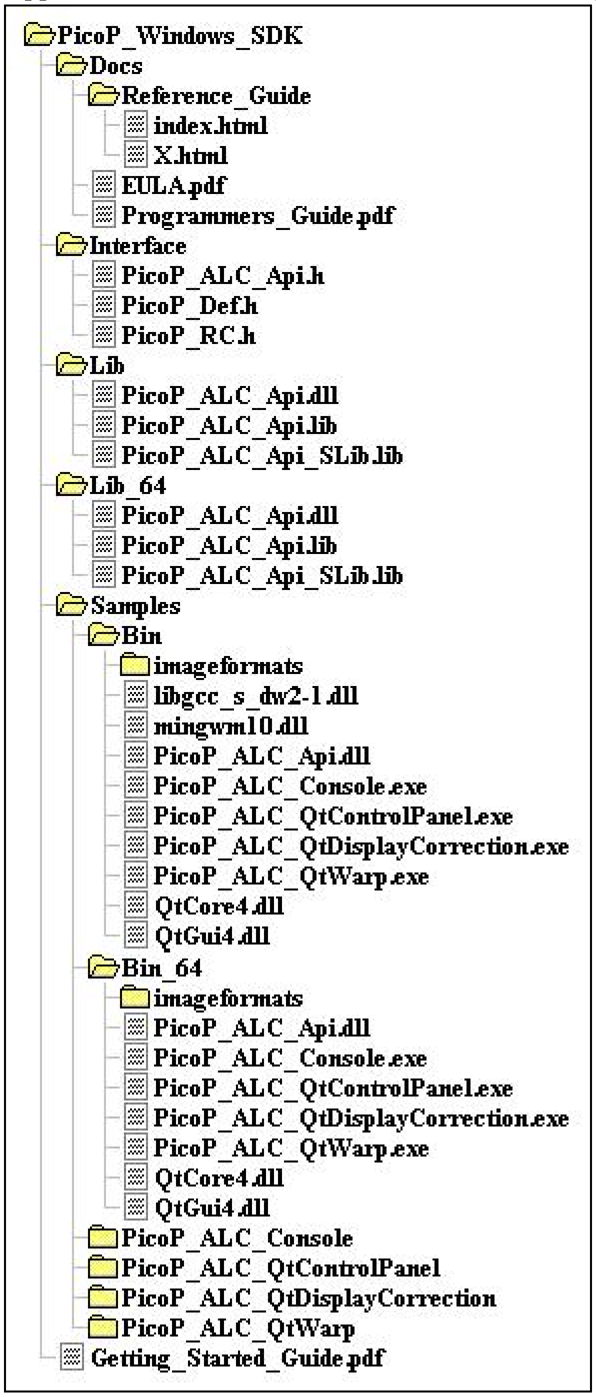
1. Abstract

This document provides instructions on how to get started with the PicoP Software Development Kit (SDK) for Windows. The PicoP SDK for Windows allows developers to quickly and easily integrate PicoP Projector control into a Windows application. The Windows Application can communicate with PicoP Display Engine over USB or UART. The SDK package includes the C Application Programming Interface (API), SDK libraries, documentation, and sample projects that demonstrate use of some basic PicoP Display Engine (PDE) functions. The sample console applications have been developed and built using Microsoft Visual Studio 2010(Ultimate). The sample Qt demo applications are built with QtCreator 2.0.0.

2. Table of Contents

3. Installing PicoP SDK

The PicoP Windows SDK is delivered as a compressed zip file named *PicoP_Windows_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_Windows_SDK* or another directory of your choice. The unzipped destination folder will contain the following:



```
PicoP_Windows_SDK
├─Docs
│ ├─Reference_Guide
│ │ ├─index.html
│ │ └─X.html
│ ├─EULA.pdf
│ └─Programmers_Guide.pdf
├─Interface
│ ├─PicoP_ALC_Api.h
│ ├─PicoP_Def.h
│ └─PicoP_RC.h
├─Lib
│ ├─PicoP_ALC_Api.dll
│ ├─PicoP_ALC_Api.lib
│ └─PicoP_ALC_Api_SLib.lib
├─Lib_64
│ ├─PicoP_ALC_Api.dll
│ ├─PicoP_ALC_Api.lib
│ └─PicoP_ALC_Api_SLib.lib
├─Samples
│ ├─Bin
│ │ ├─imageformats
│ │ ├─libgcc_s_dw2-1.dll
│ │ ├─mingwm10.dll
│ │ ├─PicoP_ALC_Api.dll
│ │ ├─PicoP_ALC_Console.exe
│ │ ├─PicoP_ALC_QtControlPanel.exe
│ │ ├─PicoP_ALC_QtDisplayCorrection.exe
│ │ ├─PicoP_ALC_QtWarp.exe
│ │ ├─QtCore4.dll
│ │ └─QtGui4.dll
│ ├─Bin_64
│ │ ├─imageformats
│ │ ├─PicoP_ALC_Api.dll
│ │ ├─PicoP_ALC_Console.exe
│ │ ├─PicoP_ALC_QtControlPanel.exe
│ │ ├─PicoP_ALC_QtDisplayCorrection.exe
│ │ ├─PicoP_ALC_QtWarp.exe
│ │ ├─QtCore4.dll
│ │ └─QtGui4.dll
│ ├─PicoP_ALC_Console
│ ├─PicoP_ALC_QtControlPanel
│ ├─PicoP_ALC_QtDisplayCorrection
│ └─PicoP_ALC_QtWarp
└─Getting_Started_Guide.pdf
```

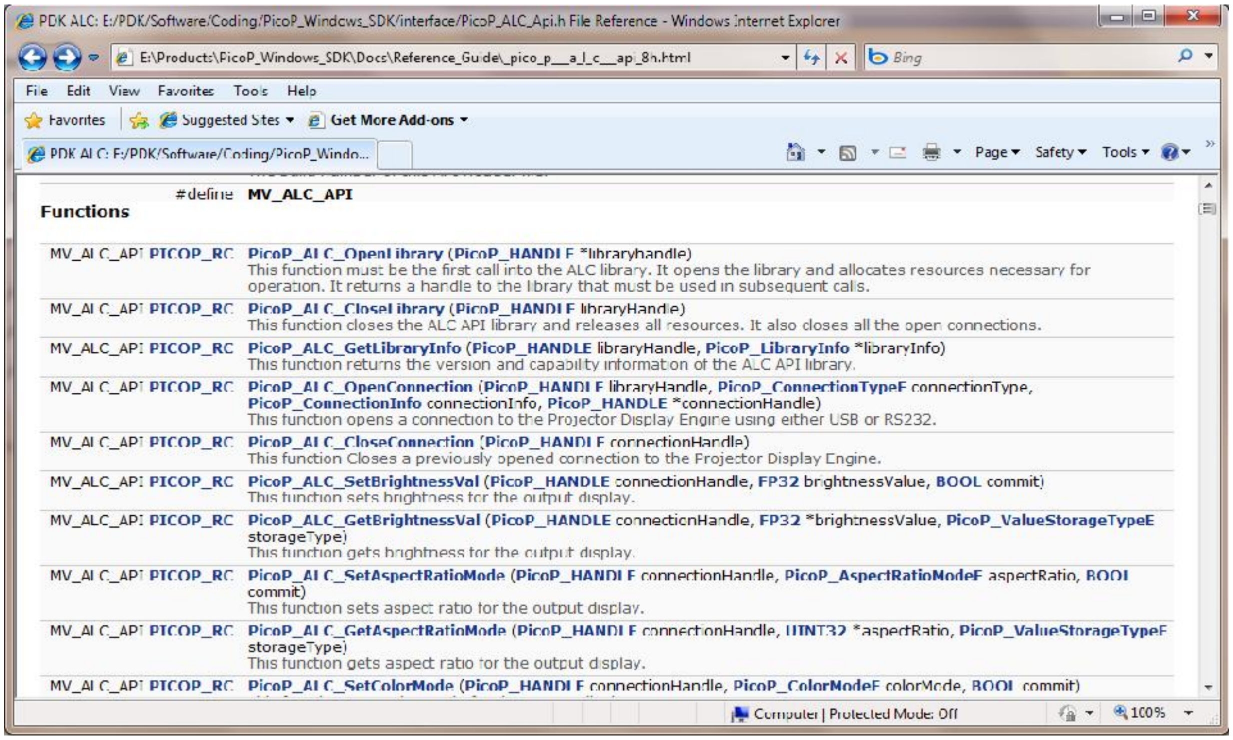
The files/folders included in the SDK distribution package are described in more detail below:

PicoP_Windows_SDK				*SDK Root folder*
	Docs			
		Reference_Guide		*Detailed Programmer's Reference Guide*
			index.html	*Programmer's Reference Guide entry point*
			*.html	*Programmer's Reference Guide documentation*
		Programmers_Guide.pdf		*High-Level Programmer's Guide*
		EULA.pdf		*SDK End User License Agreement*
	Interface			
		PicoP_ALC_Api.h		*PicoP SDK header file*
		PicoP_Def.h		*PicoP SDK definitions files*
		PicoP_RC.h		*PicoP SDK return codes*
	Lib			*PicoP SDK 32 bit Windows libraries*
		PicoP_ALC_Api.dll		*PicoP SDK dynamic library*
		PicoP_ALC_Api.lib		*PicoP SDK import library*
		PicoP_ALC_Api_SLib.lib		*PicoP SDK static library*
	Lib_64			*PicoP SDK 64 bit Windows libraries*
		PicoP_ALC_Api.dll		*PicoP SDK dynamic library*
		PicoP_ALC_Api.lib		*PicoP SDK import library*
		PicoP_ALC_Api_SLib.lib		*PicoP SDK static library*
	Samples			
		Bin		*Contains the 32 bit executables for the sample applications.*
		Bin_64		*Contains the 64 bit executables for the sample applications.*
		PicoP_ALC_Console		*Simple Console Application demonstrating the basic usage of the SDK native C APIs*
		PicoP_ALC_QtControlPanel		*More Advanced Qt Demo project demonstrating use of additional API functions*
		PicoP_ALC_QtDisplayCorrection		*More Advanced Qt Demo project demonstrating use of additional API functions*
		PicoP_ALC_QtWarp		*More Advanced Qt Demo project demonstrating use of additional API functions*
	Getting_Started_Guide.pdf			*This Getting Started guide for PicoP Windows SDK.*

4. SDK Documentation

For high level description of functions/commands supported by the PDE Application Programming Interface (API), please refer to the *Docs\Programmers_Guide.pdf.*

For detailed description of the C-language API, please refer to the Programmer's Reference Guide at *Docs\Reference_Guide\index.html*. The Reference Guide is a set of hyperlinked HTML files containing detailed description of all Function interfaces and definitions provided by the API.


PDK ALC: E:/PDK/Software/Coding/PicoP_Windows_SDK/interface/PicoP_ALC_Api.h File Reference - Windows Internet Explorer
E:\Products\PicoP_Windows_SDK\Docs\Reference_Guide_pico_p___a_l_c___api_8h.html
Bing
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Get More Add-ons
PDK ALC: E:/PDK/Software/Coding/PicoP_Windo...
Page
Safety
Tools
Computer | Protected Mode: Off
100%

#define **MV_ALC_API**

Functions

MV_ALC_API **PICOP_RC** **PicoP_ALC_OpenLibrary** (**PicoP_HANDLE** *libraryhandle)
This function must be the first call into the ALC library. It opens the library and allocates resources necessary for operation. It returns a handle to the library that must be used in subsequent calls.

MV_ALC_API **PICOP_RC** **PicoP_ALC_CloseLibrary** (**PicoP_HANDLE** libraryHandle)
This function closes the ALC API library and releases all resources. It also closes all the open connections.

MV_ALC_API **PICOP_RC** **PicoP_ALC_GetLibraryInfo** (**PicoP_HANDLE** libraryHandle, **PicoP_LibraryInfo** *libraryInfo)
This function returns the version and capability information of the ALC API library.

MV_ALC_API **PICOP_RC** **PicoP_ALC_OpenConnection** (**PicoP_HANDLE** libraryHandle, **PicoP_ConnectionTypeE** connectionType, **PicoP_ConnectionInfo** connectionInfo, **PicoP_HANDLE** *connectionHandle)
This function opens a connection to the Projector Display Engine using either USB or RS232.

MV_ALC_API **PICOP_RC** **PicoP_ALC_CloseConnection** (**PicoP_HANDLE** connectionHandle)
This function Closes a previously opened connection to the Projector Display Engine.

MV_ALC_API **PICOP_RC** **PicoP_ALC_SetBrightnessVal** (**PicoP_HANDLE** connectionHandle, **FP32** brightnessValue, **BOOL** commit)
This function sets brightness for the output display.

MV_ALC_API **PICOP_RC** **PicoP_ALC_GetBrightnessVal** (**PicoP_HANDLE** connectionHandle, **FP32** *brightnessValue, **PicoP_ValueStorageTypeE** storageType)
This function gets brightness for the output display.

MV_ALC_API **PICOP_RC** **PicoP_ALC_SetAspectRatioMode** (**PicoP_HANDLE** connectionHandle, **PicoP_AspectRatioModeE** aspectRatio, **BOOL** commit)
This function sets aspect ratio for the output display.

MV_ALC_API **PICOP_RC** **PicoP_ALC_GetAspectRatioMode** (**PicoP_HANDLE** connectionHandle, **UINT32** *aspectRatio, **PicoP_ValueStorageTypeE** storageType)
This function gets aspect ratio for the output display.

MV_ALC_API **PICOP_RC** **PicoP_ALC_SetColorMode** (**PicoP_HANDLE** connectionHandle, **PicoP_ColorModeE** colorMode, **BOOL** commit)

5. The PicoP Application Programming Interface (API)

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

The same PicoP SDKs can be used with both first and second generation PicoP Display Engines (PDEs), also known as PDE1 and PDE2. However, not all API functions are supported on each platform generation. The API description tables also indicate whether the function is supported by first generation PDE1 and/or second generation PDE2.

