Commander Architect Python

Python Mates Controller Library

Introduction

This library is developed to easily control Breadboard Mates modules using any device that can run Python by utilizing the Mates Controller Command Protocol. This applies to projects developed using Commander and Architect environments.

Supported Devices

This library is developed for Python3 and designed to be used with any operating system as long as it is supported by the pyserial library.

Installation

This library can be installed from the Python Packaging Index (PyPI) by running the command:

pip3 install mates-controller

Constructors

This section serves to provide brief discussion about the constructors that can be used to initialize the library.

MatesController(portName, resetFunction, debugStream, debugFileLength)

Constructs all the necessary attributes associated with an instance of a Mates Controller Object.

Parameters	Туре	Description
portName	str	the name of the port to be opened. Example: /dev/ttyUSB0 for linux
resetFunction	function	function used to perform a hard reset
debugStream (optional)	io.TextIOWrapper	text file object to write debugging code to, supply of none will result in no debugging. Ex. sys.stdout, open('log.txt', 'r+')
debugFileLength (optional)	int	determines the extent of debug history kept with respect to lines in a file, given a circular log. O indicates full history kept with no circular logging. Users must be careful here to manage storage space effectively

Note

If a debug file is specified, it should be opened using either 'w+' or 'r+' before running the begin() function of this library.

Example Specify Debug Output Simple def resetModule(): # perform reset of 100ms pulse to the RST pin # set reset pulse # wait for 100ms # unset reset pulse pass # Creates a new instance named 'mates' which utilizes: # - COM4 as the serial port # - resetModule as the reset function MatesController mates = MatesController("COM4", resetFunction=resetModule) def resetModule(): # perform reset of 100ms pulse to the RST pin # set reset pulse # wait for 100ms # unset reset pulse pass # Creates a new instance named 'mates' which utilizes: # - COM7 as the serial port # - resetModule as the reset function # - output_file as debug file stream # - debugFileLength of zero indicating no circular logging Th MatesController mates = cussion about the methods that ca MatesController("COM7", stance. resetFunction=resetModule, debugStream=output_file, debugFileLength=0)

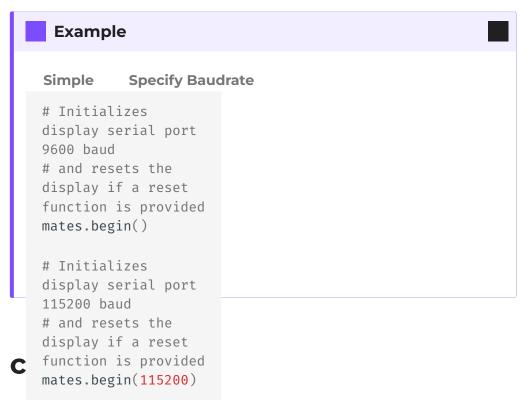
Begins the serial connection if portname not supplied in constructor.

begin(paugrate)

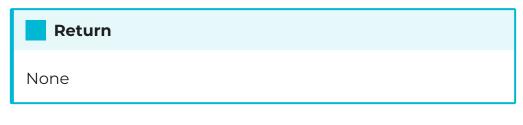
Parameters	Туре	Description
baudrate	str	the baudrate of the serial port (default: 9600)

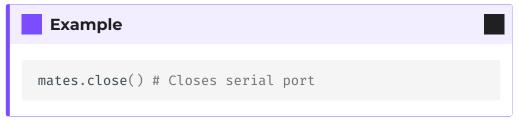


None



Closes opened serial port.





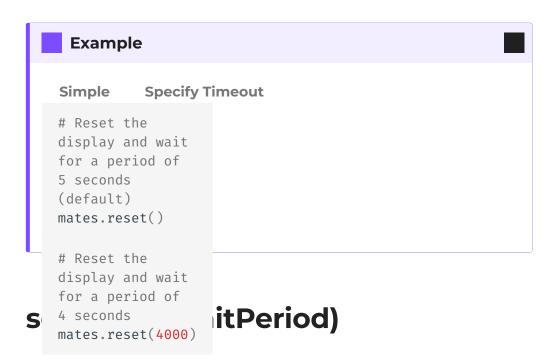
reset(waitPeriod)

Uses hardware driven signal to hard reset companion device.

