Mates Controller Arduino Library

Introduction

This library is developed to easily control Breadboard Mates modules using Arduino-compatible boards by utilizing the Mates Controller Command Protocol. This applies to projects developed using Commander and Architect environments.

For working examples of using the library and its functions in a project, refer to the *examples* and *extras* directories in the repository.

Constructors

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

MatesController(serial, resetPin, mode)

This is the main constructor for the library. It creates a unique instance and sets the specified display serial port and reset pin. If *serial* is not a HardwareSerial (or SoftwareSerial for AVR devices), the Serial stream needs to be initialized manually before running begin(baudrate, resetModule) function.

Parameters	Type	Description
serial	Stream	The serial port to use for controlling the display module
resetPin (optional)	uint8_t	Arduino reset pin to use for resetting the display module (default: 4)
mode (optional)	uint8_t	Arduino reset pulse to use when performing reset (default: LOW)

Example No. 1

```
// Creates a new instance named 'mates' which utilizes:
//  - Serial as UART
//  - Pin 4 of Arduino as Reset Pin (default)
//  - Reset mode as a LOW pulse (default)
MatesController mates = MatesController(Serial);
```

Example No. 2

```
// Creates a new instance named 'mates' which utilizes:
//  - Serial as UART
//  - Pin 5 of Arduino as Reset Pin
//  - Reset mode as a LOW pulse (default)
MatesController mates = MatesController(Serial, 5);
```

MatesController(serial, dbSerial, resetPin, mode)

This is an alternative constructor for the library. It creates a unique instance and sets the specified display serial port, debug serial and reset pin. If not using a HardwareSerial (or SoftwareSerial for AVR devices), the Serial stream needs to be initialized manually before running the begin(baudrate, resetModule) function.

Parameters	Type	Description
serial	Stream	The serial port to use for controlling the display module
dbSerial	Stream	The serial port to use for printing debug messages
resetPin (optional)	uint8_t	Arduino reset pin to use for resetting the display module (default: 4)
mode (optional)	uint8_t	Arduino reset pulse to use when performing reset (default: LOW)

> **Note**
>
> If a debug serial port is specified, it should be initialized before running the begin() function of this library.

Functions

This section serves to provide brief discussion about the functions that can be used with an initialized MatesController object.

begin(baudrate, resetModule)

This function must be used once to initialize the Serial port at the start of the Arduino application and to reset or synchronize with the display.

Parameters	Type	Description
baudrate (optional)	int32_t	Baudrate setting to be used to control the display module (default: 9600)
resetModule (optional)	bool	Indicates whether the module should be reset from the hardware reset pin (default: true)

> **Note**
>
> 1. Baudrate is ignored when not using a HardwareSerial (or SoftwareSerial for AVR devices) to communicate with the display. In that case, the Serial/Stream instance needs to be initialize before using this function.
> 2. If resetModule is false, this function will attempt to synchronize with the display.

> **Return**
>
> success or failure (*boolean*)

Example No. 1

```
// Initializes display serial port with 9600 (default) and resets the display
mates.begin();
```

Example No. 2

```
// Initializes display serial port with 115200 baud and resets the display
mates.begin(115200);
```

Example No. 3

```
// Initializes display serial port with 19200 baud and skips reset
mates.begin(19200, false);
```

> **Note**
>
> 1. Ensure that the baudrate matches the baudrate setting of the Mates Studio Commander/Architect project
> 2. If a debug serial port is specified, it should be initialized manually before running the begin() function of this library.
> 3. If not using reset, users needs to be aware of the boot timing of the module. This should be around 3-5 seconds or more depending on the project after power on. If more time is needed to sync, set a higher boot timeout using setBootTimeout(timeout)

isReady()

This function can be used to determine if the module is in sync with the Arduino host.

> **Return**
>
> Sync Status (*boolean*)

Example

```
// Check if the module is in sync
if (mates.isReady()) {
    // Write to or read from widgets
} else {
    mates.sync(); // Try to resync with the module
}
```

autoResync(attempts, waitPeriod)

This function can be used to setup auto resynchronization when an error occurs.