5.1. SDK Version Control

The SDK API version is specified in the API Header file

```
/// The major version number of this API header file
#define ALC_API_VERSION_MAJOR 0
/// The minor version number of this API header file
#define ALC_API_VERSION_MINOR 12
```

In addition to the API version, the library also contains a version that can be queried with the `PicoP_ALC_GetLibraryInfo()` function. The Major and Minor version of the library should match the Major and Minor version of the API. The library version information also includes a Patch number and capability flags which can differentiate library implementation enhancements that are compatible with the same API and header files.

```
/** Library information */
typedef struct{
   UINT8  majorVersion;    /**< Contains the major version of the library */
   UINT8  minorVersion;    /**< Contains the minor version of the library */
   UINT8  patchVersion;    /**< Contains the patch version of the library */
   UINT32 capabilityFlags; /**< Flags that describe the capability of the library */
} PicoP_LibraryInfo;
```

5.2. Step 1: Initialize the SDK Library

The first step in connecting to a PicoP is to initialize the PicoP library by calling `PicoP_ALC_OpenLibrary().` The `PicoP_ALC_OpenLibrary()` function returns a handle to the library that can be used to open a connections to the PicoP device.

5.3. Step 2: 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 either USB or UART physical interfaces (For more information on the USB and UART Physical Interfaces, please refer to the PDE Interface Control Document (ICD)). To connect, call the `PicoP_ALC_OpenConnection()` function. Upon successful connection, the library will return a connection handle to be used with subsequent library calls. The connection handle identifies the connected PicoP.

5.4. Step 3: Configure and Control PicoP

The PicoP API is split into the following functional categories:

- **Connection Management**: Connect to PicoP over USB or UART.
- **Display Control**: Configure the PicoP Display.
- **Input Control**: Configure the PicoP Input Video.
- **Rendering**: Render images and test patterns into Framebuffer and On-Screen Display (OSD).
- **System Management**: Manage the PicoP System, Firmware upgrades, Event Log, etc.

5.4.1. Connection Management

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

Function	Description	PicoP Gen1	PicoP Gen2
PicoP_ALC_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_ALC _CloseLibrary( )	Closes the library and releases all resources. It also closes all the open connections.	✓	✓
PicoP_ALC _OpenConnection( )	Opens a connection to the PicoP Display Engine using either USB or UART.	✓	✓
PicoP_ALC _CloseConnection( )	Closes a previously opened connection to the PicoP Display Engine.	✓	✓
PicoP_ALC_EnumerateDevices()	Enumerates PicoP devices connected through USB.	✓	✓
PicoP_ALC_OpenConnectionUSB()	Opens a connection to the PicoP Display Engine using USB.	✓	✓
PicoP_ALC_OpenConnectionRS232()	Opens a connection to the PicoP Display Engine using UART.	✓	✓

5.4.2. Display Control Functions

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

Function	Description	PicoP Gen1	PicoP Gen2
PicoP_ALC _SetBrightnessVal( )	Sets brightness for the output display.	✓	✓
PicoP_ALC _GetBrightnessVal( )	Returns brightness setting of the output display.	✓	✓
PicoP_ALC _SetAspectRatioMode( )	Sets aspect ratio for the output display.	✓	✓
PicoP_ALC _GetAspectRatioMode( )	Returns aspect ratio setting of the output display.	✓	✓
PicoP_ALC _SetColorMode( )	Sets color mode for the output display.	✓	✓
PicoP_ALC _GetColorMode( )	Returns color mode setting of the output display.	✓	✓
PicoP_ALC _SetGammaVal( )	Sets gamma value for the output display.	✓	✓
PicoP_ALC _GetGammaVal( )	Returns gamma value setting of the output display.	✓	✓
PicoP_ALC _FlipImage( )	Flips the image horizontally or vertically.	✓	✓
PicoP_ALC_SetFlipState()	Sets the flip state of the image to horizontal, vertical, both horizontal and vertical or none.	✓	✓
PicoP_ALC_GetFlipState()	Returns the current flip state of the output display.	✓	✓
PicoP_ALC_SetupWarp()	Sets up the input and destination size for Warp	✓	✓
PicoP_ALC_GetWarpSetup( )	Gets warp set up.	×	✓
PicoP_ALC _WarpImage( )	Applies a warp operation to the image.	✓	✓
PicoP_ALC_GetWarpParameter( )	Gets the warp settings.	×	✓

PicoP_ALC_SetWarpState()	Enables or Disables the warp operation	✓	✓
PicoP_ALC_GetWarpState()	Returns the current warp state of output display.	✓	✓
PicoP_ALC _SetOutputVideoState( )	Enables or Disables the output video. When output video is disabled, the display is blanked.	✓	✓
PicoP_ALC _GetOutputVideoState( )	Returns the current state of Output Video.	✓	✓
PicoP_ALC _CorrectKeystone( )	Apply symmetrical keystone correction operation to the output display.	✓	✓
PicoP_ALC_GetKeystoneCorrection ( )	Gets keystone correction settings.	✗	✓
PicoP_ALC _SetPhase( )	Sets the scan line phase delay to align the forward and reverse scan video.	✓	✓
PicoP_ALC _GetPhase( )	Returns the scan line phase delay setting.	✓	✓
PicoP_ALC_AutoSetGreenMagentaBalance()	Performs the auto adjust of the green-magenta balance	✓	✗
PicoP_ALC_GetAutoGreenMagentaBalanceStatus()	Gets the status of the auto green magenta balance command.	✓	✗
PicoP_ALC_SetGreenMagentaBalance()	Performs the manual adjust of the green-magenta balance	✓	✗
PicoP_ALC_GetGreenMagentaBalance()	Gets the green magenta balance offset value.	✓	✗
PicoP_ALC_SetColorAlignment()	Performs vertical or horizontal color alignment for the selected color	✓	✓
PicoP_ALC_GetColorAlignment()	Gets the color alignment offset of the chosen color	✓	✓
PicoP_ALC_SetColorConverter()	Sets the color converter values	✓	✓

PicoP_ALC_GetColorConverter()	Gets the color converter values	✓	✓
PicoP_ALC_SetScanAngle()	Set the scan angle(**NOTE: Only applicable to certain platforms**)	✓	✓
PicoP_ALC_GetScanAngle()	Gets the scan angle(**NOTE: Only applicable to certain platforms**)	✓	✓
PicoP_ALC_SetViewportDistortion( )	Sets viewport distortion parameters.	✗	✓
PicoP_ALC_GetViewportDistortion( )	Gets viewport distortion parameters.	✗	✓
PicoP_ALC_SetOutputVideoStateEx( )	Sets the output video state to be committed to enabled or disabled.	✗	✓
PicoP_ALC_GetOutputVideoStateEx( )	Gets committed Output Video State.	✗	✓
PicoP_ALC_SetSplashScreenTimeout( )	Sets the timeout for displaying splash screen.	✗	✓
PicoP_ALC_GetSplashScreenTimeout( )	Sets the timeout for displaying splash screen.	✗	✓
PicoP_ALC_SetVerticalProjectionAngleOffset( )	Sets the vertical projection angle offset.	✗	✓
PicoP_ALC_GetVerticalProjectionAngleOffset( )	Gets the vertical projection angle offset.	✗	✓

5.4.3. Input Control Functions

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

Function	Description	PicoP Gen1	PicoP Gen2
PicoP_ALC_SetInputCaptureModeInfo( )	Configures custom input video modes to be accepted by PicoP. These new modes augment the video modes already	✓	✓

	supported by the PicoP.		
PicoP_ALC_ModifyInputCaptureModeInfo()	Modifies the video capture mode.	✓	✓
PicoP_ALC _GetInputCaptureModeInfo( )	Gets information about the supported input video modes	✓	✓
PicoP_ALC_SetActiveCaptureMode( )	Sets the input video capture mode to use. Incoming video MUST match the parameters defined by the given mode to be displayed correctly.	✓	✓
PicoP_ALC_GetActiveCaptureMode( )	Returns the currently used input video capture mode.	✓	✓
PicoP_ALC_CommitInputCaptureMode()	Commits the input video capture mode.	✓	✓
PicoP_ALC_GetCommitedInputCaptureMode()	Returns the committed video capture mode.	✓	✓
PicoP_ALC_GetInputVideoProperties( )	Returns detected input video Frame Rate and Lines per Frame.	✓	✓
PicoP_ALC_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_ALC_GetInputVideoState( )	Returns the current state of the input video.	✓	✓