Parameters	Туре	Description
wait Period	int	determines how long to wait (milliseconds) before checking for connection, must be within the uint16 datatype range (default: 5000)



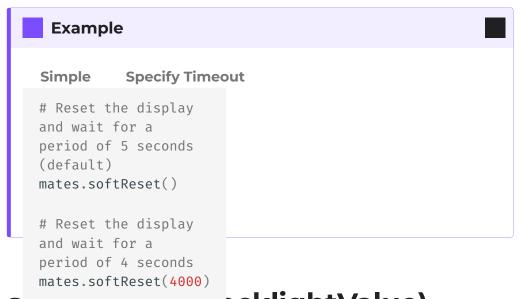
success or failure (boolean)



Sends a serial command to the connected device to trigger a reset.

Parameters	Туре	Description
waitPeriod	int	determines how long to wait (milliseconds) before checking for connection, must be within the uint16 datatype range (default: 5000)

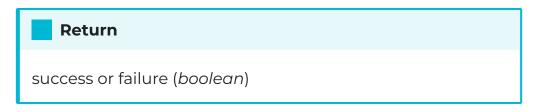




SeτραcκιιgητιρacklightValue)

Sets the intensity of the backlight of connected device.







setPage(pageIndex)

Sets the page to be displayed on the connected device.

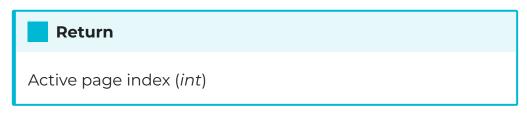


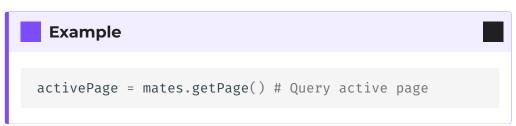




getPage()

Returns the index of the current page displayed by the connected device.





setWidgetValueByld(widgetId, value)

Sets the value of a specific widget based on the provided widgetId.

Parameters	Туре	Description
widgetId	int	the unique id of the target widget, must be within the int16 datatype range
value	int	the value the target widget will be set to, must be within the int16 datatype range





getWidgetValueById(widgetId)

Gets the value of a specific widget based on the provided identifier.

Parameters	Туре	Description
widgetId	int	the unique id of the target widget, must be within the int16 datatype range

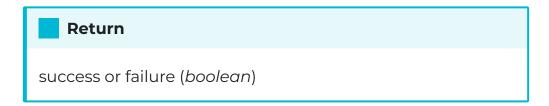




setWidgetValueByIndex(widgetType, widgetIndex, value)

Sets the value of a specific widget based on the index within a widget type.

Parameters	Туре	Description
widgetType	MatesWidget	the type of the target widget
widgetIndex	int	the index of the target widget, must be within the uint8 datatype range
value	int	the value the target widget will be set to, must be within the int16 datatype range







getWidgetValueByIndex(widgetType, widgetIndex)

Gets the value of a specific widget based on the index within a widget type.

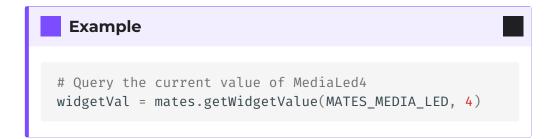
Parameters	Туре	Description
widgetType	MatesWidget	the type of the target widget
widgetIndex	int	the index of the target widget, must be within the uint8 datatype range

Return

Value of the widget specified by **widgetType** and **widgetIndex** (*int*)

Note

This function is not applicable to Int32 and Float LedDigits



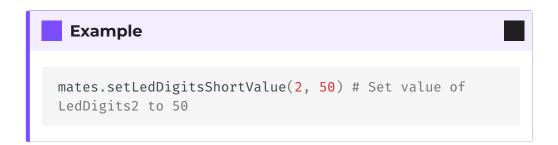
setLedDigitsShortValue(widgetIndex, value)

Sets the 16-bit integer value of the Led Digits widget specified by widgetIndex.

Parameters	Туре	Description
widgetIndex	int	the index of the LedDigits, must be within the uint8 datatype range
value	int	the value the target LedDigits will be set to, must be within the int16 datatype range



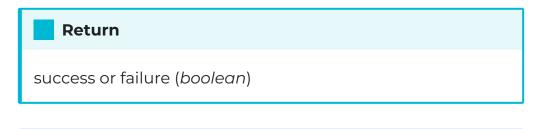




setLedDigitsLongValue(widgetIndex, value)

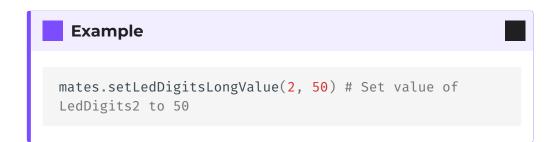
Sets the 32-bit integer value of the Led Digits widget specified by widgetIndex.