Parameters	Type	Description
attempts	uint8_t	Number of resync attempts to perform
waitPeriod (optional)	uint16_t	Timeout period to wait for every resync attempt (default: boot timeout)

> **Return**
>
> none

Example No. 1

```
// Setup 3 automatic attempts to resync with default timeout
mates.autoResync(3);
```

Example No. 2

```
// Setup 5 automatic attempts to resync with 10000ms timeout
mates.autoResync(5, 10000);
```

sync(resetToPage0, waitPeriod)

This function can be used to establish synchronization between the BBM module and the Arduino compatible host.

Parameters	Type	Description
resetToPage0	bool	Indicates whether to go to Page0 after a successful synchronization (default: true)
waitPeriod (optional)	uint16_t	Timeout period to wait until the display is ready (default: boot timeout)

> **■ Return**
>
> success or failure (*boolean*)

Example No. 1

```
// Attempts to synchronize with the display
if (mates.sync()) {
   // Do something if synchronization was successful
} else {
   // Do something if synchronization failed
}
```

Example No. 2

```
// Attempts to synchronize with the display
if (mates.sync(true)) {
   // Do something if synchronization was successful
   // and project returned to Page0
} else {
   // Do something if synchronization failed
}
```

Example No. 3

```
// Attempts to synchronize with the display with a timeout of 10000
if (mates.sync(true, 10000)) {
   // Do something if synchronization was successful
   // and project returned to Page0
} else {
```

```
    // Do something if synchronization failed
}
```

reset(waitPeriod)

This function can be used to reset the display by sending a reset pulse from the reset pin specified through the contructor. The default wait period is 5 seconds (5000 ms) or as specified by setBootTimeout(timeout).

The function finishes as soon as the display sends the ready signal or the wait period passes.

Parameters	Type	Description
waitPeriod (optional)	uint16_t	Timeout period to wait until the display is ready (default: boot timeout)

Return

success or failure (*boolean*)

Example No. 1

```
// Reset the display and wait for
mates.reset(); // a period of 5 seconds (default)
        // (actually the current boot timeout which is 5s by default)
```

Example No. 2

```
// Reset the display and wait for
mates.reset(4000); // a period of 4 seconds
```

softReset(waitPeriod)

This function can be used to reset the display by sending a reset command. The default wait period is 5 seconds (5000 ms) or as specified by setBootTimeout(timeout).

The function finishes as soon as the display sends the ready signal or the wait period passes.

Parameters	Type	Description
waitPeriod (optional)	uint16_t	Timeout period to wait until the display is ready (default: boot timeout)

Return

success or failure (*boolean*)

setBootTimeout(timeout)

This function can be used to set the wait period during reset and softReset.

Parameters	Type	Description
timeout	uint32_t	New timeout period to wait until the display is ready

> **Return**
>
> success or failure (*boolean*)

Example

```
mates.setBootTimeout(10000); // sets boot timeout to a period of 10 seconds
```

resetBootTimeout(timeout)

This function can be used to reset the wait period during reset and softReset to the default 5 seconds.

> **Return**
>
> none

Example

```
mates.resetBootTimeout(); // resets boot timeout to the default period
```

attachErrorHandler(handler)

Parameters	Type	Description
handler	MatesErrorHandler	Custom function to handle errors as they come

This function can be used to attach and error handler function to the library.

> **Return**
>
> none

Example No. 1

```cpp
MatesController mates = MatesController(Serial);

void matesErrorHandler(MatesError error) {
  while (true) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(200);
    digitalWrite(LED_BUILTIN, LOW);
    delay(200);
  } // Blink builtin LED and block project execution
  // This is not ideal but can be used to as simple error indication
  // Errors should be handled as shown in Example 2
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  // Sets 'matesErrorHandler' as the function for handling possible MatesError
  mates.attachErrorHandler(matesErrorHandler);
  mates.begin(9600);

  // do something...
}

void loop() {
  // do something...
}
```

Example No. 2

```cpp
MatesController mates = MatesController(Serial);

void matesErrorHandler(MatesError error) {
  switch (error) {
    case MATES_ERROR_COMMAND_FAILED:
      // Do something when last command is invalid
      break;
    case MATES_ERROR_RESPONSE_TIMEOUT:
      // Do something when the expected response from
      // the last command wasn't received on time
      break;
    case MATES_ERROR_COMMAND_TIMEOUT:
      // Do something when the expected acknowledgement from
      // the last command wasn't received on time
      break;
    case MATES_ERROR_NOT_INITIALIZED:
      // Do something when the display is not yet ready
      break;
    default:
      break;
  }
```

```
    }

    void setup() {
        // Sets 'matesErrorHandler' as the function for handling possible MatesError
        mates.attachErrorHandler(matesErrorHandler);
        mates.begin(9600);

        // do something...
    }

    void loop() {
        // do something...
    }
```

setBacklight(value)

This function can be used to set the backlight level to the *value* specified.