5.4.4. Rendering Functions

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

Function	Description	PicoP Gen1	PicoP Gen2
PicoP_ALC _SetOSDState( )	Enables or Disables the On-Screen	✓	✓

	Display (OSD)		
PicoP_ALC _GetOSDState( )	Returns the On-Screen Display (OSD) state.	✓	✓
PicoP_ALC _SetOSDInfo( )	Set the Size and Position of the On-Screen Display (OSD) within the output video area.	✓	✓
PicoP_ALC _GetOSDInfo( )	Returns the current Size and Position of the On-Screen Display.	✓	✓
PicoP_ALC _GetDisplayInfo( )	Returns the current size of the FrameBuffer used for rendering.	✓	✓
PicoP_ALC _SetActiveFrameBuffer( )	Set the frame buffer to be used for video output.	✓	✓
PicoP_ALC _GetActiveFrameBuffer( )	Returns the currently active frame buffer used for video output.	✓	✓
PicoP_ALC_SetActiveOSD()	Sets the active OSD to be used for video output	✓	✓
PicoP_ALC_GetActiveOSD()	Gets the active OSD used for video output	✓	✓
PicoP_ALC _LoadBitmapImage( )	Loads a bitmap image into the OSD or FrameBuffer.	✓	✓
PicoP_ALC _DrawTestPattern( )	Displays one of the built-in test patterns.	✓	✓
PicoP_ALC _DrawText( )	Queues a command to displays Text in the OSD or FrameBuffer	✓	✓
PicoP_ALC _GetTextBoxInfo( )	Returns the dimensions of the bounds of the rectangular region that will be filled with the given text (but not actually drawn). Provides the API user with feedback to determine where to draw text.	✓	✓
PicoP_ALC _DrawPoint( )	Queues a draw command to set a single pixel in the OSD or FrameBuffer.	✓	✓
PicoP_ALC _DrawLine( )	Queues a draw command to draw a line segment between two points.	✓	✓
PicoP_ALC _DrawTriangle( )	Queues a draw command to draw and fill a triangle bounded by 3 points.	✓	✓
PicoP_ALC _DrawRectangle( )	Queues a draw command to draw and fill a rectangle into the OSD or FrameBuffer.	✓	✓
PicoP_ALC _Render( )	Renders queued draw commands into a Render Target. Note that rendering into a framebuffer that is actively capturing input video will result in the rendered pixels to be overwritten by input video.	✓	✓
PicoP_ALC_ClearTarget()	Clears the selected render target.	✓	✓

5.4.5. System Management Functions

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

Function	Description	PicoP Gen1	PicoP Gen2
PicoP_ALC_GetSystemStatus( )	Retrieves the system status.	✓	✓
PicoP_ALC_GetSystemInfo( )	Retrieves system information.	✓	✓
PicoP_ALC_GetEventLog( )	Retrieves the system event log.	✓	✓
PicoP_ALC_RestoreFactoryConfig( )	Restores System Settings to Factory Configuration.	✓	✓
PicoP_ALC_SetSystemPowerState( )	Sets the System Power State (Standby, Off).	✓	✓
PicoP_ALC_GetSystemPowerState( )	Returns the current system Power State.	✓	✓
PicoP_ALC_SaveSplashScreen( )	Takes a snapshot of the specified frame buffer and saves the content as the Splash Screen.	✓	✓
PicoP_ALC_UpgradeSoftware( )	Upgrades the embedded Software.	✓	✓
PicoP_ALC_UpgradeSoftware_Ex( )	Upgrades the embedded Software and gives progress status in a callback function.	✓	✓
PicoP_ALC_UpgradeFpga( )	Loads upgrade of the embedded FPGA Firmware (**NOTE: Only applicable to certain platforms**)	×	×
PicoP_ALC_GetBatteryStatus( )	Return the charge state of the battery (**NOTE: Only applicable to certain platforms**)	×	×

5.5. Step 4: Exit Application

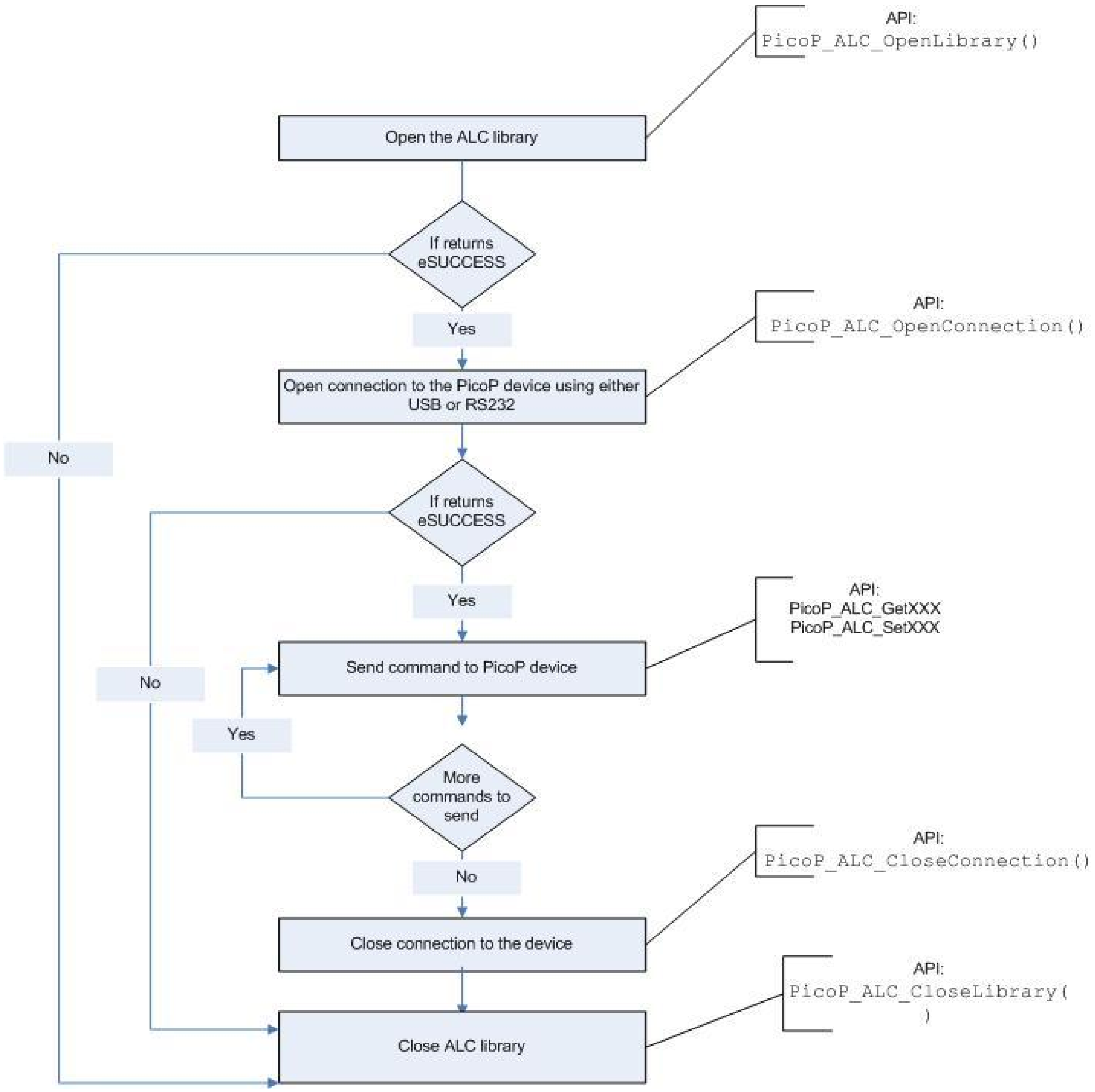
To gracefully exit the host application, call the `PicoP_ALC_CloseConnection()` and `PicoP_ALC_CloseLibrary()` functions to shut down the connection to PicoP and to release all resources used by the library.

