GlowBit.py

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 glowbit.matrix8x8._textScroll Class Reference

Public Member Functions

- def **__init__** (self, string, y=0, x=0, colour=0xFFFFFF, bgColour=0)

Public Attributes

- **x**
- **y**
- **colour**
- **bgColour**
- **string**

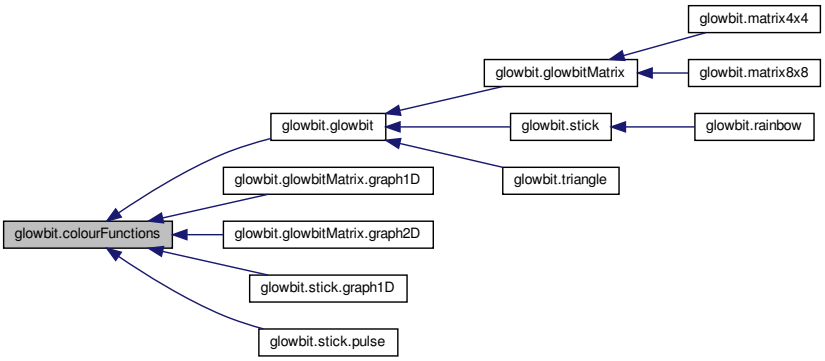
The documentation for this class was generated from the following file:

- glowbit.py

3.2 glowbit.colourFunctions Class Reference

Methods for transforming colours to 32-bit packed GlowBit colour values.

Inheritance diagram for glowbit.colourFunctions:



Public Member Functions

- def wheel (self, pos)

  Converts an integer "colour wheel position" to a packed 32-bit RGB GlowBit colour value.
- def rgbColour (self, r, g, b)

  Converts the r, g, and b integer arguments to a packed 32-bit RGB GlowBit colour value.
- def glowbitColour2RGB (self, colour)

  Converts a 32-bit GlowBit colour value to an (R,G,B) tuple.
- def red (self)

  Returns the GlowBit colour value for pure red.
- def green (self)

  Returns the GlowBit colour value for pure green.
- def blue (self)

  Returns the GlowBit colour value for pure blue.
- def yellow (self)

  Returns the GlowBit colour value for yellow.
- def purple (self)

  Returns the GlowBit colour value for purple.
- def cyan (self)

  Returns the GlowBit colour value for cyan.
- def white (self)

  Returns the GlowBit colour value for white.
- def black (self)

  Returns the GlowBit colour value for black.

3.2.1 Detailed Description

Methods for transforming colours to 32-bit packed GlowBit colour values.

A packed 32-bit GlowBit colour is an integer with 8-bits per colour channel data encoded in hexadecimal as follows:

0x00RRGGBB

where RR, GG, and BB are hexadecimal values (decimal [0,255]) and the most significant 8 bits are reserved and left as zero.

3.2.2 Member Function Documentation

3.2.2.1 glowbitColour2RGB()

```
def glowbit.colourFunctions.glowbitColour2RGB (
              self,
              colour )
```

Converts a 32-bit GlowBit colour value to an (R,G,B) tuple.

Parameters

colour	A 32-bit GlowBit colour value

Returns

A tuple in the format (R,G,B) containing the RGB components of the colour parameter

3.2.2.2 rgbColour()

```
def glowbit.colourFunctions.rgbColour (
              self,
              r,
              g,
              b )
```

Converts the r, g, and b integer arguments to a packed 32-bit RGB GlowBit colour value.

All arguments are required as this is a micropython viper function.

Parameters

r	The red intensity, [0,255]
g	The green intensity, [0,255]
b	The blue intensity, [0,255]

Returns

Packed 32-bit GlowBit colour value

3.2.2.3 wheel()

```
def glowbit.colourFunctions.wheel (
              self,
              pos )
```

Converts an integer "colour wheel position" to a packed 32-bit RGB GlowBit colour value.

The "pos" argument is required as this is a micropython viper function.

Parameters

pos	Colour wheel position [0,255] is mapped to a pure hue in the RGB colourspace. A value of 0 or 255 is mapped to pure red with a smooth red-yellow-green-blue-purple-magenta-red transion for other values.

Returns

32-bit integer GlowBit colour value

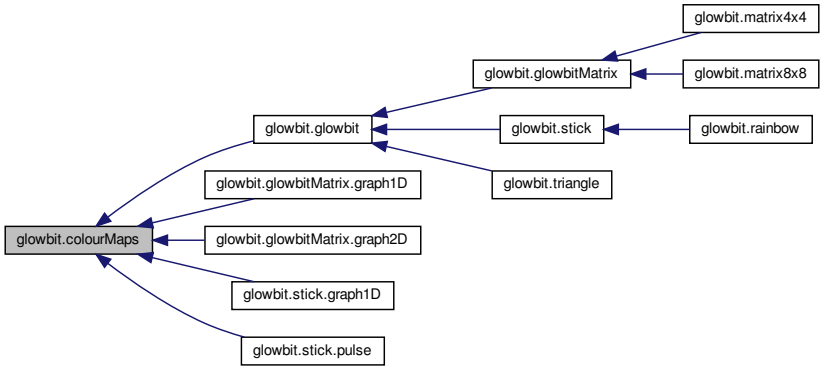
The documentation for this class was generated from the following file:

- glowbit.py

3.3 glowbit.colourMaps Class Reference

Methods which calculate colour gradients.

Inheritance diagram for glowbit.colourMaps:



Public Member Functions

- def colourMapSolid (self, index, minIndex, maxIndex)

  Trivial colourmap method which always returns the colour in the parent object.
- def colourMapRainbow (self, index, minIndex, maxIndex)

  Maps the pure hue colour wheel between minIndex and maxIndex.

3.3.1 Detailed Description

Methods which calculate colour gradients.

Custom colour map methods can be written and passed to several GlowBit library methods (eg: glowbit.stick.↩graph1D) but must accept the same positional arguments as the methods in this class:

def colourMapFunction(self, index, minIndex, maxIndex):

3.3.2 Member Function Documentation

3.3.2.1 colourMapRainbow()

```
def glowbit.colourMaps.colourMapRainbow (
              self,
              index,
              minIndex,
              maxIndex )
```

Maps the pure hue colour wheel between minIndex and maxIndex.

Parameters

index	The value to be mapped
minIndex	The value of index mapped to a colour wheel angle of 0 degrees
maxIndex	The value of index mapped to a colour wheel angle of 360 degrees

Returns

The 32-bit packed GlowBit colour value

3.3.2.2 colourMapSolid()

```
def glowbit.colourMaps.colourMapSolid (
              self,
              index,
              minIndex,
              maxIndex )
```

Trivial colourmap method which always returns the colour in the parent object.

