# PicoP® Gen4 Programmer's Guide

MVIS #: DA0140028

May 2019 Revision A.2


## ABSTRACT

This Programmer's Guide describes the functionality of MicroVision's 4th Generation PicoP® Scanning Engines (PSEs) Application Programming Interface (API).

# Contents

# Chapter 1: Introduction

This Programmer's Guide details the functionality of the command and control interface of MicroVision's 4th Generation PicoP® Scanning Engines (PSEs). The MicroVision PicoP Scanning Engine (PSE) is targeted for customers embedding a small, High Definition laser projector and/or LiDAR 3D depth sensor into their product. The PSE is delivered with PicoP® Software Development Kits (SDKs), which facilitate third-party Application Software development for the PicoP Scanning Engine.

The PicoP SDK packages consists of the following:

- Operating System specific software libraries for interfacing with the PSE
- Application Programming Interfaces (APIs)
- Sample Applications
- Documentation

The PicoP SDKs support 2 API types:

- C Function API for the Windows and Linux operating systems
- Java Class API for Android ™ operating system

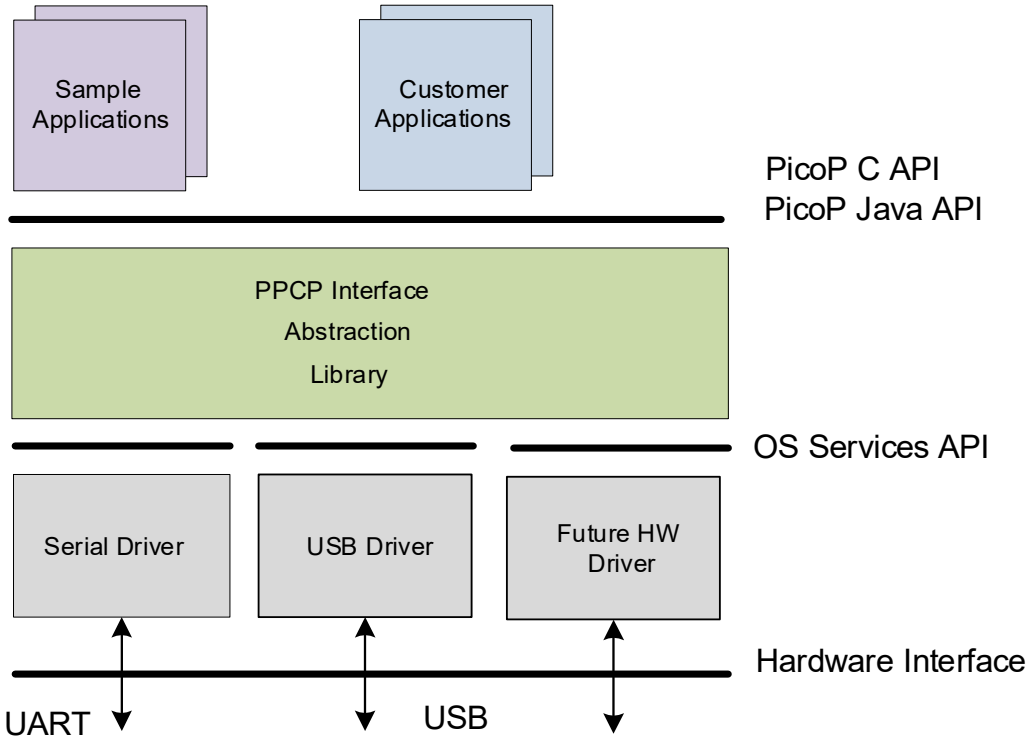
Other languages in the Windows and Linux operating systems, such as C# or Python can be supported through wrappers or controls built on top of the C Function API.

This Guide defines all applicable functions and commands supported by the PSE Application Programming Interface (API). The supported functions are described in programming language independent pseudo-code. For language specific information, please refer to the appropriate PSE Software Development Kit (SDK) documentation.

The information contained in this document is provided for guidance purposes only and is subject to change without notice.

# Chapter 2: Communication Interface

The host device can communicate with the PSE either through USB or UART physical interface. Over these interfaces the host device communicates with PSE module using MicroVision's proprietary PicoP Command Protocol (PPCP). These low level PPCP commands are abstracted by the PicoP Software Development Kit (SDK) Application Programming Interface (API). The PPCP packets are generated by the SDK Libraries as requested by the application SW thru API function/method calls.




PPCP Communication with PicoP Device

# Chapter 3: API Functionality Overview

The PicoP API is split in to the following major functional categories, which are described in detail in following chapters 4-10:

**Chapter 4:** Connection & Library Management: Connect to PicoP over USB or Serial interface.

**Chapter 5:** Display Control: Configure and control the PicoP Display.

**Chapter 6:** System Management: Manage the PicoP System, Firmware upgrades, Event Log, etc.

**Chapter 7:** Input Control: Configure the PicoP Input Video.

**Chapter 8:** 3D Sensing: Configure and control the PicoP 3D Sensing function.

**Chapter 9:** Rendering: Display test patterns by rendering them into the Framebuffer.

# Chapter 4: Connection & Library Management Functions

The connection & library management functions are used to connect the application to the PSE and to configure the communications interface.

