1. Abstract

This guide explains how to get started with the MicroVision Android Software Development Kit (SDK) for its 4th Generation PicoP Scanning Engines (PSE). The PicoP Android SDK allows developers to quickly and easily integrate PicoP Scanning Engine control into an Android application. The Android Application can communicate with PSE over USB. The SDK package includes the Java Application Programming Interface (API), SDK libraries, documentation, and sample projects that demonstrate use of basic (PSE) functions. The sample applications have been developed and built using Android Studio 3.4.1.

2. Table of Contents

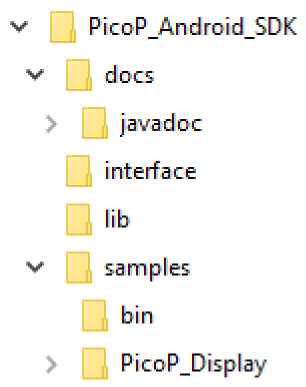
3. PicoP Android SDK

The latest PicoP Android SDK can be cloned or downloaded from:

https://github.com/MicroVision-Inc/Interactive_Projection/tree/master/PicoP_Android_SDK.

Alternatively, the SDK may be delivered as a compressed zip file named PicoP_Android_SDK_Ver_X_Y_Z.zip. The X_Y_Z postfix of the file name represents the version number of the SDK (X equals the major version, Y the minor version, and Z the patch number of the SDK). To install the SDK, simply unzip the file into c:\PicoP_Android_SDK or another directory of your choice.

The downloaded or unzipped destination folder will contain the following subfolders:



```
PicoP_Android_SDK
  docs
    javadoc
  interface
  lib
  samples
    bin
    PicoP_Display
```

The files/folders included in the SDK distribution package are described in more detail below:

PicoP_Android_SDK				*SDK Root folder*
	Docs			
		javadoc	index.html	*Programmer's Reference Guide entry point*
			*.html	*Programmer's Reference Guide documentation*
		DA0140028_PicoP_Gen4_Programmers_Guide.pdf		*High-Level Programmer's Guide*
		EULA.pdf		*SDK End User License Agreement*
	Interface			*PicoP SDK API Interface*
		IPicoP_Api.java		*PicoP SDK interface file with function prototypes*
	Lib			*PicoP SDK Android library*
		PicoP.aar		*PicoP Android Library*
	Samples			
		Bin		*Contains prebuilt apk for sample app*
		PicoP_Display		*Sample app which demonstrates the use of most of the existing Control API's.*
	DA0140036_PicoP_Gen4_Android_SDK_Getting_Started_Guide.pdf			*This Getting Started guide for PicoP Android SDK.*

4. Setup

4.1. *Development System Requirements*

- Ubuntu 16.04 or Windows 10, 64-bit workstation with 4GB RAM
- Android Studio 3.4.1
- Android host system (phone or tablet) with USB host or OTG port running Android 8.0 (API level 26) or higher.

4.2. *Test Environment*

The SDK has been tested with the following setup:

- Build: 64-bit Ubuntu 16.04 and Windows 10
- Sample apps: Google Pixel XL Phone with Android 8.1.0 and Samsung Galaxy S7 with Android 8.0.0
- USB OTG Type-C to Micro-USB Type B Cable
- MicroVision PSE-0407sti-301 and PSE-2407sti-401 Development Kit

5. Setting up Android build

Set up and install Android studio 3.4.1 or later based on the instructions found here:

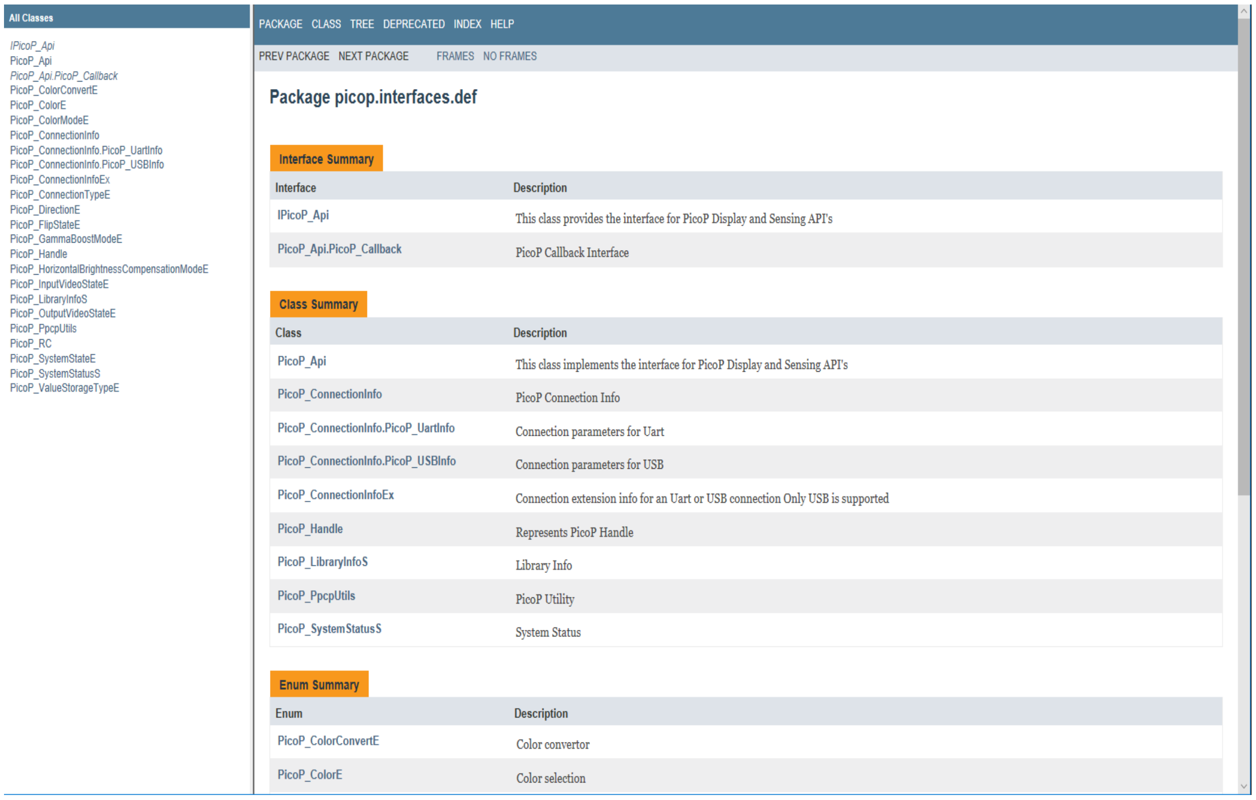
- https://developer.android.com/studio/install

To build the project in Android Studio select "Import Project". Select the location of downloaded/cloned PicoP_Display directory from your workstation and press "OK". This project uses Gradle and it may take few minutes for Gradle to sync and update the build environment.

6. Accessing SDK Documentation

For high level description of functions/commands supported by the PSE Application Programming Interface (API), please refer to the _DA0140028_PicoP_Gen4_Programmers_Guide.pdf_ .

For detailed description of the Java API, please refer to the Programmer's Reference Guide at _javadoc\index.html_. The Reference Guide is a set of hyperlinked JavaDoc containing detailed description of all Function interfaces and definitions provided by the API.


All Classes
PACKAGE CLASS TREE DEPRECATED INDEX HELP
PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

IPicoP_Api
PicoP_Api
PicoP_Api.PicoP_Callback
PicoP_ColorConvertE
PicoP_ColorE
PicoP_ColorModeE
PicoP_ConnectionInfo
PicoP_ConnectionInfo.PicoP_UartInfo
PicoP_ConnectionInfo.PicoP_USBInfo
PicoP_ConnectionInfoEx
PicoP_ConnectionTypeE
PicoP_DirectionE
PicoP_FlipStateE
PicoP_GammaBoostModeE
PicoP_Handle
PicoP_HorizontalBrightnessCompensationModeE
PicoP_InputVideoStateE
PicoP_LibraryInfoS
PicoP_OutputVideoStateE
PicoP_PpcpUtils
PicoP_RC
PicoP_SystemStateE
PicoP_SystemStatusS
PicoP_ValueStorageTypeE

Package picop.interfaces.def

Interface Summary

Interface	Description
IPicoP_Api	This class provides the interface for PicoP Display and Sensing API's
PicoP_Api.PicoP_Callback	PicoP Callback Interface