Parameters

index	Dummy argument for compatibility with colourmap method API
minIndex	Dummy argument for compatibility with colourmap method API
maxIndex	Dummy argument for compatibility with colourmap method API

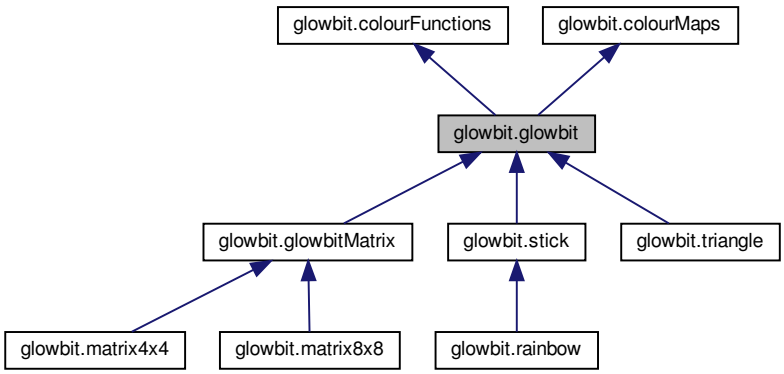
The documentation for this class was generated from the following file:

- glowbit.py

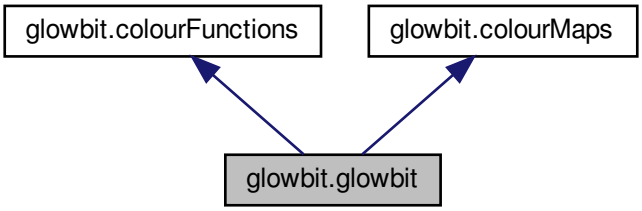
3.4 glowbit.glowbit Class Reference

Low-level methods common to all GlowBit classes.

Inheritance diagram for glowbit.glowbit:



Collaboration diagram for glowbit.glowbit:



Public Member Functions

- def pixelsShow (self)

  Pushes the internal pixel data buffer to the physical GlowBit LEDs.
- def pixelSet (self, i, colour)

  Sets the i'th GlowBit LED to a 32-bit GlowBit colour value.
- def pixelSetNow (self, i, colour)

  Sets the i'th GlowBit LED to a 32-bit GlowBit colour value and updates the physical LEDs.
- def pixelAdd (self, i, colour)

  Adds a 32-bit GlowBit colour value to the i'th LED in the internal buffer only.
- def pixelSaturatingAdd (self, i, colour)

  Adds a 32-bit GlowBit colour value to the i'th LED in the internal buffer.
- def pixelsFill (self, colour)

  Fills all pixels with a solid colour value.
- def pixelsFillNow (self, colour)

  Fills all pixels with a solid colour value and updates the physical LEDs.

- def blankDisplay (self)

  Blanks the entire GlowBit display.
- def getPixel (self, N)

  Returns the 32-bit GlowBit colour value of the i'th LED.
- def updateRateLimitFPS (self, rateLimitFPS)

  Sets a new value for the GlowBit display's frames per second (FPS) limiter.
- def power (self)

  Calculates an estimate for the total power draw given the current display data.
- def chaos (self, iters=100)

  Sets random colour values on every LED on the attached GlowBit display.

Public Attributes

- **lastFrame_ms**
- **rateLimit**

Static Public Attributes

- **sideset_init**
- **OUT_LOW**
- **out_shiftdir**
- **SHIFT_LEFT**
- **autopull**
- **True**
- **pull_thresh**

3.4.1 Detailed Description

Low-level methods common to all GlowBit classes.

3.4.2 Member Function Documentation

3.4.2.1 blankDisplay()

```
def glowbit.glowbit.blankDisplay (
            self )
```

Blanks the entire GlowBit display.

ie: sets the colour value of all GlowBit LEDs to zero in the internal buffer and updates the physical LEDs.

3.4.2.2 chaos()

```
def glowbit.glowbit.chaos (
            self,
            iters = 100 )
```

Sets random colour values on every LED on the attached GlowBit display.

This function is blocking, it does not return until the number of frames specified in the iters parameter have been drawn.

Parameters

iters	The number of frames to draw.

3.4.2.3 getPixel()

```
def glowbit.glowbit.getPixel (
              self,
              N )
```

Returns the 32-bit GlowBit colour value of the i'th LED.

Parameters

i	The index of the LED

Returns

The 32-bit GlowBit colour value of the i'th LED

3.4.2.4 pixelAdd()

```
def glowbit.glowbit.pixelAdd (
              self,
              i,
              colour )
```

Adds a 32-bit GlowBit colour value to the i'th LED in the internal buffer only.

Data colour corruption will occur if the sum result of any RGB value exceeds 255. Care must be taken to avoid this manually. eg: if the blue channel's resulting intensity value is 256 it will be set to zero and the red channel incremented by 1. See the colourFunctions class documentation for the 32-bit GlowBit colour specification.

NB: For efficiency, this method does not do any index bounds checking. If the value of the parameter i is larger than the number of LEDs it will cause an IndexError exception.

Parameters

i	An LED's index
colour	The 32-bit GlowBit colour value

3.4.2.5 pixelSaturatingAdd()

```
def glowbit.glowbit.pixelSaturatingAdd (
```

```
                self,
                i,
                colour )
```

Adds a 32-bit GlowBit colour value to the i'th LED in the internal buffer.

This function performs "saturating" arithmetic. It is much slower than pixelAdd but will saturate at 255 to avoid data corruption.

NB: For efficiency, this method does not do any index bounds checking. If the value of the parameter i is larger than the number of LEDs it will cause an IndexError exception.

Parameters

i	An LED's index
colour	The 32-bit GlowBit colour value

3.4.2.6 pixelSet()

```
def glowbit.glowbit.pixelSet (
                self,
                i,
                colour )
```

Sets the i'th GlowBit LED to a 32-bit GlowBit colour value.

NB: For efficiency, this method does not do any bounds checking. If the value of the parameter i is larger than the number of LEDs it will cause an IndexError exception.

Parameters

i	An LED's index
colour	The 32-bit GlowBit colour value

3.4.2.7 pixelSetNow()

```
def glowbit.glowbit.pixelSetNow (
                self,
                i,
                colour )
```

Sets the i'th GlowBit LED to a 32-bit GlowBit colour value and updates the physical LEDs.

NB: For efficiency, this method does not do any index bounds checking. If the value of the parameter i is larger than the number of LEDs it will cause an IndexError exception.

Parameters

i	An LED's index
colour	The 32-bit GlowBit colour value

3.4.2.8 pixelsFill()

```
def glowbit.glowbit.pixelsFill (
              self,
              colour )
```

Fills all pixels with a solid colour value.

Parameters

colour	The 32-bit GlowBit colour value

3.4.2.9 pixelsFillNow()

```
def glowbit.glowbit.pixelsFillNow (
              self,
              colour )
```

Fills all pixels with a solid colour value and updates the physical LEDs.