## 4.1. PicoP_OpenLibrary ( libraryHandle)

Opens the interface library and allocates resources necessary for operation. Returns a handle to the library to be used for opening a connection to the PSE. The application must call the PicoP_OpenLibrary() function first to obtain a libraryHandle that can then be used to open a communications connection.

| Parameter | Description |
|---|---|
| **libraryHandle [OUT]** | **A handle to the opened SDK Library** |

## 4.2. PicoP_GetLibraryInfo ( libraryHandle, libraryInfo)

This function is used to retrieve the version and capability information of the SDK Library.

| Parameter | Description |
|---|---|
| **libraryHandle [IN]** | **Handle to the SDK Library** |
| **libraryInfo [OUT]** | **Library info structure containing the following info on the SDK library:**<br>• **Major version**<br>• **Minor version**<br>• **Patch Version**<br>• **Capability flags** |

## 4.3. PicoP_CloseLibrary ( libraryHandle)

Closes the library, releases all resources and closes all open connections. This function should be called last upon exiting the application.

| Parameter | Description |
|---|---|
| libraryHandle [IN] | The handle to the SDK Library to be closed |

## 4.4. PicoP_OpenConnection ( libraryHandle, connectionInfo, connectionHandle)

Opens a connection to the PSE using either USB or UART interface and returns a connectionHandle that must be used by subsequent API calls.

| Parameter | Description |
|---|---|
| libraryHandle [IN] | Handle to the SDK Library |
| connectionInfo [IN] | Connection type and parameters for the specific connection type, such as port #, baud rate, etc. |
| connectionHandle [OUT] | A handle to the opened connection to be used with subsequent function calls |

## 4.5. PicoP_OpenConnectionUart (libraryHandle, connectionInfo, connectionHandle)

Opens a connection to the PSE using UART interface and returns a connection Handle that must be used by subsequent API calls to indicate the connection to be used.

| Parameter | Description |
|---|---|
| libraryHandle [IN] | Handle to the SDK Library |

| | |
|---|---|
| **connectionInfo [IN]** | **Information about UART the connection such as baud rate, parity, etc.** |
| **connectionHandle [OUT]** | **A handle to the opened connection to be used with subsequent function calls.** |

## 4.6. PicoP_CloseConnection ( connectionHandle)

Closes a previously opened connection to PSE.

| Parameter | Description |
|---|---|
| **connectionHandle [in]** | **A handle to the opened connection to be used with subsequent function calls** |

# Chapter 5: Display Control Functions

The Display Control Functions enable the host application to control and configure the output projection display.

## 5.1. PicoP_SetBrightness ( brightnessValue, commit)

Set the brightness for the output projection display.

| Parameter | Description |
|---|---|
| **brightnessValue [IN]** | **Brightness level to set. Range 0 - 100. 0 means display off, 100 indicates max brightness.** |
| **commit [IN]** | **FALSE - Do not persist (value restored back after power cycle)**<br>**TRUE - Persist setting** |

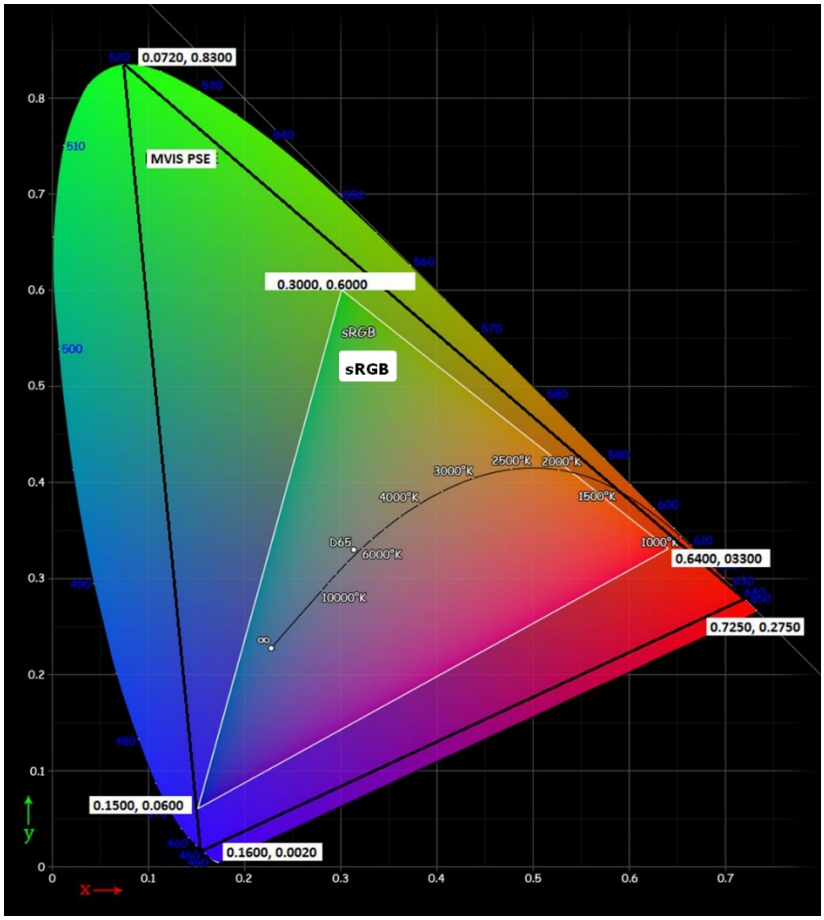
## 5.2. PicoP_GetBrightness ( brightnessValue, storageType)

Retrieve the current PSE display brightness.

| Parameter | Description |
|---|---|
| **brightnessValue [OUT]** | **Floating point value from 0.0 to 1.0** |
| **storageType [IN]** | **eCURRENT_VALUE - Current value in use**<br>**eVALUE_ON_STARTUP - Value persisted for PSE power on**<br>**eFACTORY_VALUE - Factory default** |

## 5.3. PicoP_SetColorMode ( colorMode, commit)

The PSE projection display utilizes Red (640 nm), Green (532 nm), Blue (450 nm) laser diodes as light sources to illuminate the scanned beam projection display. The laser diodes enable the PSE to provide a wide gamut of brilliant and reproducible colors. This expanded color gamut is significantly larger than standard color gamuts, such as sRBG (see the CIE chromaticity diagram below).