Class Summary

Class	Description
PicoP_Api	This class implements the interface for PicoP Display and Sensing API's
PicoP_ConnectionInfo	PicoP Connection Info
PicoP_ConnectionInfo.PicoP_UartInfo	Connection parameters for Uart
PicoP_ConnectionInfo.PicoP_USBInfo	Connection parameters for USB
PicoP_ConnectionInfoEx	Connection extension info for an Uart or USB connection Only USB is supported
PicoP_Handle	Represents PicoP Handle
PicoP_LibraryInfoS	Library Info
PicoP_PpcpUtils	PicoP Utility
PicoP_SystemStatusS	System Status

Enum Summary

Enum	Description
PicoP_ColorConvertE	Color convertor
PicoP_ColorE	Color selection

7. Using the PicoP Application Programming Interface (API)

The PicoP SDK for Android allows you to easily interface your application software with the PicoP Scanning Engine. This section explains the steps of the integration process.

7.1. *SDK Version Control*

The SDK API version can be queried with the **`PicoP_GetLibraryInfo()`** function. The Library info returns the major (X), minor (Y), and patch number (Z) for the SDK Version X.Y.Z. The library information also includes capability flags which can differentiate library implementation enhancements that are compatible with the same API and header files.

```
/**
 * Library Info
 */

public class PicoP_LibraryInfoS
{
    /**
     * Contains the major version of the library
     */
    public        byte   majorVersion;
    /**
     * Contains the minor version of the library
     */
    public        byte   minorVersion;
    /**
     * Contains the patch version of the library
     */
    public        byte   patchVersion;
    /**
     * Flags that describe the capability of this version of the PicoP library
     */
    public        int    capabilityFlags;
}
```

7.2. *Step 1: Link SDK Library to the application*

Library is available as \lib\PicoP.aar. To use it in a custom app, please refer to Android guide to add it as a dependency in the project
https://developer.android.com/studio/projects/android-library#AddDependency

7.3. *Step 2: Initialize the SDK Library*

First step is to create PicoP_Handle USB handle. Android application context has to be passed to the connection for subsequent API calls. Secondly set PicoP_Handle to PicoP_OpenLibrary call. After successful return from this call create a connection by calling PicoP_OpenConnection API.

7.4. *Step 3: Connect to PicoP*

After successful initialization of the library, the next step is to create a connection to the PicoP device. The connection can be established using Universal Serial Bus (USB) OTG. To create a connection to the PicoP Scanning Engine, call the **`PicoP_OpenConnection()`**. Upon successful connection, the library will return a connection handle to be used with subsequent library calls. The connection handle identifies the connected system.

7.5. *Step 4: Configure and Control PicoP*

The PicoP API is split into the following functional categories:

- **Connection & Library Management**: Connect to PicoP over USB interface.
- **Display Control**: Configure and control the PicoP Display.
- **System Management**: Manage the PicoP System, Firmware upgrades, Event Log, etc.
- **Input Control**: Configure the PicoP Input Video.
- **3D Sensing:** Configure and control PicoP 3D Sensing interface and retrieve 3D point cloud data.
- **Rendering**: Render images and test patterns into Framebuffer and On-Screen Display (OSD).

7.5.1. Connection Management

The Connection Management Functions are used to connect to the PicoP Display Engine using USB or UART.

Function	Description
PicoP_OpenLibrary()	Opens the library and allocates resources necessary for operation. It returns a handle to the library that must be used in subsequent calls
PicoP_GetLibraryInfo( )	Retrieves the version and capability information of the SDK Library
PicoP_CloseLibrary()	Closes the library and releases all resources. It also closes all the open connections.

PicoP_OpenConnection()	Opens a connection to the PicoP Display Engine using either USB or UART.
PicoP_CloseConnection()	Closes a previously opened connection to the PicoP Display Engine.

7.5.2. Display Control Functions

The Display Control Functions can be used to configure the output display.

Function	Description
PicoP_SetBrightness( )	Sets brightness for the output display.
PicoP_GetBrightness( )	Returns brightness setting of the output display.
PicoP_SetColorMode( )	Sets color mode for the output display.
PicoP_GetColorMode( )	Returns color mode setting of the output display.
PicoP_SetGamma( )	Sets gamma value for the output display.
PicoP_GetGamma( )	Returns gamma value setting of the output display.
PicoP_SetVideoGammaBoostMode( )	Set the color specific video gamma boost mode.
PicoP_GetVideoGammaBoostMode( )	Return the color specific video gamma boost mode.
PicoP_SetMicroWeaveMode( )	Select the MicroWeave mode.
PicoP_GetMicroWeaveMode( )	Return the MicroWeave mode.
PicoP_SetFlipState()	Sets the flip state of the image to horizontal, vertical, both horizontal and vertical or none.
PicoP_GetFlipState()	Returns the current flip state of the output display.
PicoP_SetOutputVideoState( )	Set the video output state to enabled or disabled.
PicoP_GetOutputVideoState( )	Returns the current video output state.
PicoP_GetOutputVideoProperties( )	Return output video properties.
PicoP_SetBiPhase( )	Sets the scan line phase delay to align the forward and reverse scan video.