6. “Hello, World“ - Program

The below listing shows a simple “Hello, World” sample program with a PicoP connected to the USB port:

```
int main(int c, char argv[])
{
      PicoP_HANDLE libHandle;
      PICOP_RC ret = eSUCCESS;

      /* Open the ALC library */
      ret = PicoP_ALC_OpenLibrary(&libHandle);
      if (eSUCCESS != ret)
      {
            printf(“Error opening ALC library\n”);
            return -1;
      }

      PicoP_HANDLE connectionHandle;
      PicoP_ConnectionInfo connectionInfo;
      connectionInfo.connectionType = eUSB;
      connectionInfo.usbInfo.productID = 0x0004;
      connectionInfo.usbInfo.serialNumber = “xxxxxx”;

      ret  = PicoP_ALC_OpenConnection(libHandle,
                                      connectionInfo,
                                      &connectionHandle);
      if (eSUCCESS != ret)
      {
            printf(“Error opening USB connection\n”);
            return -1;
      }

      /* Get brightness val */
      FP32 birghtnessVal;
      ret = PicoP_ALC_GetBrightnessVal(connectionHandle,
                                     &brightnessVal, 0); //get current val
      /* Disable Input Video */
      ret = PicoP_ALC_SetInputVideoState(connectionHandle,
                                           eINPUT_VIDEO_DISABLED);
      /* Draw Text */
      PicoP_Point startPoint = {430, 240};   /* ~ Middle of display */
      PicoP_Color color = {255, 255, 255, 0};  /* White */
      PicoP_Color bgColor = {0, 0, 255, 0};  /* Blue */

      ret = PicoP_ALC_DrawText(connectionHandle, 0,
                               (char *)”Hello, World from PicoP”,
                               23, startPoint, color , bgColor);
      ret = PicoP_ALC_Render( connectionHandle);

      /* Close connection and ALC library */
      PicoP_ALC_CloseConnection(connectionHandle);
      PicoP_ALC_CloseLibrary(libHandle);
}
```

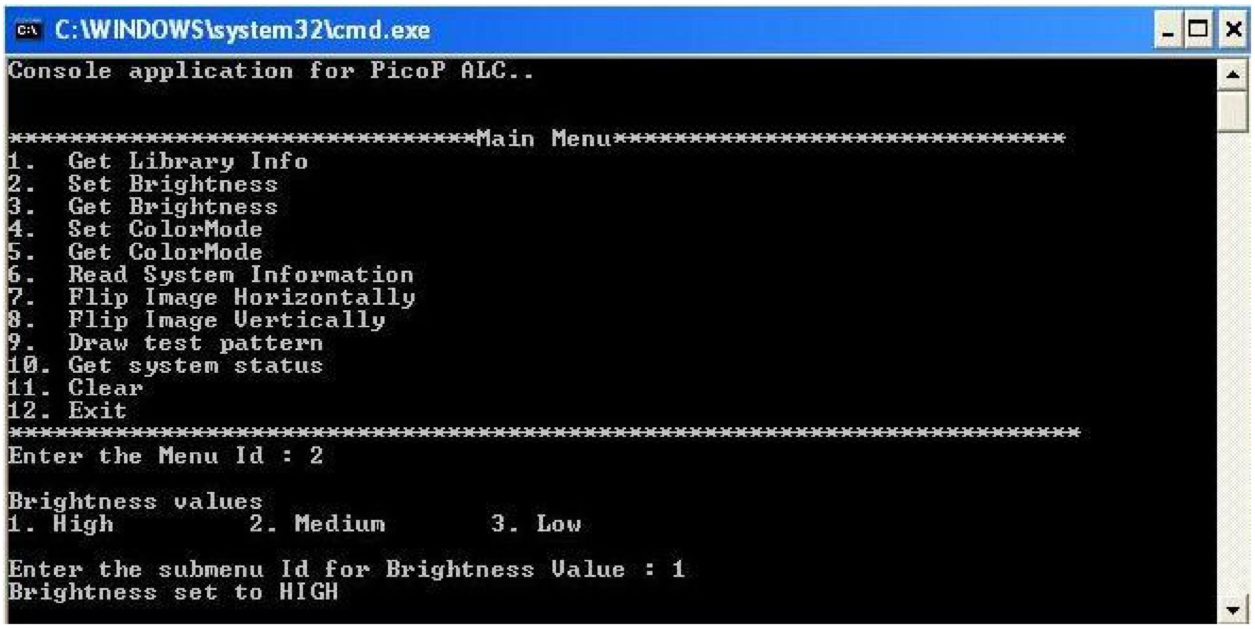
6.1. Recommended Program Flow



7. The Console Demo Project

The Console Demo project is a simple console application developed in C. It demonstrates the use of SDK interfaces to communicate with the PicoP device and to control basic Projector functions. The application can control, e.g., the following PicoP operations:

- Get Library Info
- Change Brightness
- Change Color Mode
- Read System Info
- Display Test Pattern
- Flip Image Horizontally
- Flip Image Vertically



```
Console application for PicoP ALC..

*******************************Main Menu******************************
1.  Get Library Info
2.  Set Brightness
3.  Get Brightness
4.  Set ColorMode
5.  Get ColorMode
6.  Read System Information
7.  Flip Image Horizontally
8.  Flip Image Vertically
9.  Draw test pattern
10. Get system status
11. Clear
12. Exit
***********************************************************************
Enter the Menu Id : 2

Brightness values
1. High         2. Medium       3. Low

Enter the submenu Id for Brightness Value : 1
Brightness set to HIGH
```

Console Demo Application – Sets brightness value to HIGH

7.1. *Prerequisites*

The System Requirements for running the Console Demo project are:

- Pentium class or faster CPU with at least 512 MB of RAM.
- Windows 32-bit operating system or Windows 64-bit operating system.: Windows XP, Vista, or Windows 7
- Microsoft Visual Studio 2010(Ultimate) or later.

  Note: For compiling 64 bit applications, Microsoft Visual Studio with 64 bit compiler is required.