This function sets the PSE color mode.

| Parameter | Description |
|---|---|
| **colorMode [IN]** | **BRILLIANT - PicoP wide gamut color space**<br>**STANDARD - Standard color space, close to sRGB or similar**<br>**RICH - Color Space between BRILLIAND and STANDARD**<br>**INVERTED - inverted color space** |
| **commit [IN]** | **FALSE - Do not persist (value restored back after power cycle)**<br>**TRUE - Persist setting** |

## 5.4. PicoP_GetColorMode ( colorMode, storageType)

This function retrieves the current color mode setting from the PSE.

| Parameter | Description |
|---|---|
| colorMode [OUT] | BRILLIANT - PicoP wide gamut color space<br>STANDARD - Standard color space, close to sRGB or similar<br>RICH - Color Space between BRILLIAND and STANDARD<br>INVERTED - inverted color space |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

## 5.5. PicoP_SetGamma ( color, gammaValue, commit)

Gamma is the relationship between the numerical value of a pixel in an image file and the brightness of the projected pixel when viewed on screen. This relationship is non-linear, meaning that a change in pixel value does not translate into an equivalent change in brightness. For almost all TVs and computer monitors, a change in pixel value results in a change in brightness raised to the 2.2 power. The gamma for these devices, therefore, is said to be 2.2. The PicoP_SetGamma() functions allows adjustment of the gamma value.

Default Display

Display with increased Gamma

| Parameter | Description |
|---|---|

| | |
|---|---|
| color [IN] | RED - Set the gamma of the red laser<br>GREEN - Set the gamma of the green laser<br>BLUE - Set the gamma of the blue laser |
| gammaValue [IN] | A floating point value for gamma, range 1.0-3.0 . |
| commit [IN] | FALSE - Do not persist (value restored back after power cycle)<br>TRUE - Persist setting |

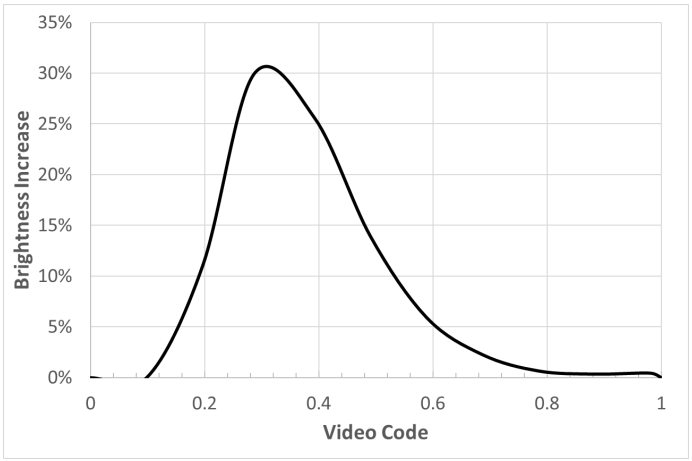
## 5.6. PicoP_GetGamma ( color, gammaValue, storageType)

This function retrieves the current display gamma value.

| Parameter | Description |
|---|---|
| color [IN] | RED - Get the gamma of the red laser<br>GREEN - Get the gamma of the green laser<br>BLUE - Get the gamma of the blue laser |
| gammaValue [OUT] | Current gamma value |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

## 5.7. PicoP_SetVideoGammaBoostMode ( color, boostMode, commit)

The PSE supports an Enhanced Brightness mode which is achieved by boosting the brightness of video codes around code level 0.3 (or 76 of 255), as illustrated in figure below. These video code values are the most common in typical pictures and movie frames. When the Gamma Boost mode is enabled, the brightness of these video codes is increased up to 30% to correspond to a ~30% brighter display when typical pictures or movies.




The boost can be performed to individual primary colors.

| Parameter | Description |
|---|---|
| color [IN] | RED - Set boost mode for the red laser<br>GREEN - Set boost mode for the green laser<br>BLUE - Set boost mode for the blue laser<br>ALL - Set boost mode for all (red, blue and green) lasers |
| boostMode [IN] | NONE - Standard mode, no brightness boost<br>VIRTUAL_LUMENS_30PCT - Boost mode, enhanced lumens boosted 30% |
| commit [IN] | FALSE - Do not persist (value restored back after power cycle)<br>TRUE - Persist setting |