PicoP_GetBiPhase( )	Returns the scan line phase delay setting.
PicoP_SetColorAlignment()	Performs vertical or horizontal color alignment for the selected color
PicoP_GetColorAlignment()	Gets the color alignment offset of the chosen color
PicoP_SetColorConverter()	Sets the color converter values
PicoP_GetColorConverter()	Gets the color converter values
PicoP_SetFrameRateMode( )	Sets the output frame rate mode (frame rate vs. vertical display and sensing resolutions) and display scaling.
PicoP_GetFrameRateMode( )	Returns the output frame rate mode (frame rate vs. vertical display and sensing resolutions) and display scaling

7.5.3. System Management Functions

The System Management Functions are used to control the PicoP System and to access the system information.

Function	Description
PicoP_GetSystemInfo( )	Retrieves system information.
PicoP_GetSystemStatus( )	Retrieves the system status.
PicoP_GetEventLog( )	Retrieves the system event log.
PicoP_RestoreFactoryConfig( )	Restores System Settings to Factory Configuration.
PicoP_CommitAll( )	Commits all user settings to flash.

7.5.4. Input Control Functions

The Input Control functions are used to configure the PicoP Input Video.

Function	Description
PicoP_GetInputVideoProperties( )	Returns detected input video Frame Rate and Lines per Frame.
PicoP_SetInputVideoState( )	Enables or Disables the input video. When input video is disabled, the framebuffer will not be updated and the output video will contain the last captured frame.
PicoP_GetInputVideoState( )	Returns the current state of the input video.

7.5.1. 3D Sensing Functions

The 3D Sensing functions are used to configure the 3D sensor as well as to retrieve the 3D depth data.

Function	Description
PicoP_SetSensingState( )	Turns the 3D Sensing function on or off. When the 3D Sensing function is off, the IR laser is not pulsing and no 3D sensing data is sent over the USB.
PicoP_GetSensingState( )	Returns the current Sensing state.

7.5.2. Rendering Functions

The Rendering function allows the host system to render information into the PicoP On-Screen Display (OSD) or FrameBuffer.

Function	Description
PicoP_DrawTestPattern( )	Displays one of the built-in test patterns.

7.6. *Step 5: Exit Application*

To gracefully exit the host application, call the **`PicoP_CloseConnection()`** and **`PicoP_CloseLibrary()`** functions to shut down the connection to PicoP and to release all resources used by the library.

8. Sample code

To use PicoP.aar library in a new Android app module, add the file to your project as following:

- Click File > New > New Module.
- Click “Import .JAR/.AAR Package” then click Next
- Enter the location of the compiled PicoP.aar file then click Finish.
- Make sure the library is listed in the dependencies block of build.gradle (Library name: PicoP).
- For more information refer to: https://developer.android.com/studio/projects/android-library#AddDependency
- Import interface definitions from Package picop.interfaces.def
- Provide USB permission in the app to communicate with the Development kit as described in: https://developer.android.com/guide/topics/connectivity/usb/host
- For USB permissions `vendor-id=0x148A and product-id= 0x0004`

The below listing shows the sample code to establish the connection and issue PicoP_SetInputVideoState request.

```
import picop.interfaces.def.*;
       …
        Context appContext = getApplicationContext();
        PicoP_LibraryInfoS libraryInfo = new PicoP_LibraryInfoS();
        PicoP_Handle connectionHandle = new PicoP_Handle(PicoP_ConnectionTypeE.eUSB);
        PicoP_Api picoPApi = new PicoP_Api();
        PicoP_ConnectionInfo connectionInfo = connectionHandle.getConnectInfo();
connectionInfo.setConnectionContext(appContext);
        PicoP_RC result = picoPApi.PicoP_OpenLibrary(connectionHandle);
        result = picoPApi.PicoP_OpenConnection(connectionHandle,
PicoP_ConnectionTypeE.eUSB, connectionInfo);
        result = picoPApi.PicoP_SetInputVideoState(connectionHandle,
PicoP_InputVideoStateE.eINPUT_VIDEO_ENABLED, true);
        result = picoPApi.PicoP_CloseConnection(connectionHandle,
PicoP_ConnectionTypeE.eUSB);
```