Parameters	Туре	Description
widgetIndex	int	the index of the LedDigits, must be within the uint8 datatype range
value	int	the value the target LedDigits will be set to, must be within the int32 datatype range



Note

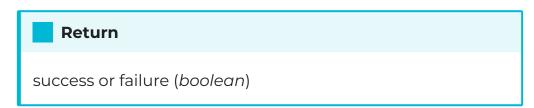
This function is only applicable for Int32 LedDigits



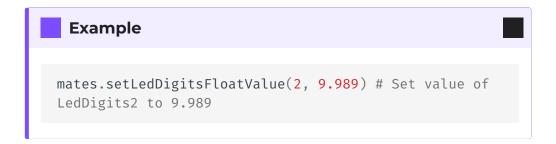
setLedDigitsFloatValue(widgetIndex, value):

Sets the 32-bit float value of the Led Digits widget specified by widgetIndex.

Parameters	Туре	Description
widgetIndex	int	the index of the LedDigits, must be within the uint8 datatype range
value	int	the value the target LedDigits will be set to, must be within the 32-bit float datatype range







setSpectrumValue(spectrumId, gaugeIndex, value)

Sets the value of the column (specified by gaugeIndex) of the spectrum widget (specified by spectrumId).

Parameters	Туре	Description
spectrumId	int	the unique id of the Spectrum widget, must be within the int16 datatype range
gaugeIndex	int	the gauge index/column of the Spectrum widget, must be within the uint8 datatype range
value	int	the value the target spectrum column will be set to, must be within the uint8 datatype range



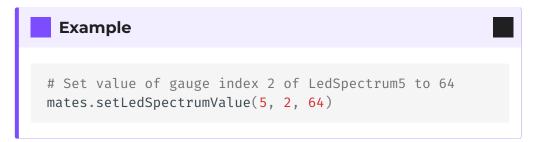


setLedSpectrumValue(ledSpectrumIndex, gaugeIndex, value)

Sets the value of the column (specified by gaugeIndex) of the Led Spectrum widget (specified by ledSpectrumIndex).

Parameters	Туре	Description
ledSpectrumIndex	int	the index of the LedSpectrum widget, must be within the uint8 datatype range
gaugeIndex	int	the gauge index/column of the Spectrum widget, must be within the uint8 datatype range
value	int	the value the target spectrum column will be set to, must be within the uint8 datatype range



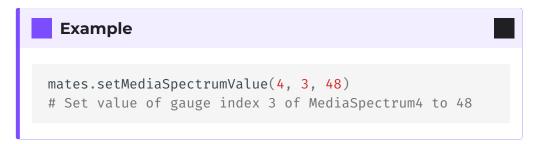


setMediaSpectrumValue(mediaIndex, gaugeIndex, value)

Sets the value of the column (specified by gaugeIndex) of the Media Spectrum widget (specified by ledSpectrumIndex).

Parameters	Туре	Description
mediaIndex	int	the index of the MediaSpectrum widget, must be within the uint8 datatype range
gaugeIndex	int	the gauge index/column of the Spectrum widget, must be within the uint8 datatype range
value	int	the value the target spectrum column will be set to, must be within the uint8 datatype range





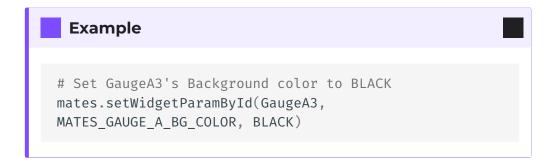
setWidgetParamByld(widgetId, param, value)

Sets the value of a widget parameter based on widget id and parameter id.

Parameters	Туре	Description
widgetId	int	the unique id of the target widget, must be within the int16 datatype range
param	int	

Parameters	Туре	Description
		the unique id of the target parameter, must be within the int16 datatype range
value	int	the value the target parameter will be set to, must be within the int16 datatype range

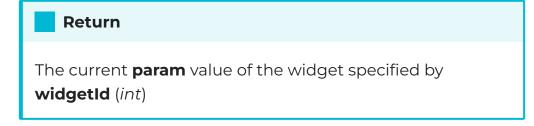


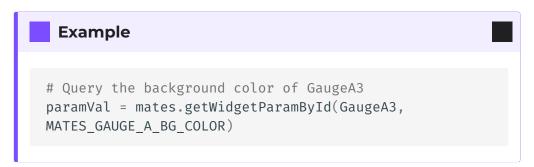


getWidgetParamById(widgetId, param)

Gets the value of a widget parameter based on widget id and parameter id.