Parameters

colour	The 32-bit GlowBit colour value

3.4.2.10 pixelsShow()

```
def glowbit.glowbit.pixelsShow (
              self )
```

Pushes the internal pixel data buffer to the physical GlowBit LEDs.

This function must be called before the connected GlowBit LEDs will change colour.

Note that several GlowBit library methods call this method unconditionally (eg: glowbit.blankDisplay) or optionally (eg: by passing the update = True parameter to stick.graph1D())

3.4.2.11 power()

```
def glowbit.glowbit.power (
              self )
```

Calculates an estimate for the total power draw given the current display data.

Use as a general guide only, error range is around 10-20%.

The estimate is a 4th order polynomical interpolation given measurements of white brightness. The power consumption of pure colours will tend to be under-estimated by up to about 10-20%.

Data was measured at a supply voltage of 3.3V but supply current was not found to vary significantly with supply voltage.

Returns

The current consumption of the framebuffer in amps.

3.4.2.12 updateRateLimitFPS()

```
def glowbit.glowbit.updateRateLimitFPS (
              self,
              rateLimitFPS )
```

Sets a new value for the GlowBit display's frames per second (FPS) limiter.

Parameters

rateLimitFPS	An integer in units of frames per second.

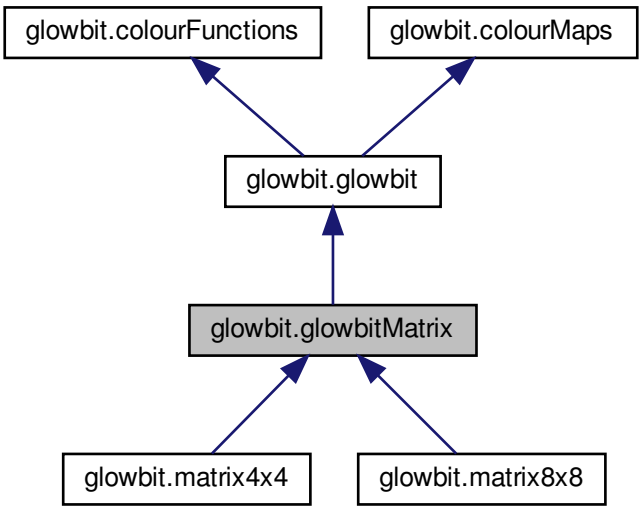
The documentation for this class was generated from the following file:

- glowbit.py

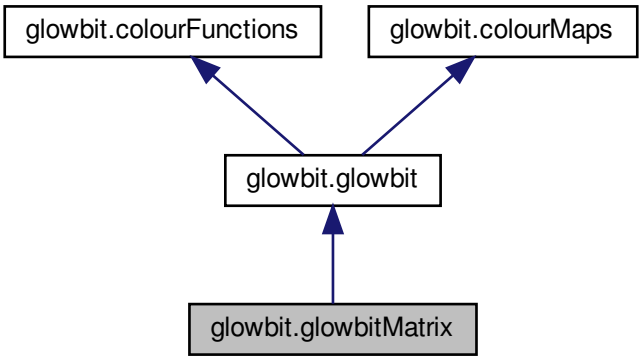
3.5 glowbit.glowbitMatrix Class Reference

Methods specific to 2D matrix displays and tiled arrangements thereof.

Inheritance diagram for glowbit.glowbitMatrix:



Collaboration diagram for glowbit.glowbitMatrix:



Classes

- class graph1D

  One dimensional graph object for graph bars on GlowBit Matrix displays.
- class graph2D

  Object for drawing 2 dimensional time series graphs on GlowBit Matrix displays.
- class raindrop

  A class used by the rain() demonstration.

Public Member Functions

- def pixelSetXY (self, x, y, colour)

  Sets the colour value of the GlowBit LED at a given x-y coordinate.
- def pixelSetXYNow (self, x, y, colour)

  Sets the colour value of the GlowBit LED at a given x-y coordinate and immediately calls pixelsShow() to update the physical LEDs.
- def pixelSetXYClip (self, x, y, colour)

  Sets the colour value of the GlowBit LED at a given x-y coordinate.
- def pixelAddXY (self, x, y, colour)

  Adds the colour value to the GlowBit LED at a given (x,y) coordinate.
- def pixelAddXYClip (self, x, y, colour)

  Adds the colour value to the GlowBit LED at a given (x,y) coordinate.
- def getPixelXY (self, x, y)

  Returns the 32-bit GlowBit colour value of the LED at a given (x,y) coordinate.
- def drawLine (self, x0, y0, x1, y1, colour)

  Draws a straight line between (x0,y0) and (x1,y1) in the specified 32-bit GlowBit colour.
- def drawTriangle (self, x0, y0, x1, y1, x2, y2, colour)

  Draws a triangle with vertices (corners) at (x0,y0), (x1, y1), and (x2,y2).
- def drawRectangle (self, x0, y0, x1, y1, colour)

  Draws a rectangle with upper-left corner (x0,y0) and lower right corner (x1, y1).
- def drawRectangleFill (self, x0, y0, x1, y1, colour)

  Draws a rectangle with upper-left corner (x0,y0) and lower right corner (x1, y1).
- def drawRectangleFillAdd (self, x0, y0, x1, y1, colour)

  Draws a rectangle with upper-left corner (x0,y0) and lower right corner (x1, y1).
- def drawCircle (self, x0, y0, r, colour)

  Draws a circle with center (x0,y0) and radius r.
- def newGraph1D (self, originX=0, originY=7, length=8, direction="Up", minValue=0, maxValue=255, colour=0xFFFFFF, colourMap="Solid", update=False)

  Wrapper method to create a graph1D object.
- def updateGraph1D (self, graph, value)

  Updates a graph1D object, draws it to the display buffer.
- def updateGraph2D (self, graph, value)

  Updates a 2D graph with a new value.
- def lineDemo (self, iters=10)

  Demonstrate drawing an animated line.
- def fireworks (self, iters=10)

  Demonstrate drawing randomly placed, randomly coloured, expanding circles.
- def circularRainbow (self)

  Demonstration of a rainbow effect is pseudo-polar coordinates.
- def rain (self, iters=200, density=1)

  A "digital rain" demonstration.
- def textDemo (self, text="Scrolling Text Demo")

  Demonstrates creation of non-blocking scrolling text.
- def bounce (self, iters=500)

  Draws a single pixel at a random coordinate and "bounces" it around the display.
- def demo (self)

  Runs several demo functions.

Additional Inherited Members

3.5.1 Detailed Description

Methods specific to 2D matrix displays and tiled arrangements thereof.

This class should not be used directly; its methods are inherited by the glowbit.matrix8x8 and glowbit.matrix4x4 classes.

3.5.2 Member Function Documentation

3.5.2.1 circularRainbow()

```
def glowbit.glowbitMatrix.circularRainbow (
              self )
```

Demonstration of a rainbow effect is pseudo-polar coordinates.