9. Building and Running test app

Run/Install the app on the device based on the instructions found here:
https://developer.android.com/studio/run/device

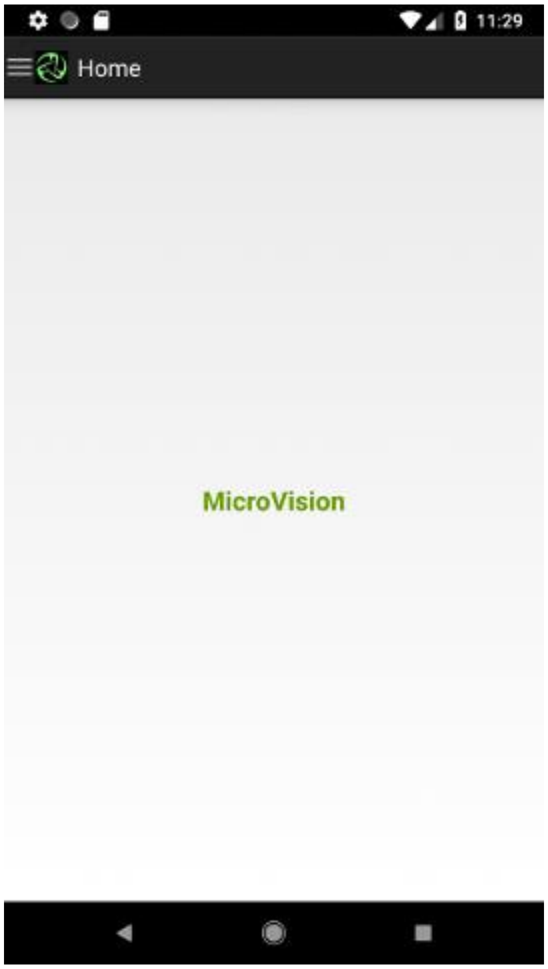
Alternatively we can use adb tool to install prebuilt apk. APK can be found in PicoPAndroidSDK/samples/bin/. For information on adb tool installation for various platforms refer to:
https://developer.android.com/studio/command-line/adb

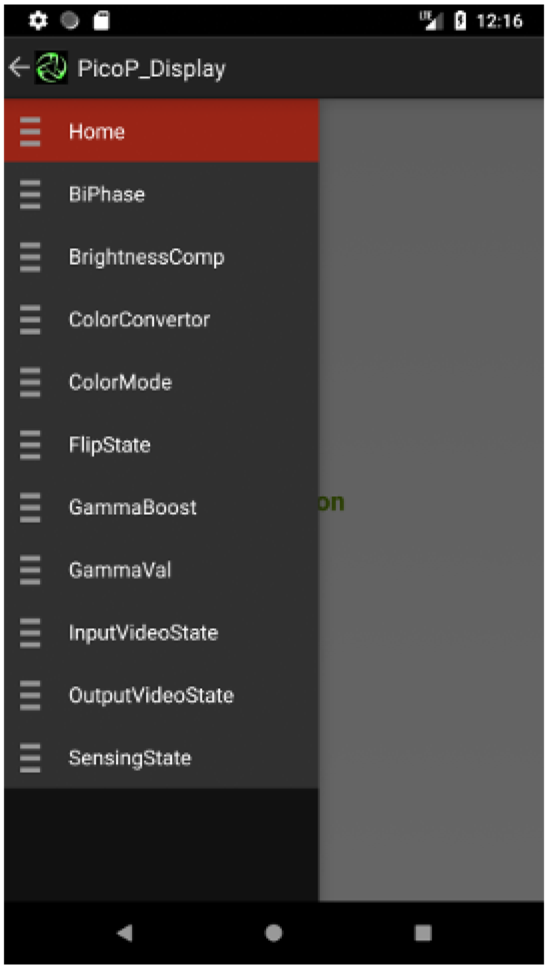
Before installing app on a device, enable USB debugging on the device. This is under Settings > Developer options. Note: On Android 4.2 and newer, Developer options is hidden by default. To make it available, go to Settings > About phone and tap Build number seven times. Return to the previous screen to find Developer options. Once device is set up and connected via USB, install the app using the adb tool:

```
adb -d install path/to/_app.apk
```

After installation is completed, connect the Dev Kit USB connectors to the Android device using OTG. Provide USB permissions for the app in the pop up as shown in the USB permissions screenshot below.