Parameters	Type	Description
value	uint8_t	The target backlight level

> ■ **Return**
>
> success or failure (*boolean*)

Example

```
mates.setBacklight(7); // Set backlight value to 7
```

setPage(page)

This function can be used to navigate to the specified *page*.

Parameters	Type	Description
page	uint16_t	The target page index

> ■ **Return**
>
> success or failure (*boolean*)

Example

```
mates.setPage(1); // Navigate to Page1
```

getPage()

This function can be used to query the current active page.

> **Return**
>
> Active page index (*uint16_t*)

Example

```
uint16_t activePage = mates.getPage(); // Query active page
```

setWidgetValue(widget, value)

This function can be used to set the 16-bit integer *value* of the specified *widget*

Parameters	Type	Description
widget	int16_t	The ID of the target widget
value	int16_t	The new value for the widget

> **Return**
>
> success or failure (*boolean*)

Example

```
mates.setWidgetValue(MediaGaugeB0, 50); // Set value of MediaGaugeB0 to 50
```

> **Note**
>
> 1. All applicable widget types are listed in here.
> 2. This function is not applicable to *Int32* and *Float* LedDigits

getWidgetValue(widget)

This function can be used to query the specified *widget*'s value.

Parameters	Type	Description
widget	int16_t	The ID of the target widget

Example

```
// Query the current value of MediaLed4
int16_t widgetVal = mates.getWidgetValue(MediaLed4);
```

> **Note**
>
> 1. All applicable widget types are listed in here.
> 2. This function is not applicable to *Int32* and *Float* LedDigits

setWidgetValue(type, index, value)

This function can be used to set the 16-bit integer *value* of the widget specified by *type* and *index*.

Parameters	Type	Description
type	MatesWidget	The type of the target widget
index	int8_t	The index of the target widget
value	int16_t	The new value for the widget

> **Return**
>
> success or failure (*boolean*)

Example

```
// Set value of MediaGaugeB0 to 50
mates.setWidgetValue(MATES_MEDIA_GAUGE_B, 0, 50);
```

> **Note**
>
> 1. All applicable widget types are listed in here.
> 2. This function is not applicable to *Int32* and *Float* LedDigits

getWidgetValue(type, index)

This function can be used to query the value of the widget specified by *type* and *index*.

Parameters	Type	Description
type	MatesWidget	The type of the target widget
index	int8_t	The index of the target widget

> **Return**
>
> Value of the widget specified by **type** and **index** (*int16_t*)

Example

```
// Query the current value of MediaLed4
int16_t widgetVal = mates.getWidgetValue(MATES_MEDIA_LED, 4);
```

> **Note**
>
> 1. All applicable widget types are listed in here.
> 2. This function is not applicable to *Int32* and *Float* LedDigits

setLedDigitsValue(index, value)

This function can be used to set the 16-bit integer *value* of the LedDigits specified by *index*.

Parameters	Type	Description
index	uint8_t	The index of the target LedDigits
value	int16_t	The new value for the LedDigits

> **Return**
>
> success or failure (*boolean*)

Example

```
mates.setLedDigitsValue(0, 1234); // Set value of LedDigits0 to 1234
```

> **Note**
>
> This function is only applicable for *Int16* LedDigits

setLedDigitsValue(index, value)

This function can be used to set the 32-bit integer *value* of the LedDigits specified by *index*.

Parameters	Type	Description
index	uint8_t	The index of the target LedDigits
value	int32_t	The new value for the LedDigits

> **Return**
>
> success or failure (*boolean*)

Example

```
mates.setLedDigitsValue(0, 602214076); // Set value of LedDigits0 to 602214076
```

> **Note**
>
> This function is only applicable for *Int32* LedDigits

setLedDigitsValue(index, value)

This function can be used to set the float *value* of the LedDigits specified by *index*.