## 5.8. PicoP_GetVideoGammaBoostMode ( color, boostMode, storageType)

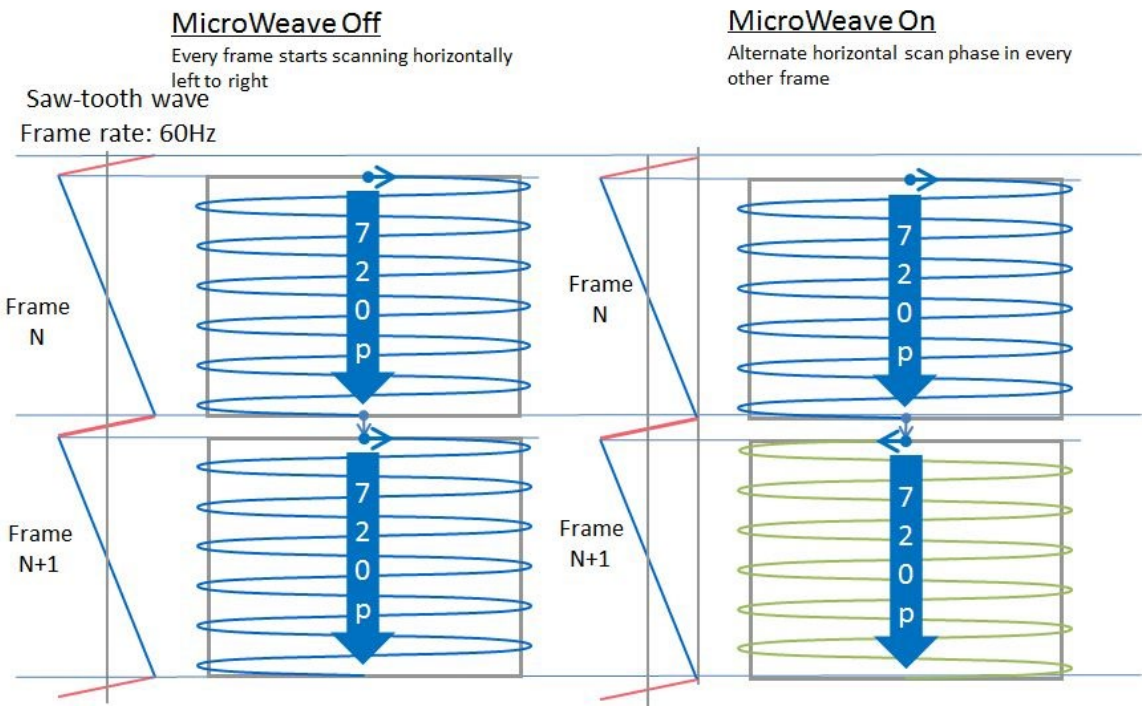
This function returns the gamma boost mode.

| Parameter | Description |
|---|---|

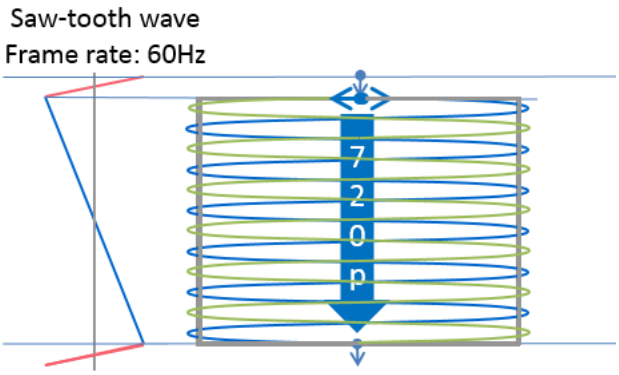
| | |
|---|---|
| color [IN] | RED - Boost mode of the red laser<br>GREEN - Boost mode of the green laser<br>BLUE - Boost mode of the blue laser |
| boostMode [OUT] | NONE - Standard mode, no brightness boost<br>VIRTUAL_LUMENS_30PCT - Boost mode, enhanced lumens boosted 30% |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

## 5.9. PicoP_SetMicroWeaveMode ( microWeaveMode, commit)

PSE supports a “MicroWeave” mode, in which successive frames are displayed with horizontal scan starting Left->Right, then Right->Left, then Left->Right, etc., as illustrated below:




The MicroWeave mode fills in gaps between MEMS raster lines, as shown below:




| Parameter | Description |
|---|---|
| microWeaveMode [IN] | OFF - MicroWeave Off<br>ON - MicroWeave On |
| commit [IN] | FALSE - Do not persist (value restored back after power cycle)<br>TRUE - Persist setting |

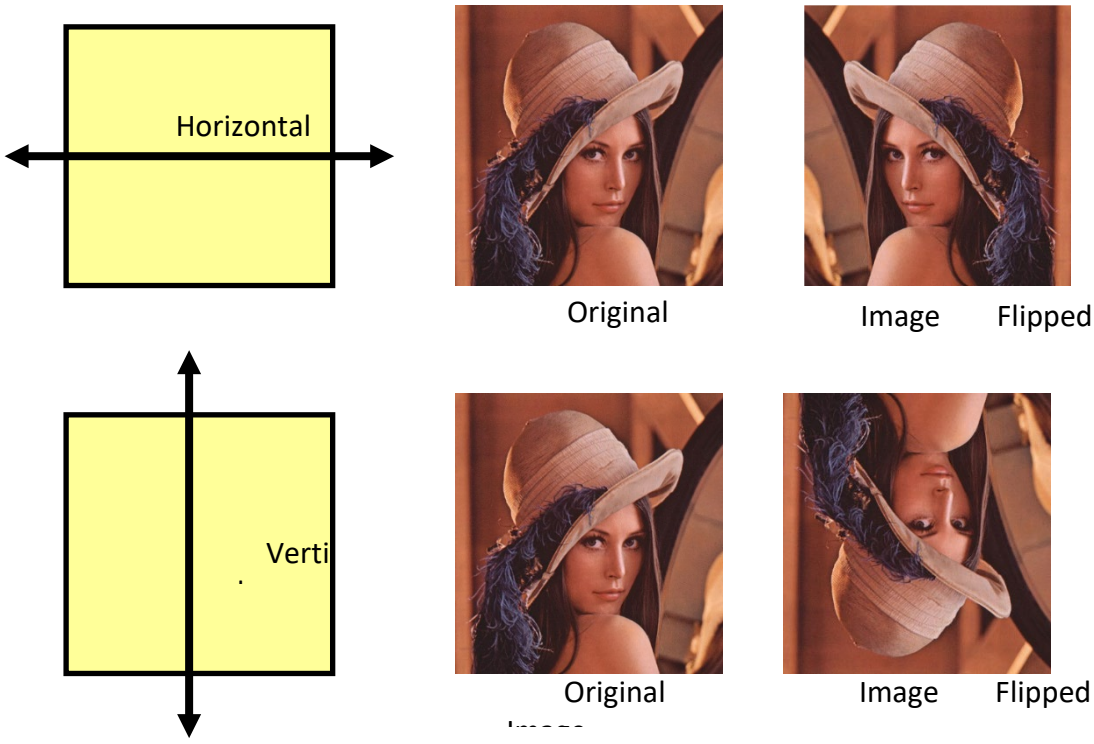
## 5.10. PicoP_GetMicroWeaveMode ( microWeaveMode, storageType)