Parameters	Туре	Description
widgetId	int	the unique id of the target widget, must be within the intl6 datatype range
param	int	the unique id of the target parameter, must be within the int16 datatype range

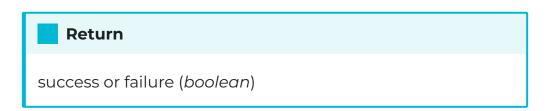




setWidgetParamByIndex(widgetType, widgetIndex, param, value)

Sets the value of a widget parameter based on widget index and parameter id.

Parameters	Туре	Description
widgetType	MatesWidget	the type of the target widget
widgetIndex	int	the index of the target widget, must be within the uint8 datatype range
param	int	the unique id of the target parameter, must be within the int16 datatype range
value	int	the value the target parameter will be set to, must be within the int16 datatype range





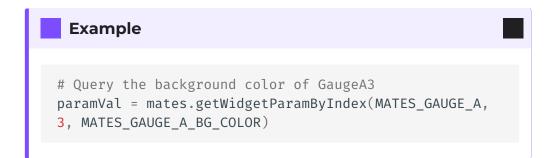
getWidgetParamByIndex(widgetType, widgetIndex, param)

Gets the value of a widget parameter based on widget index and parameter id.

Parameters	Туре	Description
widgetType	MatesWidget	the type of the target widget
widgetIndex	int	the index of the target widget, must be within the uint8 datatype range
param	int	the unique id of the target parameter, must be within the int16 datatype range



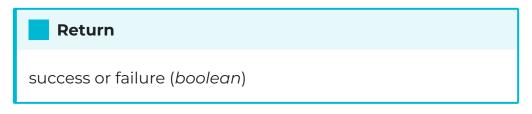
The current **param** value of the widget specified by **widgetType** and **widgetIndex** (*int*)

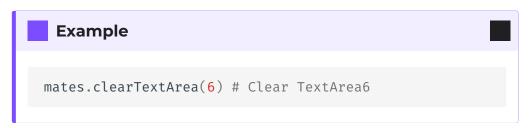


clearTextArea(textAreaIndex)

Clears a targeted Text Area.

Parameters	Туре	Description
textAreaIndex	int	the index of the target TextArea, must be within the uint16 datatype range





updateTextArea(textAreaIndex, textFormat, *formatArgs)

Updates the text displayed within Text Area widget.

Parameters	Туре	Description
textAreaIndex	int	the index of the target TextArea, must be within the uint16 datatype range
textFormat	str	the string or text format to be written to the TextArea
formatArgs		zero or more values to be formatted into the provided text format string



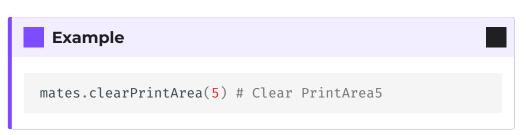


clearPrintArea(printAreaIndex: int)

Clears a targeted Print Area.

Parameters	Туре	Description
printAreaIndex	int	the index of the target PrintArea, must be within the uint16 datatype range





setPrintAreaColor565(printAreaIndex, rgb565)

Sets the color of a PrintArea Widget based on an rgb565 value.

Parameters	Туре	Description
printAreaIndex	int	the index of the target PrintArea, must be within the uint16 datatype range
rgb565	int	16-bit color to set widget to, value must be within uint16 datatype range



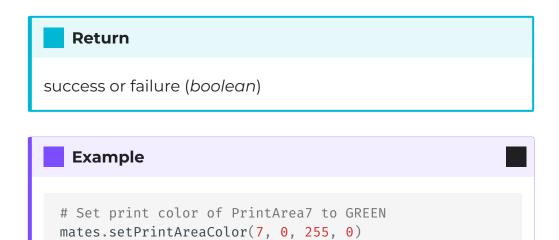


setPrintAreaColorRGB(printAreaIndex, red, green, blue)

Sets the color of a targeted Print Area.

Parameters	Туре	Description
printAreaIndex	int	the index of the target PrintArea, must be within the uint16 datatype range
red	int	

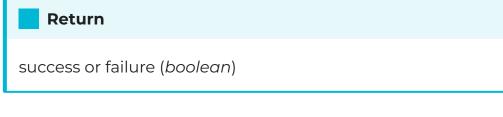
Parameters	Туре	Description
		8-bit red component of the target color, value must be within uint8 datatype range
green	int	8-bit green component of the target color, value must be within uint8 datatype range
blue	int	8-bit blue component of the target color, value must be within uint8 datatype range



appendArrayToPrintArea(printAreaIndex, array)

Appends an array of 8-bit integers to a targeted Print Area.