This function intentionally uses integer arithmetic for performance reasons. As such, it is drawn half a pixel off center.

The function animates 255 frames then returns.

3.5.2.2 drawCircle()

```
def glowbit.glowbitMatrix.drawCircle (
              self,
              x0,
              y0,
              r,
              colour )
```

Draws a circle with center (x0,y0) and radius r.

The circle's outline is drawn in the specified colour. Pixels inside the circle are not modified.

Parameters

x0	The x coordinate of the circle's center
y0	The y coordinate of the circle's center
colour	A packed 32-bit GlowBit colour value

3.5.2.3 drawLine()

```
def glowbit.glowbitMatrix.drawLine (
```

```
                self,
                x0,
                y0,
                x1,
                y1,
                colour )
```

Draws a straight line between (x0,y0) and (x1,y1) in the specified 32-bit GlowBit colour.

If pixel is drawn off the screen a "clipping" effect will be inherited from the behaviour of pixelSetXYClip(). ie: Pixels landing off the screen will not be drawn.

Parameters

x0	The line's starting x coordinate
y0	The line's starting y coordinate
x1	The line's ending x coordinate
y1	The line's ending y coordinate
colour	A packed 32-bit GlowBit colour value

3.5.2.4 drawRectangle()

```
def glowbit.glowbitMatrix.drawRectangle (
                self,
                x0,
                y0,
                x1,
                y1,
                colour )
```

Draws a rectangle with upper-left corner (x0,y0) and lower right corner (x1, y1).

All edge lines are drawn with the specified colour.

Pixels inside the rectangle are left unmodified.

If pixel is drawn off the screen a "clipping" effect will be inherited from the behaviour of pixelSetXYClip(). ie: Pixels landing off the screen will not be drawn.

Parameters

x0	The x coordinate of the upper left corner
y0	The y coordinate of the upper left corner
x1	The x coordinate of the lower right corner
y1	The y coordinate of the lower right corner
colour	A packed 32-bit GlowBit colour value

3.5.2.5 drawRectangleFill()

```
def glowbit.glowbitMatrix.drawRectangleFill (
                self,
                x0,
                y0,
                x1,
                y1,
                colour )
```

Draws a rectangle with upper-left corner (x0,y0) and lower right corner (x1, y1).

The rectangle is then filled to form a solid block of the specified colour.

This method overwrites pixel data with the colour value.

If pixel is drawn off the screen a "clipping" effect will be inherited from the behaviour of pixelSetXYClip(). ie: Pixels landing off the screen will not be drawn.

Parameters

x0	The x coordinate of the upper left corner
y0	The y coordinate of the upper left corner
x1	The x coordinate of the lower right corner
y1	The y coordinate of the lower right corner
colour	A packed 32-bit GlowBit colour value

3.5.2.6 drawRectangleFillAdd()

```
def glowbit.glowbitMatrix.drawRectangleFillAdd (
                self,
                x0,
                y0,
                x1,
                y1,
                colour )
```

Draws a rectangle with upper-left corner (x0,y0) and lower right corner (x1, y1).

The rectangle is then filled to form a solid block of the specified colour.

This method adds a colour to every pixel, allowing a rectangle to be drawn over the top of other pixel data.

If pixel is drawn off the screen a "clipping" effect will be inherited from the behaviour of pixelSetXYClip(). ie: Pixels landing off the screen will not be drawn.

Parameters

x0	The x coordinate of the upper left corner
y0	The y coordinate of the upper left corner
x1	The x coordinate of the lower right corner
y1	The y coordinate of the lower right corner
colour	A packed 32-bit GlowBit colour value

3.5.2.7 drawTriangle()

```
def glowbit.glowbitMatrix.drawTriangle (
              self,
              x0,
              y0,
              x1,
              y1,
              x2,
              y2,
              colour )
```

Draws a triangle with vertices (corners) at (x0,y0), (x1, y1), and (x2,y2).

All lines are drawn with the specified colour.

If pixel is drawn off the screen a "clipping" effect will be inherited from the behaviour of pixelSetXYClip(). ie: Pixels landing off the screen will not be drawn.

Parameters

x0	The x coordinate of the first vertex
y0	The y coordinate of the first vertex
x1	The x coordinate of the second vertex
y1	The y coordinate of the second vertex
x2	The x coordinate of the third vertex
y2	The y coordinate of the third vertex
colour	A packed 32-bit GlowBit colour value

3.5.2.8 fireworks()

```
def glowbit.glowbitMatrix.fireworks (
              self,
              iters = 10 )
```

Demonstrate drawing randomly placed, randomly coloured, expanding circles.

Note that pixelsFill(0) is only called after drawing an expanding circle, simulating a mostly filled circle. Gaps are an artefact of the circle drawing algorithm, not a bug.

3.5.2.9 getPixelXY()

```
def glowbit.glowbitMatrix.getPixelXY (
              self,
              x,
              y )
```

Returns the 32-bit GlowBit colour value of the LED at a given (x,y) coordinate.

If the (x,y) coordinate falls outside of the display's boundary an IndexError exception may be thrown or the GlowBit colour value of an undefined pixel may be returned.

Parameters

i	The index of the LED

Returns

The 32-bit GlowBit colour value of the i'th LED

3.5.2.10 newGraph1D()

```
def glowbit.glowbitMatrix.newGraph1D (
              self,
              originX = 0,
              originY = 7,
              length = 8,
              direction = "Up",
              minValue = 0,
              maxValue = 255,
              colour = 0xFFFFFF,
              colourMap = "Solid",
              update = False )
```

Wrapper method to create a graph1D object.

Calling matrix.graph1D() directly is recommended - this is only here so that it appears more clearly in the Doxygen.

3.5.2.11 pixelAddXY()

```
def glowbit.glowbitMatrix.pixelAddXY (
              self,
              x,
              y,
              colour )
```

Adds the colour value to the GlowBit LED at a given (x,y) coordinate.

The coordinate assumes an origin in the upper left of the display with x increasing to the right and y increasing downwards.

If the x-y coordinate falls outside the display's boundary this function will "wrap-around". For example, A dot placed just off the right edge will appear along the left edge.

Data colour corruption will occur if the sum result of any RGB value exceeds 255. Care must be taken to avoid this manually. eg: if the blue channel's resulting intensity value is 256 it will be set to zero and the red channel incremented by 1. See the colourFunctions class documentation for the 32-bit GlowBit colour specification.

Parameters

x	The x coordinate of the GlowBit LED. x must be an integer.
y	The y coordinate of the GlowBit LED. y must be an integer.
colour	A packed 32-bit GlowBit colour value

3.5.2.12 pixelAddXYClip()

```
def glowbit.glowbitMatrix.pixelAddXYClip (
              self,
              x,
              y,
              colour )
```

Adds the colour value to the GlowBit LED at a given (x,y) coordinate.