Return system's MicroWeave mode.

| Parameter | Description |
|---|---|
| microWeaveMode [OUT] | OFF - MicroWeave Off<br>ON - MicroWeave On |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

## 5.11. PicoP_SetFlipState( flipState, commit)

This function allows the host to flip the display horizontally, vertically, or both horizontally and vertically. The effect of the display flip is illustrated below:




| Parameter | Description |
|---|---|
| **flipState[IN]** | **Flip Direction:**<br>**FLIP_NEITHER - Normal display, not flipped**<br>**FLIP_HORIZONTAL - Display flipped across the display axis**<br>**FLIP_VERTICAL - Display flipped across the display vertical axis**<br>**FLIP_BOTH - Display flipped across both the horizontal axis and display axis** |
| **commit [IN]** | **FALSE - Do not persist (value restored back after power cycle)**<br>**TRUE - Persist setting** |

## 5.12. PicoP_GetFlipState( flipState, storageType)

Returns the flip state of the display

| Parameter | Description |
|---|---|
| flipState [OUT] | Flip Direction:<br>FLIP_NEITHER - Normal display, not flipped<br>FLIP_HORIZONTAL - Display flipped across the display axis<br>FLIP_VERTICAL - Display flipped across the display vertical axis<br>FLIP_BOTH - Display flipped across both the horizontal axis and display axis |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

## 5.13. PicoP_SetOutputVideoState ( state, commit)

This function enables or disables the output video. When the output video is disabled, the display is blanked.

| Parameter | Description |
|---|---|
| state [IN] | ENABLED - Enable the output video<br>DISABLED - Disable the output video and blank the display |
| commit [IN] | FALSE - Do not persist (value restored back after power cycle)<br>TRUE - Persist setting |

## 5.14. PicoP_GetOutputVideoState ( state, storageType)

Returns the current state of the output video.

| Parameter | Description |
|---|---|
| state [OUT] | ENABLED - The output video is enabled |

| | |
|---|---|
| | DISABLED - The output video is disabled |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

## 5.15. PicoP_GetOutputVideoProperties( horizontalResolution, verticalResolution, frameRate)

Return the current display output properties, such as display resolution and frame rate.

| Parameter | Description |
|---|---|
| horizontalResolution [OUT] | Current output display horizontal resolution in number of horizontal pixels |
| verticalResolution [OUT] | Current output display vertical resolution in number of vertical pixels |
| frameRate [OUT] | Current output display frame rate in frames per second |

## 5.16. PicoP_SetBiPhase ( phaseValue, commit)

The PicoP Laser Beam Scanning display illuminates pixels during both horizonal left to right and right to left scans. The PSE automatically adjusts the phase delay so that left to right and right to left scanned pixels are perfectly aligned in the vertical direction. However, in case one can observe a misalignment (display appears out of focus), this function allows the host to manipulate the phase delay. This manual override should not be needed in normal operation.

The effect of the Phase setting is illustrated below:

Original Image

Phase Offset = 30

| Parameter | Description |
|---|---|
| phaseValue [IN] | Value for the horizontal phase offset (Range -50 … +50) |
| commit [IN] | FALSE - Do not persist (value restored back after power cycle)<br>TRUE - Persist setting |

## 5.17. PicoP_GetBiPhase ( phaseValue, storageType)

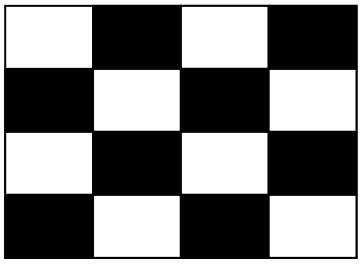
This function retrieves the scan line phase delay value.

| Parameter | Description |
|---|---|
| phaseValue [OUT] | Value for the horizontal phase offset (Range -50 … +50) |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

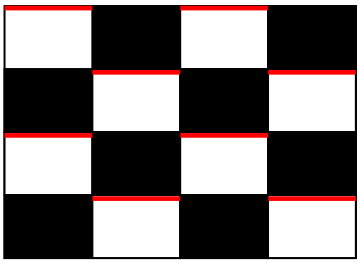
## 5.18. PicoP_SetColorAlignment ( direction, laser, offset, commit )

This function performs vertical or horizontal color alignment for the selected laser/color component. The lasers have been carefully aligned in the factory, but this function allows re-alignment if the colors are no longer perfectly overlapping. The use of this function should not be needed in normal operation.