7.2. Running the Console Demo

To build and run the Console Demo, please follow the below steps:

1. **Open the Solution/Project file**
    - Navigate to 'PicoP_Windows_SDK\samples\PicoP_ALC_Console' and double click on the 'PicoP_ALC_Console.sln' file.
    - This solution is created using VS2010(Ultimate). If you are using a newer version of Visual studio, conversion wizard will assist you to convert it to the newer version.

2. **Check Dependencies**
    - This solution assumes SDK header files are available at relative path 'PicoP_Windows_SDK\Interface' and library from 'PicoP_Windows_SDK\Lib'.
    - If you are running the Console demo from a different path, update following project settings.
        - SDK header file path at : Project→Properties →C/C++ →Additional Include directories
        - SDK lib path at : Project→Properties →Linker→Additional Dependencies
    - If you are using a 64 bit platform, do the following.
        - Change the solution platform type from Win32 to x64.
        - Change the library path from 'PicoP_Windows_SDK\Lib' to 'PicoP_Windows_SDK\Lib_64' by updating the following project settings
            - SDK lib path at : Project→Properties →Linker→Additional Dependencies

3. **Build and Run the application**
    - For running the application the PicoP_ALC_Api.dll is to be copied where the executable 'PicoP_ALC_Console.exe 'is present.

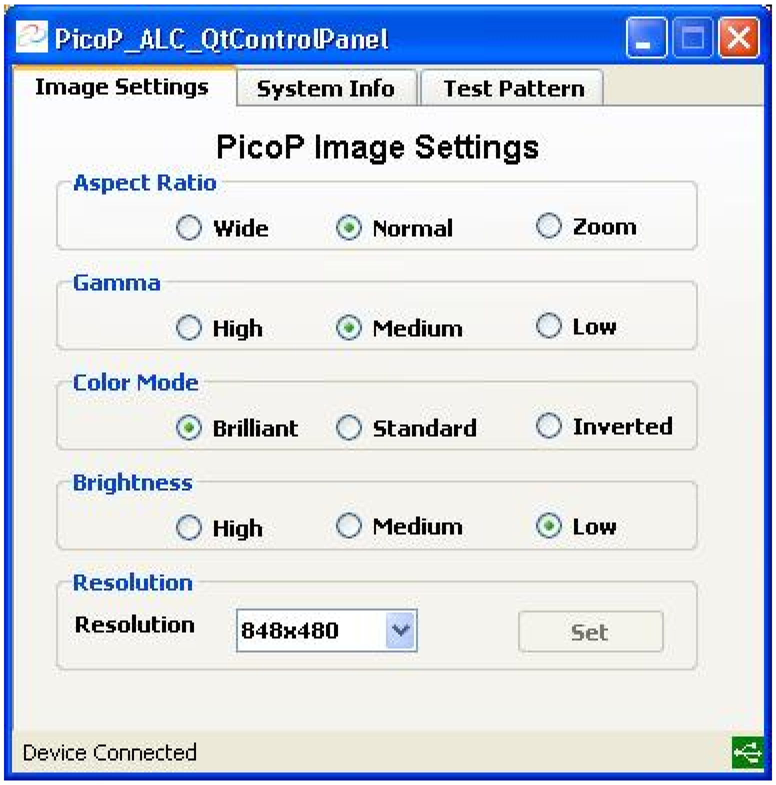
8. The Qt Demo Projects

The Qt Demo applications are more advanced Graphical User Interface (GUI) based host application developed with Qt. These applications connect to the PicoP device through USB port.

8.1. *PicoP_ALC_QtControlPanel*

This application demonstrates how to perform the following PicoP Operations from a GUI

- Get Library Info
- Read System Info
- Change Brightness
- Change Aspect Ratio
- Change Color Mode
- Set Gamma
- Flip Image Horizontally
- Flip Image Vertically
- Display Test Pattern
- Set Display Resolution to match built-in capture modes
- Perform Phase adjustment



PicoP Image Settings

Aspect Ratio: Wide [✓ Normal] Zoom

Gamma: High [✓ Medium] Low

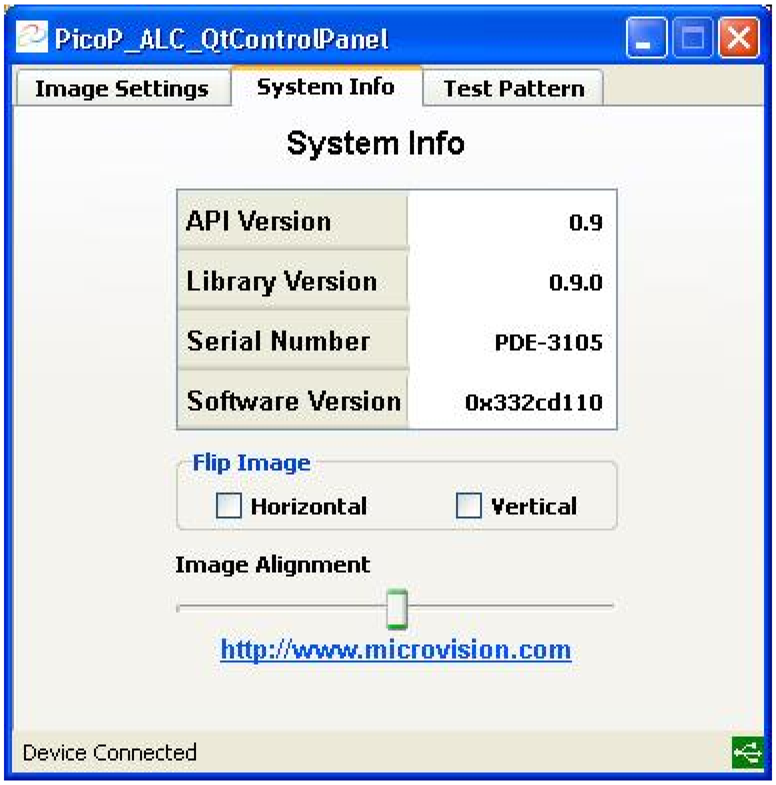
Color Mode: [✓ Brilliant] Standard Inverted

Brightness: High Medium [✓ Low]

Resolution: Resolution 848x480

Device Connected

Qt ControlPanel Application: Image Settings tab



System Info

API Version	0.9
Library Version	0.9.0
Serial Number	PDE-3105
Software Version	0x332cd110