Parameters	Type	Description
index	uint8_t	The index of the target LedDigits
value	float	The new value for the LedDigits

> **Return**
>
> success or failure (*boolean*)

Example

```
// Set value of LedDigits1 to 3.1416
mates.setLedDigitsValue(LedDigits1, 3.1416);
```

setSpectrumValue(widget, gaugeIndex, value)

This function can be used to set the *value* of a specified gauge index of the spectrum *widget* specified.

Parameters	Type	Description
widget	int16_t	The ID of the target spectrum widget
gaugeIndex	uint8_t	The gauge index of the target spectrum widget
value	uint8_t	The new value for the widget

> **■ Return**
>
> success or failure (*boolean*)

Example

```
// Set value of gauge index 2 of LedSpectrum5 to 64
mates.setSpectrumValue(LedSpectrum5, 2, 64);
```

> **■ Note**
>
> This function is only applicable for LedSpectrum and MediaSpectrum

setLedSpectrumValue(index, gaugeIndex, value)

This function can be used to set the *value* of a specified *gaugeIndex* of the Led Spectrum widget determined by *index*.

Parameters	Type	Description
index	uint8_t	The index of the target Led Spectrum widget
gaugeIndex	uint8_t	The gauge index of the target Led Spectrum widget
value	uint8_t	The new value for the column/row of the widget

setMediaSpectrumValue(type, index, gaugeIndex, value)

This function can be used to set the *value* of a specified *gaugeIndex* of the Media Spectrum widget determined by *index*.

Parameters	Type	Description
index	uint8_t	The index of the target Led Spectrum widget
gaugeIndex	uint8_t	The gauge index of the target Led Spectrum widget
value	uint8_t	The new value for the column/row of the widget

setMediaColorLedValue(index, r, g, b)

This function can be used to set the 32-bit integer *value* of the LedDigits specified by *index*.

Parameters	Type	Description
index	uint8_t	The index of the target MediaColorLed
r	uint8_t	The red component of the new color of the MediaColorLed
g	uint8_t	The green component of the new color of the MediaColorLed
b	uint8_t	The blue component of the new color of the MediaColorLed

Example

mates.setMediaColorLedValue(3, 255, 0, 0); // Set value of MediaColorLed3 to RED

Note

This function is only applicable for MediaColorLeds

setWidgetParam(widget, param, value)

This function can be used to set the parameter (*param*) of the target *widget* to the specified *value*.

Parameters	Type	Description
widget	int16_t	The ID of the target widget
param	int16_t	The target widget parameter
value	int16_t	The new value for the widget parameter

Return

success or failure (*boolean*)

Example

// Set GaugeA3's Background color to BLACK
mates.setWidgetParam(GaugeA3, MATES_GAUGE_A_BG_COLOR, BLACK);

Note

All applicable widget types are listed in here.

getWidgetParam(widget, param)

This function can be used to query the parameter (*param*) of the target *widget*.

Parameters	Type	Description
widget	int16_t	The ID of the target widget
param	int16_t	The target widget parameter

> **Return**
>
> The current **param** value of the **widget** (int16_t)

Example

```
// Query the background color of GaugeA3
int16_t paramVal = mates.getWidgetParam(GaugeA3, MATES_GAUGE_A_BG_COLOR);
```

> **Note**
>
> All applicable widget types are listed in here.

setWidgetParam(type, index, param, value);

This function can be used to set the parameter (*param*) of the target widget, determined by *type* and *index*, to the specified *value*.

Parameters	Type	Description
type	MatesWidget	The type of the target widget
index	int8_t	The index of the target widget
param	int16_t	The target widget parameter
value	int16_t	The new value for the widget parameter

> **Return**
>
> success or failure (*boolean*)

Example

```
// Set GaugeA3's Background color to BLACK
mates.setWidgetParam(MATES_GAUGE_A, 3, MATES_GAUGE_A_BG_COLOR, BLACK);
```

> **Note**
>
> All applicable widget types are listed in here.

getWidgetParam(type, index, param)

This function can be used to query the parameter (*param*) of the target widget, determined by *type* and *index*.

Parameters	Type	Description
type	MatesWidget	The type of the target widget
index	int8_t	The index of the target widget
param	int16_t	The target widget parameter

> **Return**
>
> The current **param** value of the widget specified by **type** and **index** (int16_t)

Example

```
// Query the background color of GaugeA3
int16_t paramVal = mates.getWidgetParam(MATES_GAUGE_A, 3, MATES_GAUGE_A_BG_COLOR);
```

> **Note**
>
> All applicable widget types are listed in here.

setBufferSize(size)

This function can be used to adjust the max string buffer *size* to be used when composing a string for a TextArea or a PrintArea. The string composition is done by updateTextArea(index, format, ...), updateDotMatrix(index, format, ...) and appendToPrintArea(index, format, ...)