The coordinate assumes an origin in the upper left of the display with x increasing to the right and y increasing downwards.

If the x-y coordinate falls outside the display's boundary the display's internal buffer will not be modified.

Data colour corruption will occur if the sum result of any RGB value exceeds 255. Care must be taken to avoid this manually. eg: if the blue channel's resulting intensity value is 256 it will be set to zero and the red channel incremented by 1. See the colourFunctions class documentation for the 32-bit GlowBit colour specification.

Parameters

x	The x coordinate of the GlowBit LED. x must be an integer.
y	The y coordinate of the GlowBit LED. y must be an integer.
colour	A packed 32-bit GlowBit colour value

3.5.2.13 pixelSetXY()

```
def glowbit.glowbitMatrix.pixelSetXY (
              self,
              x,
              y,
              colour )
```

Sets the colour value of the GlowBit LED at a given x-y coordinate.

The coordinate assumes an origin in the upper left of the display with x increasing to the right and y increasing downwards.

If the x-y coordinate falls outside the display's boundary this function will "wrap-around". For example, A dot placed just off the right edge will appear along the left edge in the same row.

Advanced: If seeking maximum speed consider modifying the ar[] array directly

Parameters

x	The x coordinate of the GlowBit LED. x must be an integer.
y	The y coordinate of the GlowBit LED. y must be an integer.
colour	A packed 32-bit GlowBit colour value

3.5.2.14 pixelSetXYClip()

```
def glowbit.glowbitMatrix.pixelSetXYClip (
              self,
              x,
              y,
              colour )
```

Sets the colour value of the GlowBit LED at a given x-y coordinate.

The coordinate assumes an origin in the upper left of the display with x increasing to the right and y increasing downwards.

If the x-y coordinate falls outside the display's boundary the display's internal buffer will not be modified.

Parameters

x	The x coordinate of the GlowBit LED. x must be an integer.
y	The y coordinate of the GlowBit LED. y must be an integer.
colour	A packed 32-bit GlowBit colour value

3.5.2.15 pixelSetXYNow()

```
def glowbit.glowbitMatrix.pixelSetXYNow (
              self,
              x,
              y,
              colour )
```

Sets the colour value of the GlowBit LED at a given x-y coordinate and immediately calls pixelsShow() to update the physical LEDs.

The coordinate assumes an origin in the upper left of the display with x increasing to the right and y increasing downwards.

If the x-y coordinate falls outside the display's boundary this function will "wrap-around". For example, A dot placed just off the right edge will appear along the left edge.

Advanced: If seeking maximum speed consider modifying the ar[] array directly

Parameters

x	The x coordinate of the GlowBit LED. x must be an integer.
y	The y coordinate of the GlowBit LED. y must be an integer.
colour	A packed 32-bit GlowBit colour value

3.5.2.16 rain()

```
def glowbit.glowbitMatrix.rain (
              self,
              iters = 200,
              density = 1 )
```

A "digital rain" demonstration.

Parameters

iters	The number of frames on which raindrops can be drawn
density	The density of raindrops in units of "drops per 4x4 square". The number of drops on the screen will be kept at (number of pixels)∗(density)/16

3.5.2.17 textDemo()

```
def glowbit.glowbitMatrix.textDemo (
              self,
              text = "Scrolling Text Demo" )
```

Demonstrates creation of non-blocking scrolling text.

Only compatible with the GlowBit Matrix 8x8 and tiled arrangements thereof.

Parameters

text	A string of text which is scrolled across the top row of the display.

3.5.2.18 updateGraph1D()

```
def glowbit.glowbitMatrix.updateGraph1D (
              self,
              graph,
              value )
```

Updates a graph1D object, draws it to the display buffer.

If the graph1D object was created with "update = True" this function will call pixelsShow() to update the physical display before returning.

Parameters

graph	A graph1D object as returned by glowbitMatrix.graph1D()
value	The value to draw to the graph. It will be mapped to the graph bet

3.5.2.19 updateGraph2D()

```
def glowbit.glowbitMatrix.updateGraph2D (
              self,
              graph,
              value )
```

Updates a 2D graph with a new value.

Parameters

graph	A graph2D object created graph2D
value	A new value to draw to the graph. This value will be drawn on the right edge and the oldest value will be deleted.

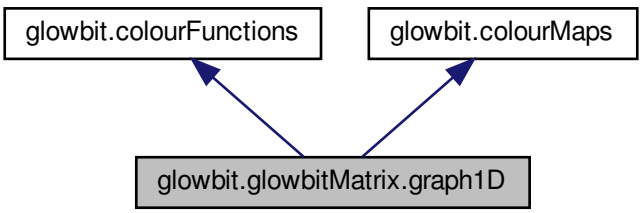
The documentation for this class was generated from the following file:

- glowbit.py

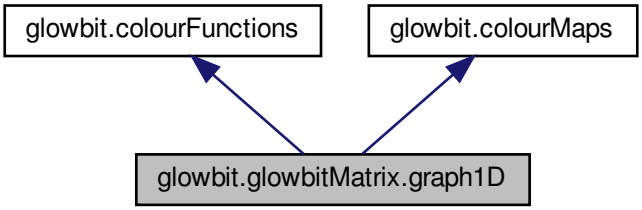
3.6 glowbit.glowbitMatrix.graph1D Class Reference

One dimensional graph object for graph bars on GlowBit Matrix displays.

Inheritance diagram for glowbit.glowbitMatrix.graph1D:



Collaboration diagram for glowbit.glowbitMatrix.graph1D:



Public Member Functions

- def __init__ (self, originX=0, originY=7, length=8, direction="Up", minValue=0, maxValue=255, colour=0xF↩FFFFF, colourMap="Solid", update=False)

  Initialisation routine for the glowbit.matrix.graph1D object.

Public Attributes

- **minValue**
- **maxValue**
- **originX**
- **originY**
- **length**
- **orientation**
- **inc**
- **m**
- **update**
- **colour**
- **colourMap**

3.6.1 Detailed Description

One dimensional graph object for graph bars on GlowBit Matrix displays.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 __init__()

```
def glowbit.glowbitMatrix.graph1D.__init__ (
                self,
                originX = 0,
                originY = 7,
                length = 8,
                direction = "Up",
                minValue = 0,
                maxValue = 255,
                colour = 0xFFFFFF,
                colourMap = "Solid",
                update = False )
```

Initialisation routine for the glowbit.matrix.graph1D object.

A graph1D object will be drawn with a fixed width of 1 pixel and arbitrary length.