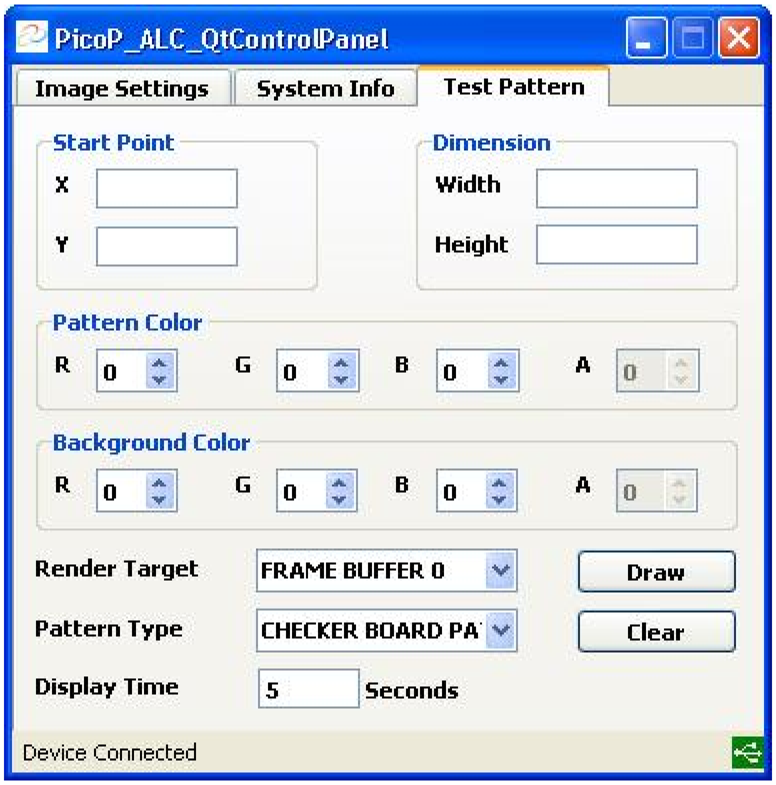
Flip Image
[ ] Horizontal [ ] Vertical

Image Alignment

http://www.microvision.com

Device Connected

Qt ControlPanel Application: System Status tab



Start Point
X
Y

Dimension
Width
Height

Pattern Color
R 0 G 0 B 0 A 0

Background Color
R 0 G 0 B 0 A 0

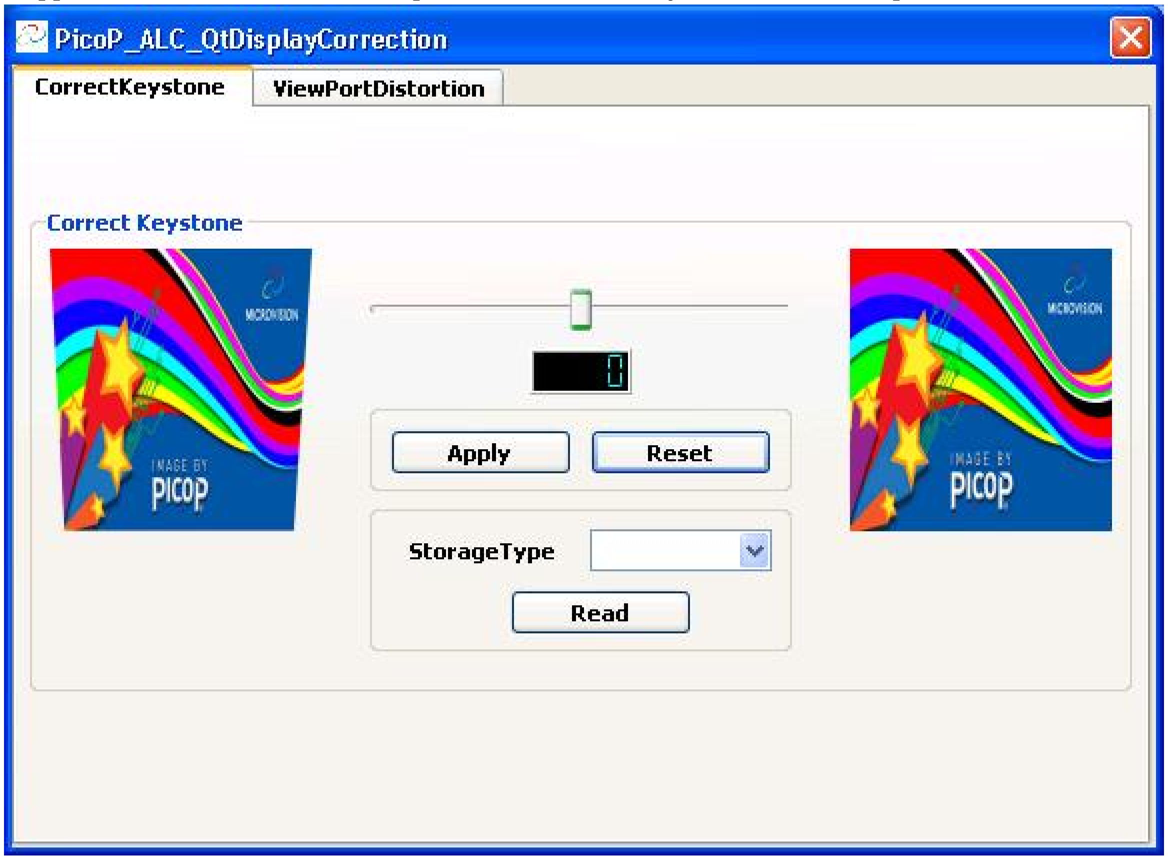
Render Target: FRAME BUFFER 0
Pattern Type: CHECKER BOARD PA
Display Time: 5 Seconds

Device Connected

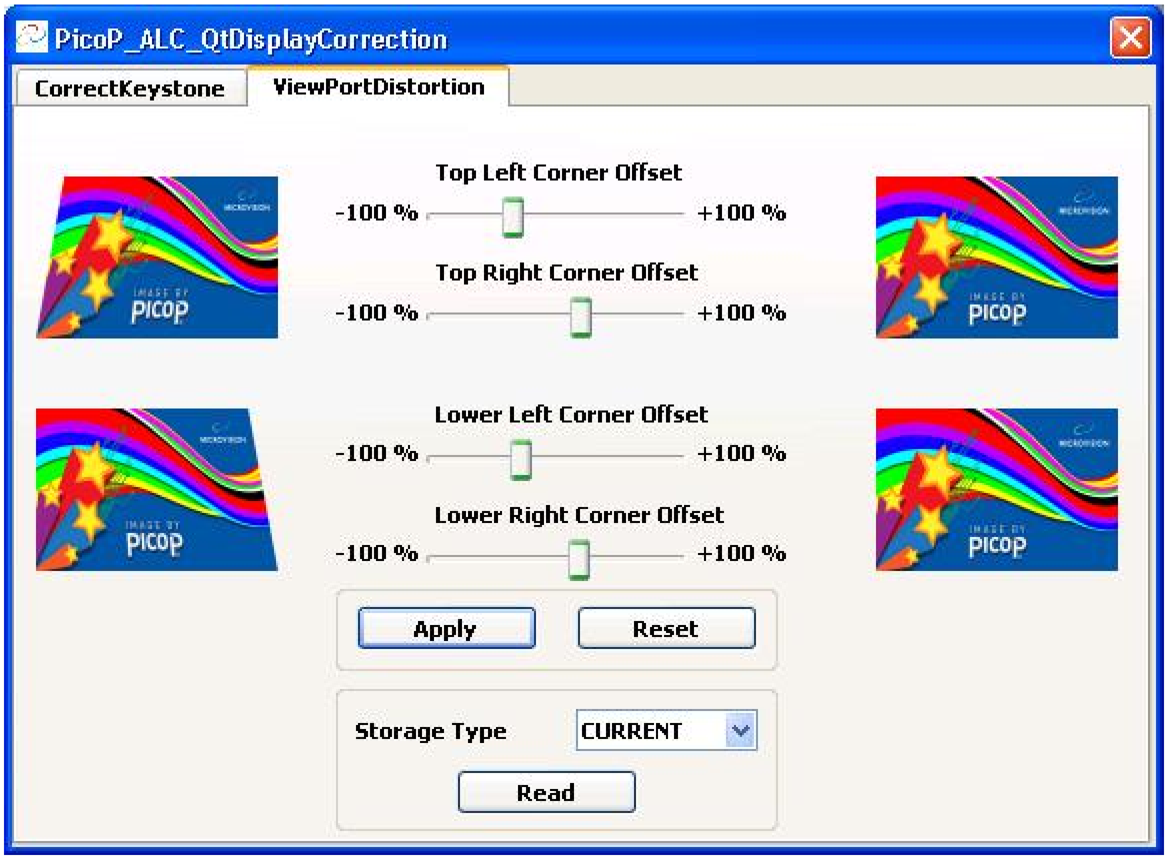
Qt ControlPanel Application: Test pattern tab

8.2. PicoP_ALC_QtDisplayCorrection

This application demonstrates how to perform the PicoP Keystone correction operations from a GUI.