The effect of the color alignment setting is illustrated below:



Original Display, all colors in alignment



Manual color alignment direction = VERTICAL, Laser = RED_1, RED_2, offset = 32

| Parameter | Description |
|---|---|
| direction [IN] | HORIZONTAL - perform color alignment in Horizontal direction<br>VERTICAL - perform color alignment in Vertical direction |
| laser [IN] | RED_1 - Align Red Laser #1<br>GREEN_1 - Align Green Laser #1<br>BLUE_1 - Align Blue Laser #1<br>RED_2 - Align Red Laser #2<br>GREEN_2 - Align Green Laser #2 |
| offset [IN] | Value for the alignment (Range -32 … +32) |
| commit [IN] | FALSE - Do not persist (value restored back after power cycle)<br>TRUE - Persist setting |

## 5.19. PicoP_GetColorAlignment ( direction, laser, offset, storageType )

This function returns the alignment offset in vertical or horizontal direction for a specified laser/color component.

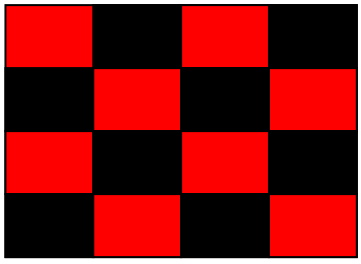
| Parameter | Description |
|---|---|

| | |
|---|---|
| direction [IN] | HORIZONTAL - color alignment in Horizontal direction<br>VERTICAL - color alignment in Vertical direction |
| laser [IN] | RED_1 - Align Red Laser #1<br>GREEN_1 - Align Green Laser #1<br>BLUE_1 - Align Blue Laser #1<br>RED_2 - Align Red Laser #2<br>GREEN_2 - Align Green Laser #2 |
| offset [OUT] | Value for the alignment (Range -32 … +32) |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

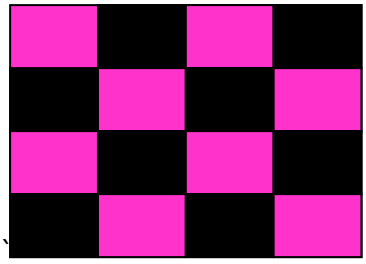
## 5.20. PicoP_SetColorConverter ( color, coefficient, commit)

The PSE Color converter is a 3x3 matrix that allows the display colors to be remapped. This function sets the color converter value for a specific color pair in the 3x3 color converter matrix.

The effect of the color converter setting is illustrated below on a checkerboard display when adding RED to BLUE mapping:



color = RED_TO_RED, offset = 32768



color = RED_TO_BLUE, offset = 32768

| Parameter | Description |
|---|---|
| color [IN] | RED_TO_RED - Red to Red |

| | |
|---|---|
| | GREEN_TO_RED - Green to Red<br>BLUE_TO_RED - Blue to Red<br>RED_TO_GREEN - Red to Green<br>GREEN_TO_GREEN - Green to Green<br>BLUE_TO_GREEN - Blue to Green<br>RED_TO_BLUE - Red to Blue<br>GREEN_TO_BLUE - Green to Blue<br>BLUE_TO_BLUE - Blue to Blue |
| coefficient [IN] | Value for the color converter (Range -32,768 … +32,767) |
| commit [IN] | FALSE - Do not persist (value restored back after power cycle)<br>TRUE - Persist setting |

## 5.21. PicoP_GetColorConverter ( color, coefficient, storageType )

This function returns the color converter value for a specific color pair in the 3x3 color converter matrix.

| Parameter | Description |
|---|---|
| color [IN] | RED_TO_RED - Red to Red<br>GREEN_TO_RED - Green to Red<br>BLUE_TO_RED - Blue to Red<br>RED_TO_GREEN - Red to Green<br>GREEN_TO_GREEN - Green to Green<br>BLUE_TO_GREEN - Blue to Green<br>RED_TO_BLUE - Red to Blue<br>GREEN_TO_BLUE - Green to Blue<br>BLUE_TO_BLUE - Blue to Blue |
| coefficient [OUT] | Value for the color converter (Range -32768 … +32767) |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

## 5.22. PicoP_SetFrameRateMode ( frameRateMode, displayVerticalScalingMode, displayHorizontalScalingMode, commit)

A MEMS Laser Beam Scanning system can dynamically adjust vertical resolution and frame rate; one can trade off between faster frame rate and slightly lower vertical resolution. The MicroVision PSE supports the following modes for Frame Rate vs. Vertical Resolution of Display and 3D Sensing:

a) 60Hz Frame Rate, 720 display vertical resolution, 740 sensing vertical resolution (w/ IR Overscan)
b) 70Hz Frame Rate, 640 display vertical resolution, 640 sensing vertical resolution (no IR Overscan)
c) 70Hz Frame Rate, 600 display vertical resolution, 640 sensing vertical resolution (w/ IR Overscan)

The PicoP_SetFrameRateMode() function allows the host to select these different modes, as well as to set how the display scaling will be performed if the input video resolution does not match the output resolution.