Parameters	Type	Description
size	uint16_t	The new buffer size (max: 1000)

clearTextArea(index)

This function can be used to clear the TextArea specified by *index*.

Parameters	Type	Description
index	uint16_t	The index of the target TextArea widget

> **Return**
>
> success or failure (*boolean*)

Example

```
mates.clearTextArea(6); // Clear TextArea6
```

updateTextArea(index, format, ...)

This function can be used to update the contents of the TextArea specified by *index* with the text formed by *format* and the additional arguments.

Parameters	Type	Description
index	uint16_t	The index of the target TextArea widget
format	const char *	The text to be written to the Text Area
...	-	Additional values to replace the format specifiers in *format*

> **Return**
>
> success or failure (*boolean*)

Example No. 1

```
mates.updateTextArea(2, "Mates"); // Update TextArea2 to "Mates"
```

Example No. 2

```
int value = 76;
mates.updateTextArea(3, "Value is %d", 76); // Print value to TextArea3
```

updateTextArea(index, str)

This function can be used to update the contents of the TextArea specified by *index* with the String 'str'.

Parameters	Type	Description
index	uint16_t	The index of the target TextArea widget
str	String	The String to be written to the Text Area

> **Return**
>
> success or failure (*boolean*)

Example

```
String str = "Mates";
mates.updateTextArea(2, str); // Update TextArea2 to 'str'
```

clearPrintArea(index)

This function can be used to clear the PrintArea specified by *index*.

Parameters	Type	Description
index	uint16_t	The index of the target PrintArea widget

> **Return**
>
> success or failure (*boolean*)

Example

```
mates.clearPrintArea(5); // Clear PrintArea5
```

setPrintAreaColor(index, rgb565)

This function can be used to set the print color (*rgb565*) used by the PrintArea specified by *index*.

Parameters	Type	Description
index	uint16_t	The index of the target PrintArea widget
rgb565	int16_t	The color as a 16-bit RGB565 value

> **Return**
>
> success or failure (*boolean*)

Example

```
// Set print color of PrintArea4 to RED (0xF800)
mates.setPrintAreaColor(4, 0xF800);
```

setPrintAreaColor(index, r, g, b)

This function can be used to set the print color used by the PrintArea specified by *index*. The color is determined by *r*, *g* and *b*.

Parameters	Type	Description
index	uint16_t	The index of the target PrintArea widget
r	uint8_t	The red component of the new color value
g	uint8_t	The green component of the new color value
b	uint8_t	The blue component of the new color value

> **Return**
>
> success or failure (*boolean*)

Example

```
mates.setPrintAreaColor(7, 0, 255, 0); // Set print color of PrintArea7 to GREEN
```

appendToPrintArea(index, buffer, len)

This function can be used to append a number of bytes (*len*) from the data in *buffer* to the PrintArea specified by *index* .

Parameters	Type	Description
index	uint16_t	The index of the target Print Area widget
buffer	const int8_t *	The source of data to be appended
len	uint16_t	The number of bytes to be sent

> **Return**
>
> success or failure (*boolean*)

Example

```
int8_t data[] = {0xF8, 0x7F, 0x1F};
mates.appendToPrintArea(7, data, 3); // Append data to PrintArea7
```

appendToPrintArea(index, format, ...)

This function can be used to append contents to the PrintArea specified by *index* with the text formed by *format* and the additional arguments.

Parameters	Type	Description
index	uint16_t	The index of the target Print Area widget
format	const char *	The text to be written to the PrintArea
...	-	Additional values to replace the format specifiers in *format*

> **Return**
>
> success or failure (*boolean*)

Example No. 1

```
mates.appendToPrintArea(8, "Mates"); // Append "Mates" to PrintArea8
```

Example No. 2

```
int value = 108;
// Append value as text to PrintArea9
mates.appendToPrintArea(9, "Value: %d", 108);
```

appendToPrintArea(index, str)

This function can be used to append contents to the PrintArea specified by *index* with the String provided.