Parameters	Туре	Description
printAreaIndex	int	the index of the target PrintArea, must be within the uint16 datatype range
buffer	[int]	the list of values to be appended to PrintArea, values must be within the uint8 datatype range

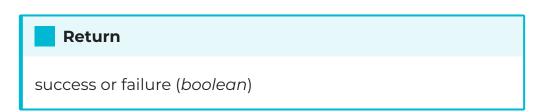




appendStringToPrintArea(printAreaIndex, textFormat, *formatArgs)

Appends text to a targeted Print Area.

Parameters	Туре	Description
printAreaIndex	int	the index of the target PrintArea, must be within the uint16 datatype range
textFormat	str	the string or text format to be appended to the PrintArea
formatArgs		zero or more values to be formatted into the provided text format string





appendToScopeWidget(scopeIndex, buffer)

Appends a list of integers to a Scope widget.

Parameters	Туре	Description
scopelndex	int	the index of the target Scope, must be within the uint8 datatype range
buffer	[int]	the list of values to be appended to Scope, values must be within the int16 datatype range



```
data = {0xF8, 0x7F, 0x1F}
mates.appendToScopeWidget(7, data, 3) # Append data
to Scope Widget 7
```

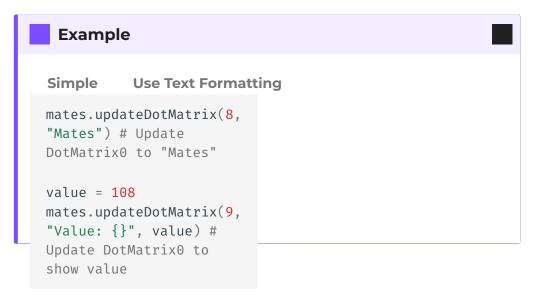
updateDotMatrixWidget(matrixIndex, textFormat, *formatArgs)

Changes the text displayed by the target Dot Matrix widget.

Parameters	Туре	Description
matrixIndex	int	the index of the target DotMatrix, must be within the uint8 datatype range
textFormat	str	the string or text format to be appended to the DotMatrix
formatArgs		zero or more values to be formatted into the provided text format string



success or failure (boolean)



getButtonEventCount()

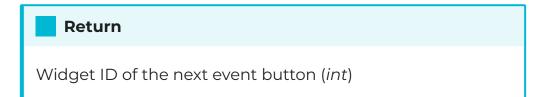
Gets the number of events recorded from applicable button widgets.



```
# Get the number of logged button events
buttonEvents = mates.getButtonEventCount()
```

getNextButtonEvent()

Gets the next event source logged from applicable buttons.





getSwipeEventCount()

Gets the number of events recorded from swipe gestures.

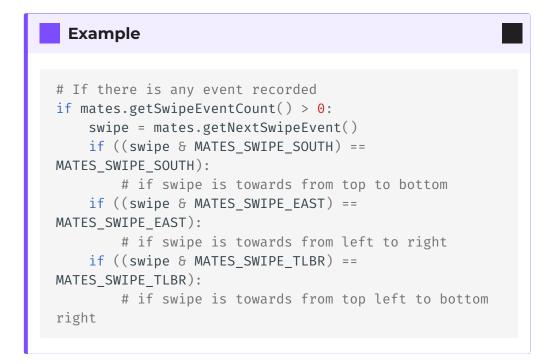


```
# Get the number of logged swipe events
swipeEvents = mates.getSwipeEventCount()
```

getNextSwipeEvent()

Gets the next swipe event value.





getVersion()

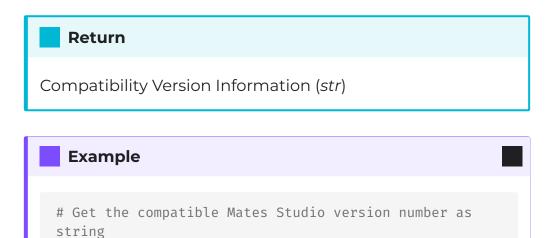
Helper function to obtain the version of the Python Mates Controller library.





getCompatibility()

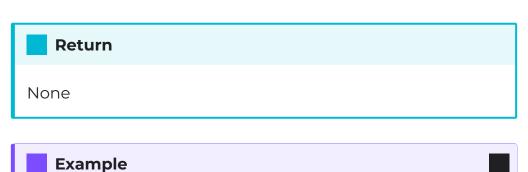
Helper function to obtain the version of the Mates Studio compatible with this library version.



compatVersion = mates.getCompatibility()

printVersion()

Debugging function to print the version of the Mates Studio compatible along with this specific library version.



Prints library version and compatible Mates Studio
version to debug serial
mates.printVersion()

getError()

This function can be used to investigate errors that occurred while controlling the display module. Description of the possible errors is discussed in here.