| Parameter | Description |
|---|---|
| **frameRateMode [IN]** | **FRAMERATE_60HZ_DISPLAY_720p_SENSING_740p - 60Hz Frame Rate, 720 display vertical resolution, 740 sensing vertical resolution (Default)**<br>**FRAMERATE_70HZ_DISPLAY_640p_SENSING_640p - 70Hz Frame Rate, 640 display vertical resolution, 640 sensing vertical resolution**<br>**FRAMERATE_70HZ_DISPLAY_600p_SENSING_640p - 70Hz Frame Rate, 600 display vertical resolution, 640 sensing vertical resolution** |
| **displayVerticalScalingMode [IN]** | **DISPLAY_VERTICAL_SCALING_DOWN - (Default) If input resolution > display resolution, downscale vertically to display resolution**<br>**DISPLAY_VERTICAL_SCALING_CROP_BOTTOM - If input resolution > display resolution, crop the bottom of the input video to reach the correct display vertical resolution**<br>**DISPLAY_VERTICAL_SCALING_CROP_EVEN - If input resolution > display resolution, crop evenly from the top and bottom of the input video to reach the correct display vertical resolution** |

| | |
|---|---|
| displayHorizontalScalingMode [IN] | DISPLAY_HORIZONTAL_SCALING_DOWN - (Default) If input resolution > display resolution, use horizontal interpolator to map video horizontally to correct horizontal resolution to maintain 16:9 aspect ratio<br>DISPLAY_HORIZONTAL_SCALING_NONE - Display horizontal pixels to fill the display horizontally with input pixels without maintaining original aspect ratio. |
| commit [IN] | FALSE - Do not persist (value restored back after power cycle)<br>TRUE - Persist setting |

## 5.23. PicoP_GetFrameRateMode ( frameRateMode, displayVerticalScalingMode, displayHorizontalScalingMode, storageType)

Return the output frame rate mode (frame rate vs. vertical display and sensing resolutions) and display scaling

| Parameter | Description |
|---|---|
| frameRateMode [OUT] | FRAMERATE_60HZ_DISPLAY_720p_SENSING_740p - 60Hz Frame Rate, 720 display vertical resolution, 740 sensing vertical resolution (Default)<br>FRAMERATE_70HZ_DISPLAY_640p_SENSING_640p - 70Hz Frame Rate, 640 display vertical resolution, 640 sensing vertical resolution<br>FRAMERATE_70HZ_DISPLAY_600p_SENSING_640p - 70Hz Frame Rate, 600 display vertical resolution, 640 sensing vertical resolution |
| displayVerticalScalingMode [OUT] | DISPLAY_VERTICAL_SCALING_DOWN - (Default) If input resolution > display resolution, downscale vertically to display resolution<br>DISPLAY_VERTICAL_SCALING_CROP_BOTTOM - If input resolution > display resolution, crop the bottom of the input video to reach the correct display vertical resolution<br>DISPLAY_VERTICAL_SCALING_CROP_EVEN - If input resolution > display resolution, crop evenly from the top and bottom of the input video to reach the correct display vertical resolution |

| | |
|---|---|
| displayHorizontalScalingMode [OUT] | DISPLAY_HORIZONTAL_SCALING_DOWN - (Default) If input resolution > display resolution, use horizontal interpolator to map video horizontally to correct horizontal resolution to maintain 16:9 aspect ratio<br>DISPLAY_HORIZONTAL_SCALING_NONE - Display horizontal pixels to fill the display horizontally with input pixels without maintaining original aspect ratio. |
| storageType [IN] | eCURRENT_VALUE - Current value in use<br>eVALUE_ON_STARTUP - Value persisted for PSE power on<br>eFACTORY_VALUE - Factory default |

# Chapter 6: System Management Functions

The System management functions enable the host application to query system status and health, set power modes, and field upgrade firmware.

## 6.1. PicoP_GetSystemInfo ( systemInfo)

This function allows the host to retrieve information about the PSE system which includes System Serial Number, Software Version and Electronics Version. This information can be used for troubleshooting and configuration management.

| Parameter | Description |
|---|---|
| **systemInfo [OUT]** | **Returned system information structure with the following information:**<br>**PSE System Serial Number**<br>**PSE Firmware Version**<br>**PSE Electronics Version**<br>**versionA reserved for future use**<br>**versionB reserved for future use**<br>**versionC reserved for future use**<br>**versionD reserved for future use**<br>**versionE reserved for future use** |

## 6.2. PicoP_GetSystemStatus ( systemStatus)

PSE allows the host to query the status of the PSE system; i.e. whether it is working properly or not. The function will also retrieve the system internal temperature.

| Parameter | Description |
|---|---|
| **systemStatus [OUT]** | **Returned systemStatus structure with the following information:**<br>**systemState:**<br>**ON = 0 - System running and ready to accept input video**<br>**STARTING - System Starting Up** |

| | |
|---|---|
| | CALIBRATION - System in Calibration mode (for MicroVision use only)<br>FAULT - System has faulted. Please power cycle.<br>SystemFault:<br>0 - system OK<br>Not 0 - System not OK, please contact MicroVision customer service<br>Temperature:<br>PSE internal Temperature<br>Data 0: for future expansion<br>Data 1: for future expansion<br>Data 2: for future expansion<br>Data 3: for future expansion<br>Data 4: for future expansion |

## 6.3. PicoP_GetEventLog ( numEvents, event)

PSE maintains a log of system events that have occurred. The host can retrieve the event log information of last events with the PicoP_GetEventLog() function. This function is typically used for system troubleshooting.

| Parameter | Description |
|---|---|
| numEvents [IN] | Number of events to be retrieved |
| event [OUT] | Returned events, list of event data structures with the following information:<br>session - session number when event occurred<br>eventide - Event Id<br>time - Event time stamp<br>data - Event data<br>cid - Event component ID<br>line - Event line number |

## 6.4. PicoP_RestoreFactoryConfig ( )

This function restores PSE system configuration to the factory default settings.

## 6.5. PicoP_CommitAll ( )

Commit all user settings to flash that have not yet been committed; i.e. were set with commit = 0.

## 6.6. PicoP_UpgradeSoftware (byteTotal, image)

The PSE supports field upgrading the embedded software of the system with the PicoP_UpgradeSoftware() function.

| Parameter | Description |
|---|---|
| **byteTotal [IN]** | **Size of the SW binary** |
| **image [IN]** | **Data of the SW binary** |

# Chapter 7: Input Control Functions

The Input Control Functions allow the host device to control and configure the PSE video input interface.