Parameters	Type	Description
index	uint16_t	The index of the target Print Area widget
str	String	The text to be written to the PrintArea

> ### ◼ Return
>
> success or failure (*boolean*)

Example

```
String str = "Mates";
mates.appendToPrintArea(2, str); // // Append 'str' to PrintArea2
```

appendToScope(index, buffer, len)

This function can be used to append a number of 16-bit values (*len*) from the data in *buffer* to the Scope widget specified by *index*.

Parameters	Type	Description
index	uint16_t	The index of the target Scope widget
buffer	const int16_t *	The source of data to be appended
len	uint16_t	The number of values to be sent

> ### ◼ Return
>
> success or failure (*boolean*)

Example

```
int16_t data[] = {0xF8, 0x7F, 0x1F};
mates.appendToScope(7, data, 3); // Append data to Scope7
```

updateDotMatrix(index, format, ...)

This function can be used to append contents to the DotMatrix specified by *index* with the text formed by *format* and the additional arguments.

Parameters	Type	Description
index	uint16_t	The index of the target DotMatrix widget
format	const char *	The text to be written to the DotMatrix
...	-	Additional values to replace the format specifiers in *format*

> **Return**
>
> success or failure (*boolean*)

Example No. 1

```
mates.updateDotMatrix(8, "Mates"); // Update DotMatrix0 to "Mates"
```

Example No. 2

```
int value = 108;
mates.updateDotMatrix(9, "Value: %d", 108); // Update DotMatrix0 to show value
```

updateDotMatrix(index, str)

This function can be used to update the contents of the DotMatrix specified by *index* with the String 'str'.

Parameters	Type	Description
index	uint16_t	The index of the target DotMatrix widget
str	String	The String to be written to the DotMatrix

> **Return**
>
> success or failure (*boolean*)

```
String str = "Mates";
mates.updateDotMatrix(2, str); // Update DotMatrix2 to 'str'
```

getButtonEventCount()

This function can be used to query the number of button events recorded by a touch screen module

> **Return**
>
> Number of recorded button events (*uint16_t*)

Example

```
// Query the number of button events recorded
uint16_t btnEvents = mates.getButtonEventCount();
```

getNextButtonEvent()

This function can be used to query the source of next recorded button event

> **Return**
>
> Widget ID of the next event button (*int16_t*)

Example

```
// If there is any event recorded
if (mates.getButtonEventCount() > 0) {
    int16_t button = mates.getNextButtonEvent();
    switch (button) {
        case MediaButton1: // if the button pressed is MediaButton1
            // do something
            break;
        // add more possible cases here...
        default:
            break;
    }
}
```

getSwipeEventCount()

This function can be used to query the number of swipe events recorded by a touch screen module

> **Return**
>
> Number of recorded swipe events (*uint16_t*)

getNextButtonEvent()

This function can be used to query the source of next recorded button event

> **Return**
>
> Swipe event (int16_t)

Example

```
// If there is any event recorded
if (mates.getSwipeEventCount() > 0) {
    int16_t swipe = mates.getNextSwipeEvent();
    if ((swipe & MATES_SWIPE_SOUTH) == MATES_SWIPE_SOUTH) {
        // if swipe is towards from top to bottom
    }
    if ((swipe & MATES_SWIPE_EAST) == MATES_SWIPE_EAST) {
        // if swipe is towards from left to right
    }
    if ((swipe & MATES_SWIPE_TLBR) == MATES_SWIPE_TLBR) {
        // if swipe is towards from top left to bottom right
    }
}
```

getVersion()

This function can be used to query the version number of the library.

> **Return**
>
> Version Information (*String*)

Example

```
// Get the library version number as string
String matesVersion = mates.getVersion();
```

getCompatibility()

This function can be used to query the version number of Mates Studio compatible with the version of the library.

Example

```
// Get the compatible Mates Studio version number as string
String compatVersion = mates.getCompatibility();
```

printVersion()

This function can be used to print the version number of the library and the compatible Mates Studio version to the debug serial port. If no debug serial was specified in the constructor, this function does nothing.

> **Return**
>
> none

Example

```
// 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.

> **Return**
>
> Current error code (*MatesError*)

Example

```
// Checks the last error that occurred
int error = mates.getError();
if (error == MATES_ERROR_NONE) {
    // Last command was successful
}
```