Parameters

originX	The x coordinate of the graph's origin. The "minValue" argument will be mapped to this pixel.
originY	The y coordinate of the graph's origin. The "minValue" argument will be mapped to this pixel.
length	The length, in pixels, of the graph's drawing area.
direction	One of "Up", "Down", "Left", or "Right". Specifies the direction in which the graph will be drawn.
minValue	The value which will be mapped to the origin.
maxValue	The value which will be mapped to the 'end' of the graph. The (x,y) coordinate will be 'length' pixels away from the origin in the direction specified by 'direction'.
colour	A packed 32-bit GlowBit colour value. Used by the "Solid" colourmap, ignored by the "Rainbow" colourmap. Can also be accessed when writing custom colour map functions.
colourMap	Either the string "Solid" or "Rainbow" or a pointer to a custom colour map function. Custom colour maps must take the parameters colourMap(self, index, minIndex, maxIndex).
update	If update=True then a call to updateGraph1D() will, in turn, call glowbit.pixelsShow() to update the physical LEDs.

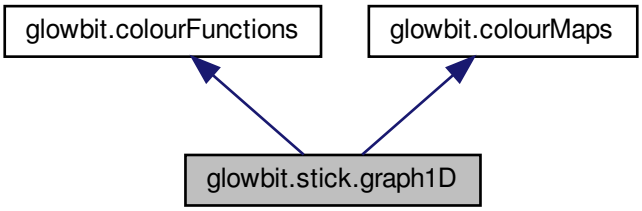
The documentation for this class was generated from the following file:

- glowbit.py

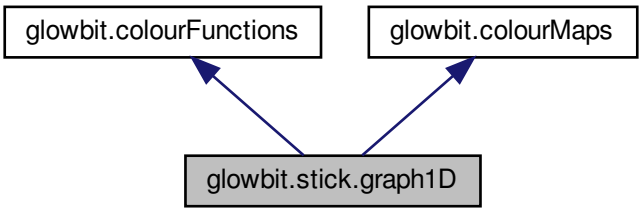
3.7 glowbit.stick.graph1D Class Reference

One dimensional graph oject for drawing a graph bar on a GlowBit Stick display.

Inheritance diagram for glowbit.stick.graph1D:



Collaboration diagram for glowbit.stick.graph1D:



Public Member Functions

- def __init__ (self, minIndex=0, maxIndex=7, minValue=0, maxValue=255, colour=0xFFFFFF, colourMap="↩ Solid", update=False)

  Initialisation routine for the glowbit.stick.graph1D object.

Public Attributes

- **minValue**
- **maxValue**
- **minIndex**
- **maxIndex**
- **m**
- **offset**
- **update**
- **colour**
- **colourMap**

3.7.1 Detailed Description

One dimensional graph oject for drawing a graph bar on a GlowBit Stick display.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 __init__()

```
def glowbit.stick.graph1D.__init__ (
              self,
              minIndex = 0,
              maxIndex = 7,
              minValue = 0,
              maxValue = 255,
              colour = 0xFFFFFF,
              colourMap = "Solid",
              update = False )
```

Initialisation routine for the glowbit.stick.graph1D object.

This object is drawn to the display with glowbit.stick.updateGraph1D.

Parameters

minIndex	The pixel index for the start of the graph
maxIndex	The pixel index for the end of the graph
minValue	The numerical value of the start of the graph
maxValue	The numerical value of the end of the graph
colour	The graph's colour if using the Solid colourmap
colourMap	Either the string "Solid" or "Rainbow" or a function pointer to a custom colour map. Custom colour maps must take the parameters colourMap(Self, index, minIndex, maxIndex).
update	If this is set to True then a call to updateGraph1D() will automatically call pixelsShow() to update the physical display.

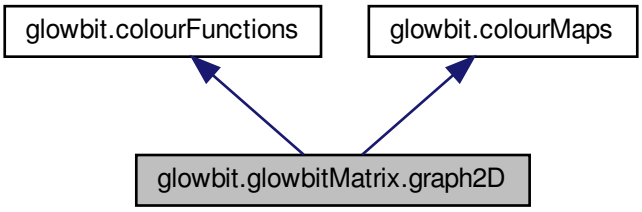
The documentation for this class was generated from the following file:

- glowbit.py

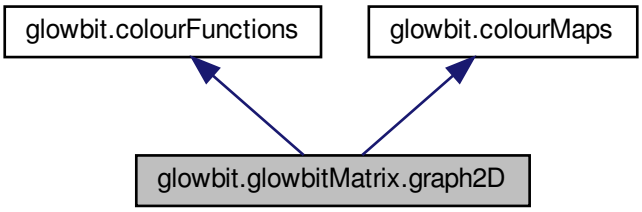
3.8 glowbit.glowbitMatrix.graph2D Class Reference

Object for drawing 2 dimensional time series graphs on GlowBit Matrix displays.

Inheritance diagram for glowbit.glowbitMatrix.graph2D:



Collaboration diagram for glowbit.glowbitMatrix.graph2D:



Public Member Functions

- def __init__ (self, originX=0, originY=7, width=8, height=8, minValue=0, maxValue=255, colour=0xFFFFFF, bgColour=0x000000, colourMap="Solid", update=False, bars=False)

  Initialisation routine for the glowbit.matrix.graph2D object.

Public Attributes

- **minValue**
- **maxValue**
- **originX**
- **originY**
- **width**
- **height**
- **colour**
- **bgColour**
- **update**
- **m**
- **offset**
- **bars**
- **data**
- **colourMap**

3.8.1 Detailed Description

Object for drawing 2 dimensional time series graphs on GlowBit Matrix displays.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 __init__()

```
def glowbit.glowbitMatrix.graph2D.__init__ (
                self,
                originX = 0,
                originY = 7,
                width = 8,
                height = 8,
                minValue = 0,
                maxValue = 255,
                colour = 0xFFFFFF,
                bgColour = 0x000000,
                colourMap = "Solid",
                update = False,
                bars = False )
```

Initialisation routine for the glowbit.matrix.graph2D object.

A graph2D object will be drawn to a rectangular region specified by the origin, width, and height.

This graph type is explicitly designed to draw time series data.

Parameters

originX	The x coordinate of the graph's origin (lower left corner).
originY	The y coordinate of the graph's origin (lower left corner).
width	The width, in pixels, of the graph's drawing area.
height	The height, in pixels, of the graph's drawing area
minValue	The value which will be mapped to the bottom edge.
maxValue	The value which will be mapped to the upper edge.
colour	A packed 32-bit GlowBit colour value. Used by the "Solid" colourmap, ignored by the "Rainbow" colourmap. Can also be accessed when writing custom colour map functions.
bgColour	A packed 32-bit GlowBit colour value which is drawn to the entire graph area prior to drawing the data.
colourMap	Either the string "Solid" or "Rainbow" or a pointer to a custom colour map function. Custom colour maps must take the parameters colourMap(self, index, minIndex, maxIndex).
update	If update=True then a call to updateGraph2D() will, in turn, call glowbit.pixelsShow() to update the physical LEDs.