## 7.1. PicoP_GetInputVideoProperties ( horizontalPixels, verticalLines, frameRate )

Return the detected input video resolution and Frame Rate. This is a utility helper function that can be used by the application to help determine the input video format if it is not known.

| Parameter | Description |
|---|---|
| **horizontalPixels [OUT]** | **Detected number of pixels per line** |
| **verticalLines [OUT]** | **Detected number of video lines per single video frame** |
| **frameRate [OUT]** | **Detected number of video frames per second** |

## 7.2. PicoP_SetInputVideoState ( inputVideoState, commit)

This function enables or disables the PSE input video interface. When the input video is disabled, the PSE frame buffer will not be updated and the output video will contain the last received frame.

| Parameter | Description |
|---|---|
| **inputVideoState [IN]** | **ENABLED - Enable the input video**<br>**DISABLED - Disable the input video and statically display the last received video frame contained in the frame buffer.** |
| **commit [IN]** | **FALSE - Do not persist (value restored back after power cycle)**<br>**TRUE - Persist setting** |

## 7.3. PicoP_GetInputVideoState ( inputVideoState, storageType)

Returns the current state of the input video.

| Parameter | Description |
|---|---|
| **inputVideoState [OUT]** | **ENABLED - The input video is enabled**<br>**DISABLED - The input video is disabled** |
| **storageType [IN]** | **eCURRENT_VALUE - Current value in use**<br>**eVALUE_ON_STARTUP - Value persisted for PSE power on**<br>**eFACTORY_VALUE - Factory default** |

# Chapter 8: 3D Sensing Functions

## 8.1. PicoP_SetSensingState ( sensingState, commit)

Set the 3D Sensing function On or Off. When the 3D Sensing function is off, the IR laser is not pulsing and no touch events will be sent to the host device.

| Parameter | Description |
|---|---|
| **sensingState [IN]** | **ENABLED - Enable 3D Sensing function**<br>**DISABLED - Disable 3D Sensing function** |
| **commit [IN]** | **FALSE - Do not persist (value restored back after power cycle)**<br>**TRUE - Persist setting** |

## 8.2. PicoP_GetSensingState ( sensingState, storageType)

Return the state of the 3D sensing function.

| Parameter | Description |
|---|---|
| **sensingState [OUT]** | **ENABLED - 3D Sensing function is enabled**<br>**DISABLED - 3D Sensing function disabled** |
| **storageType [IN]** | **eCURRENT_VALUE - Current value in use**<br>**eVALUE_ON_STARTUP - Value persisted for PSE power on**<br>**eFACTORY_VALUE - Factory default** |

# Chapter 9: Rendering Functions

The Rendering functions allow the host to display test patterns by rendering them into the display Framebuffer.

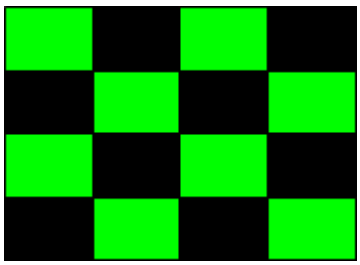
## 9.1. PicoP_DrawTestPattern ( testPattern, foregroundColor, backgroundColor)

This function enables the host to request the display of set of PSE internally generated test patterns. When enabled, the selected test pattern will replace any input video and the display the pattern covering the entire display area.

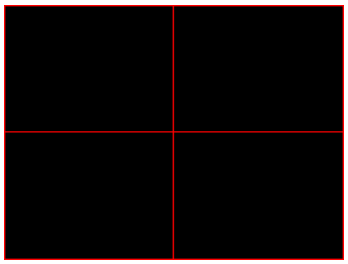
| Parameter | Description |
|---|---|
| **testPattern [IN]** | **eTEST_PATTERN_OFF: Turn Test Patterns Off, re-enable input video**<br>**eCHECKERBOARD_PATTERN: 4 x 4 Checkerboard**<br>**eCROSSHAIR_PATTERN: Bordered cross hair (1 pixel wide)**<br>**eGRID_PATTERN: Bordered grid pattern (1 pixel wide)**<br>**eCONSTANT_COLOR_PATTERN: Solid (constant) single color** |
| **foregroundColor [IN]** | **24-bit RGB Color Value for the test pattern** |
| **backgroundColor[IN]** | **24-bit RGB Color Value for the test pattern background** |

To turn off test patterns, the user shall call the PicoP_DrawTestPattern() function again with the eTEST_PATTERN_OFF parameter (the foregroundColor and backgroundColor parameters are don't care and ignored in this case).

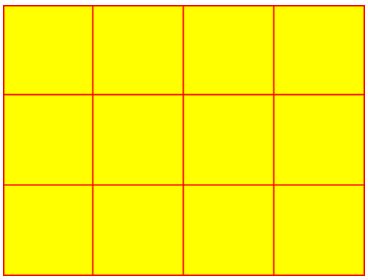
The built-in test patterns with sample colors are illustrated below:



4 x 4 Checkerboard
(Green foreground, Black background)
(0x00ff00, 0x000000)



Bordered crosshair
(Red foreground, Black background)
(0xff0000, 0x000000)



Bordered grid pattern
(Red foreground, yellow background)
(0xff0000, 0xffff00)

Solid single color pattern
(Purple foreground)
(0xff7fff)

For the solid single color pattern, the foreground color parameter is used to set the display color and the background color parameter is ignored (don't care)