40
```

Argument	Description
arg1	Movie file name
arg2	Sensor id(Specify 0)
arg3	Width
arg4	Height
arg5	Fps
arg6	Capture frame num

動画再生(yuv)

```
#cd ~/demo/script/  
#./yuv_viewer.sh /mnt/ram/test0.yuv 4032 3040 20
```

Argument	Description
arg1	Movie file path
arg2	Width
arg3	Height
arg4	Fps

Appendix

デモソフトのファイル構成について説明します。

```
demo
├── appicon.png
├── Install.sh
├── bin
│   ├── demo.ini
│   ├── DemoGUI
│   └── preview
├── script
│   ├── demo.sh
│   ├── preview.sh
│   ├── ramdisk.sh
│   ├── raw_capture.sh
│   ├── stillCapture12M_Normal.sh
│   ├── yuv_capture.sh
│   └── yuv_viewer.sh
├── src
│   ├── af_control.c
│   ├── af_control.h
│   ├── communication.h
│   ├── communication_jetson.c
│   ├── debug_util.h
│   ├── demo_control.c
│   ├── demo_control.h
│   ├── DemoGUI.pro
│   ├── lsc_control.c
│   ├── lsc_control.h
│   ├── main.cpp
│   ├── mainwindow.cpp
│   ├── mainwindow.h
│   ├── mainwindow.ui
│   ├── Makefile
│   ├── ois_control.c
│   ├── ois_control.h
│   ├── slave_address.h
│   └── types_util.h
└── tool
    ├── Makefile
    ├── preview.c
    ├── raw2dng
    ├── raw2dng.c
    ├── raw2hdr
    ├── raw2hdr.c
    └── tools.h
```

機能	説明
bin	DemoGUI: デモソフト Demo.ini: デモソフトの設定ファイル
script	スクリプトが記述されています。 仕様に応じてカスタマイズ可能です。 demo.sh preview.sh stillCapture12M_Normal.sh
src	デモソフトのソースコード一式です。
tool	画像ファイル変換ツールが記述されています。

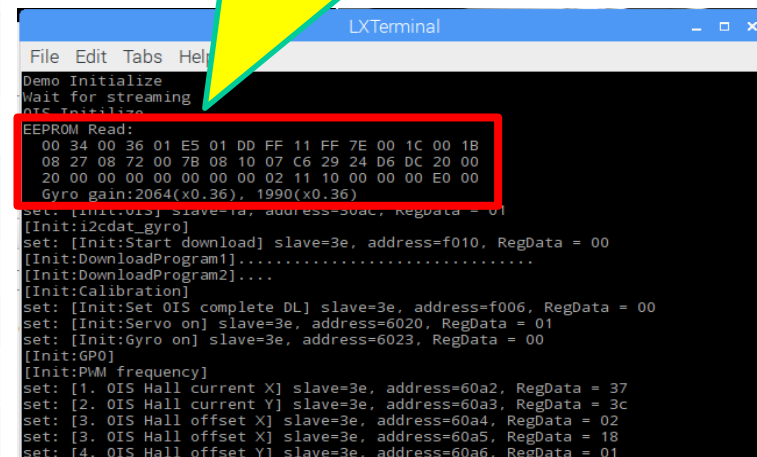
A. 2. 設定ファイルについて

設定ファイルのdemo.iniについて説明します。

```
# DEMO Setting
preview = /home/pi/demo/script/preview.sh
stillCapture12M_Normal = /home/pi/demo/script/stillCapture12M_Normal.sh
gyroGainRateX=1.00
gyroGainRateY=1.00
autoFocusGain=2.0
autoFocusConfidenceThreshold=10
autoFocusMoveLimit=100
AutoFocusAverageNum=1
Exposure=1000
Gain=200
```

機能	説明
preview	プレビュー用のスクリプトPathです。
stillCapture12M_Normal	12M静止画撮影用のスクリプトPathです。
gyroGainRateX gyroGainRateY	EEPROMにOISキャリブレーション結果が書き込まれていた場合のみ有効です。
autoFocusGain	オートフォーカスのゲインを調整します。
autoFocusConfidenceThreshold	Phase Differenceの信頼度の閾値を指定します。
autoFocusMoveLimit	一度のフォーカス移動量を制限します。
AutoFocusAverageNum	オートフォーカスの平均量を調整します。
Exposure	露光時間を調整します。
Gain	ゲインを調整します。

OISキャリブレーション済みの場合は、Terminalに以下のログが出力されます。



```
LXTerminal
File Edit Tabs Help
Demo Initialize
Wait for streaming
OIS Initialize
EEPROM Read:
00 34 00 36 01 E5 01 DD FF 11 FF 7E 00 1C 00 1B
08 27 08 72 00 78 08 10 07 C6 29 24 D6 DC 20 00
20 00 00 00 00 00 00 02 11 10 00 00 00 E0 00
Gyro gain:2064(x0.36), 1990(x0.36)
set: [Init:OIS] slave=3e, address=30ac, RegData = 01
[Init:i2cdat_gyro]
set: [Init:Start download] slave=3e, address=f010, RegData = 00
[Init:DownloadProgram1].....
[Init:DownloadProgram2]....
[Init:Calibration]
set: [Init:Set OIS complete DL] slave=3e, address=f006, RegData = 00
set: [Init:Servo on] slave=3e, address=6020, RegData = 01
[Init:Gyro on] slave=3e, address=6023, RegData = 00
[Init:GPO]
[Init:PWM frequency]
set: [1. OIS Hall current X] slave=3e, address=60a2, RegData = 37
set: [2. OIS Hall current Y] slave=3e, address=60a3, RegData = 3c
set: [3. OIS Hall offset X] slave=3e, address=60a4, RegData = 02
set: [3. OIS Hall offset X] slave=3e, address=60a5, RegData = 18
set: [4. OIS Hall offset Y] slave=3e, address=60a6, RegData = 01
```