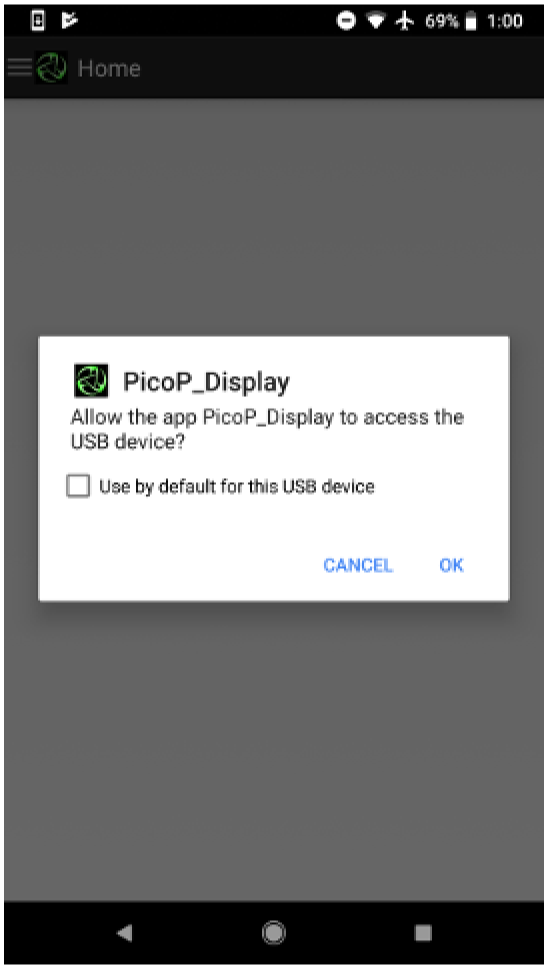
If there is an issue with USB permissions, there will be error message seen when the user executes a get/set control command.





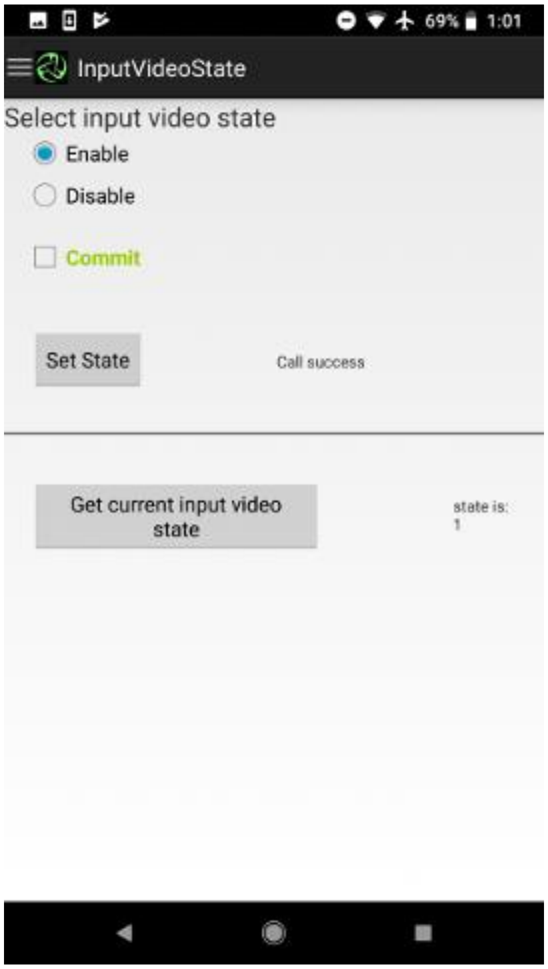
Home Screen

Navigation Drawer List



Allow the app PicoP_Display to access the USB device?

☐ Use by default for this USB device



Select input video state

● Enable

○ Disable

☐ Commit

Call success

state is:
1

USB Permissions Confirmation Dialog

Result as seen after successful connection and Get/Set Call