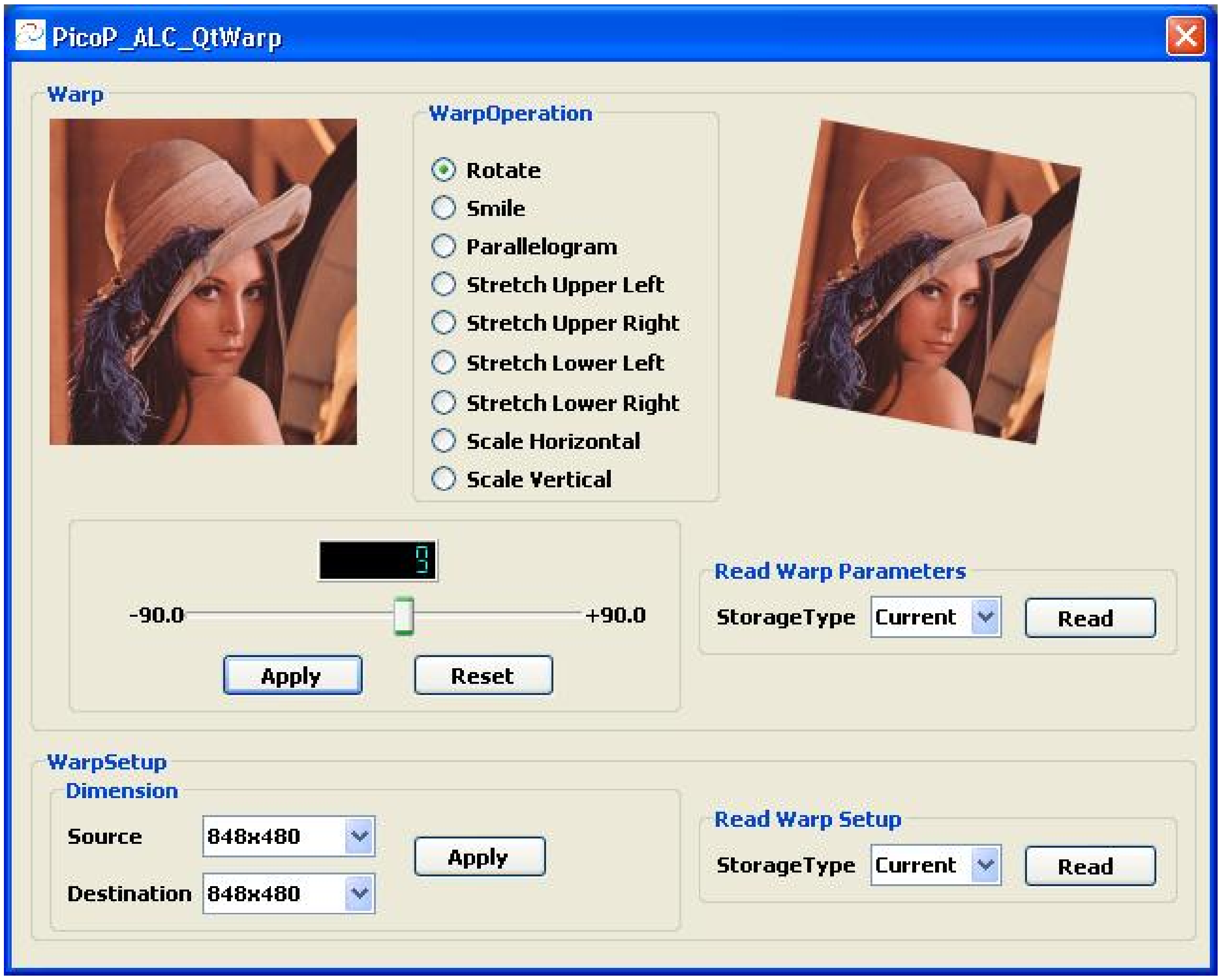
Qt DisplayCorrection Application: Correct Keystone tab



Qt DisplayCorrection Application: ViewPort Distortion tab

8.3. PicoP_ALC_QtWarp

This application demonstrates how to perform the WarpImage operations from a GUI.



WarpOperation: [✓ Rotate]

Slider value: 9

StorageType: Current

Source: 848x480

Destination: 848x480

StorageType: Current

Qt Warp Application

Note:
The warp functions only works for external video. It does not work for splash screen or test patterns.

8.4. *Prerequisites*

The System Requirements for running the Qt Demo project are:

- Pentium class or faster CPU with at least 512 MB of RAM.
- Windows 32-bit operating system or Windows 64-bit operating system: Windows XP, Vista, or Windows 7
- Qt 4.7.0 or later. Qt can be installed from http://qt.nokia.com/downloads
- QtCreator 2.0.0
- Qt Visual Studio Add-in to build applications in 64 bit operating system.

8.5. *Building and running Qt Demo Applications*

To build and run Qt Demo applications in Windows 32 bit operating system

1)Using QtCreator

To build and run Qt Demo applications using the IDE QtCreator v. 2.0.0, please follow the below steps.

1. Open QtCreator.
2. Browse and open the .pro file using File->Open option in the QtCreator IDE.
3. Set the build directory path for the project using the option Projects->Build Settings->Build directory.
4. Create Makefile using Build->Run qmake option in the QtCreator IDE.
5. Build the project using Build->Build Project option in the QtCreator IDE.
6. Run the application using Build->Run option in the QtCreator IDE.

To build and run Qt Demo applications in Windows 64 bit operating system

1)Using QtCreator

The Qt demo applications can also be build using QtCreator in windows 64 bit OS by just following all the above steps (mentioned for 32 bit) and by changing the library path to ../Lib_64/

2) Using Qt Visual Studio Add-in

To build and run Qt Demo applications in Windows 64 bit operating system, use Qt Visual Studio Add-in.

1. Open Microsoft Visual Studio 2005.
2. Browse and open the .pro file by selecting the menu Qt-> Open Qt Project File(.pro) in the Visual Studio IDE.
3. Check whether the platform type is x64 in the Solution Platform combo box. If not select the same.
4. Set the library path to ../Lib_64/ by updating the following project settings.
   SDK lib path at : Project→Properties →Linker→Input→Additional Dependencies
5. Build and run the application.

Note:

- This solution assumes SDK header files are available at relative path ‘PicoP_Windows_SDK\Interface’ and library from ‘PicoP_Windows_SDK\Lib’. If you are running the Qt demo from a different path, update following project settings in the .pro file
    - SDK header file path at : ALC_LIB_PATH
    - SDK lib path at : INCLUDEPATH
- Copy the dll ‘PicoP_Windows_SDK\Lib\ PicoP_ALC_Api.dll’ to the path where the executable is being created.