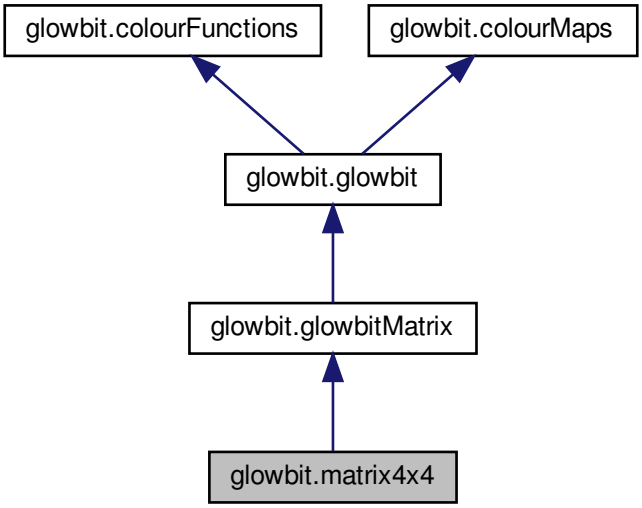
The documentation for this class was generated from the following file:

- glowbit.py

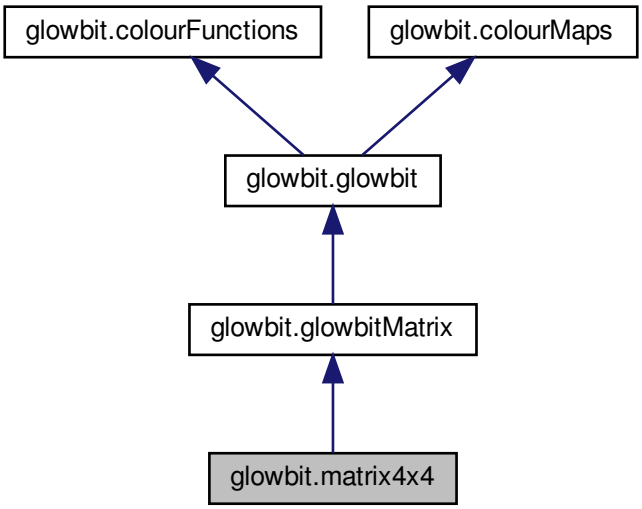
3.9 glowbit.matrix4x4 Class Reference

Class for driving GlowBit Matrix 4x4 modules and horizontally tiled arrangements thereof.

Inheritance diagram for glowbit.matrix4x4:



Collaboration diagram for glowbit.matrix4x4:



Public Member Functions

- def __init__ (self, tiles=1, pin=18, brightness=20, mapFunction=None, rateLimitFPS=30, sm=0)

  Initialisation routine for GlowBit stick modules and tiled arrays thereof.
- def remap4x4 (self, x, y)

  Maps an (x,y) coordinate on a 4 row, Nx4 column, tiled GlowBit Matrix 4x4 array to an array index for the internal buffer.

Public Attributes

- **sm**
- **pixelsShow**
- **ticks_ms**
- **tiles**
- **numLEDs**
- **numLEDsX**
- **numLEDsY**
- **numCols**
- **numRows**
- **strip**
- **ar**
- **dimmer_ar**
- **lastFrame_ms**
- **scrollingText**
- **brightness**
- **remap**
- **rateLimit**

Additional Inherited Members

3.9.1 Detailed Description

Class for driving GlowBit Matrix 4x4 modules and horizontally tiled arrangements thereof.

NB: The 4x4 matrix is designed to only tile horizontally, making an Nx4 pixel display.

If manually tiling horizontally and vertically a custom remapping function will need to be written.

The custom mapping function has the form mapFunction(self, x: int, y: int) -> int and returns a "one dimensional" pixel index given an (x,y) coordinate.

No checking is performed before calling the mapping function. An exception will be raised if the positional arguments are incorrect.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 __init__()

```
def glowbit.matrix4x4.__init__ (
              self,
              tiles = 1,
              pin = 18,
              brightness = 20,
              mapFunction = None,
              rateLimitFPS = 30,
              sm = 0 )
```

Initialisation routine for GlowBit stick modules and tiled arrays thereof.

Parameters

tiles	The number of tiled GlowBit Matrix 4x4 modules.
pin	The GPIO pin connected to the GlowBit stick module. Defaults to 18 as that pin is compatible with the Raspberry Pi and Raspberry Pi Pico. Any pin can be used on the Raspberry Pi Pico, only pins 18 and 12 are valid on the Raspberry Pi.
brightness	The relative brightness of the LEDs. Colours drawn to the internal buffer should be in the range [0,255] and the brightness parameter scales this value before drawing to the physical display. If brightness is an integer it should be in the range [0,255]. If brightness is floating point it is assumed to be in the range [0,1.0].
mapFunction	A function pointer to a custom pixel mapping function. Only required if mapping pixels to non-standard tiling arrangements.
rateLimitFPS	The maximum frame rate of the display in frames per second. The pixelsShow() function blocks to enforce this limit.
sm	(Raspberry Pi Pico only) The PIO state machine to generate the GlowBit data stream. Each connected GlowBit display chain requires a unique state machine. Valid values are in the range [0,7].

3.9.3 Member Function Documentation

3.9.3.1 remap4x4()

```
def glowbit.matrix4x4.remap4x4 (
              self,
              x,
              y )
```

Maps an (x,y) coordinate on a 4 row, Nx4 column, tiled GlowBit Matrix 4x4 array to an array index for the internal buffer.

It is recommended to use pixelSetXY() (and variants) instead of this function.

The return value can be passed to pixelSet(i, colour) (and its variants pixelSetNow() etc) in place of the paramter "i".

The (x,y) coordinates assume (0,0) in the upper left corner of the display with x increasing to the right and y increasing down

This function does not do boundary checking and may return a value which is outside the array, causing an IndexError exception to be raised.

Parameters

x	The x coordinate of the pixel to index
y	The y coordinate of the pixel to index

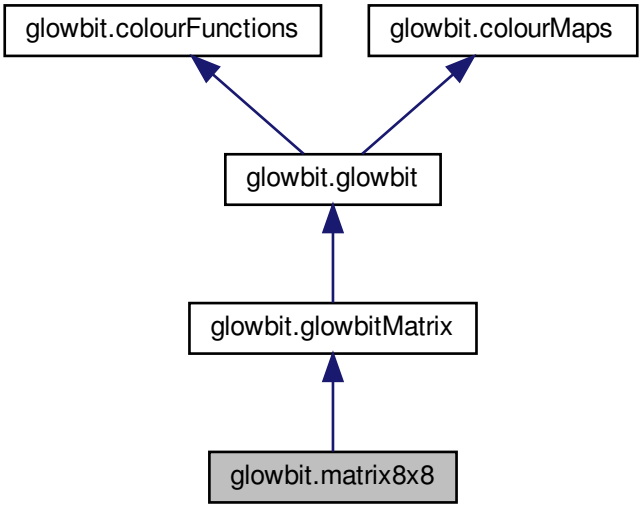
The documentation for this class was generated from the following file:

- glowbit.py

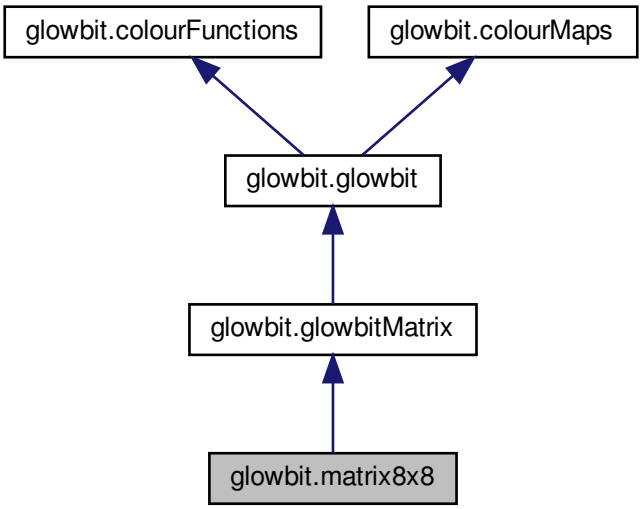
3.10 glowbit.matrix8x8 Class Reference

Class for driving GlowBit Matrix 8x8 modules and tiled arrangements thereof.

Inheritance diagram for glowbit.matrix8x8:



Collaboration diagram for glowbit.matrix8x8:



Classes

- class _textScroll

Public Member Functions

- def __init__ (self, tileRows=1, tileCols=1, pin=18, brightness=20, mapFunction=None, rateLimitFPS=-1, rateLimitCharactersPerSecond=-1, sm=0)

  Initialisation routine for GlowBit stick modules and tiled arrays thereof.
- def printTextWrap (self, string, x=0, y=0, colour=0xFFFFFF)

  Prints a string of text to a tiled GlowBit Matrix 8x8 display, automatically wrapping to new lines as required.
- def addTextScroll (self, string, y=0, x=0, colour=0xFFFFFF, bgColour=0x000000, update=False, blocking=↩ False)

  Adds a line of scrolling text to the display.
- def updateTextScroll (self)

  Update a scrolling text animation.
- def remap8x8 (self, x, y)

  Maps an (x,y) coordinate on a tiled GlowBit Matrix 8x8 array to an internal buffer array index.
- def drawChar (self, char, Px, Py, colour)

  Draw a single character to the display.
- def updateRateLimitCharactersPerSecond (self, rateLimitCharactersPerSecond)

  Changes the 8x8 matrix display's update rate in units of "characters of scrolling text per second".

Public Attributes

- **tileRows**
- **tileCols**
- **numLEDs**
- **numLEDsX**
- **numLEDsY**
- **numCols**
- **numRows**
- **sm**
- **pixelsShow**
- **ticks_ms**
- **strip**
- **ar**
- **dimmer_ar**
- **brightness**
- **scrollingText**
- **lastFrame_ms**
- **rateLimit**
- **scrollingTextList**
- **remap**
- **updateText**

Additional Inherited Members

3.10.1 Detailed Description

Class for driving GlowBit Matrix 8x8 modules and tiled arrangements thereof.

The GlowBit Matrix 8x8 is designed to tile in two dimensions to create arbitratily large displays without the need for "air-wiring" of the data signal.

The remap8x8() method maps an (x,y) coordinate to a pixel index if the data signal "snakes" back and forth. When viewed from the REAR of a tiled array data-in is soldered to the top right module, moves right to left, then left to right on the 2nd row, etc. See https://glowbit.io/02 for assembly details.

```
  ---<  Data routing: view from REAR (ie: when soldering Din / Dout pads).
 |
  >---
     |
  ---<
```

3.10.2 Constructor & Destructor Documentation

3.10.2.1 __init__()

```
def glowbit.matrix8x8.__init__ (
                self,
                tileRows = 1,
                tileCols = 1,
                pin = 18,
                brightness = 20,
                mapFunction = None,
                rateLimitFPS = -1,
                rateLimitCharactersPerSecond = -1,
                sm = 0 )
```

Initialisation routine for GlowBit stick modules and tiled arrays thereof.

Parameters

tileRows	The number of tiled GlowBit Matrix 8x8 module rows.
tileCols	The number of tiled GlowBit Matrix 8x8 module columns.
pin	The GPIO pin connected to the GlowBit stick module. Defaults to 18 as that pin is compatible with the Raspberry Pi and Raspberry Pi Pico. Any pin can be used on the Raspberry Pi Pico, only pins 18 and 12 are valid on the Raspberry Pi.
brightness	The relative brightness of the LEDs. Colours drawn to the internal buffer should be in the range [0,255] and the brightness parameter scales this value before drawing to the physical display. If brightness is an integer it should be in the range [0,255]. If brightness is floating point it is assumed to be in the range [0,1.0].
mapFunction	A function pointer to a custom pixel mapping function. Only required if mapping pixels to non-standard tiling arrangements.

Parameters

rateLimitFPS	The maximum frame rate of the display in frames per second. The pixelsShow() function blocks to enforce this limit. This argument defaults to -1 to allow rateLimitCharactersPerSecond to preference this parameter if it is not set. If neither rateLimitFPS or rateLimitCharactersPerSecond are set the limit is set to 30 FPS.
rateLimitCharactersPerSecond	If given a positive value the display update rate is set to display this many characters of scrolling text per second. A value of 1 is fast, but readable. This value can be fractional (eg: 0.5).
sm	(Raspberry Pi Pico only) The PIO state machine to generate the GlowBit data stream. Each connected GlowBit display chain requires a unique state machine. Valid values are in the range [0,7].

3.10.3 Member Function Documentation

3.10.3.1 addTextScroll()

```
def glowbit.matrix8x8.addTextScroll (
              self,
              string,
              y = 0,
              x = 0,
              colour = 0xFFFFFF,
              bgColour = 0x000000,
              update = False,
              blocking = False )
```

Adds a line of scrolling text to the display.

This method can be blocking or non-blocking.

If blocking the scrolling text will be drawn to the physical display and the method won't return until the animation is complete.

If non-blockig this method will return quickly, allowing subsequent calls to updateTextScroll() to control the rate of scrolling text animation. The text will scroll to the left one pixel with each call to updateTextScroll().

The update prameter is provided for convinience; if it is set to True a call to updateTextScroll() will automatically call pixelsShow(). Setting update to False allows the text scroll to be synchronised with other drawing updates.

Parameters

string	The string of text to scroll across the display
y	The y coordinate of the top edge of the text
x	The initial location of the text relative to the right edge of the display. Setting positive will scroll the text from further off the edge, producing a delay before it is visible. Setting negative will cause the text to appear on the display instantly before scrolling to the left. In units of pixels.
colour	The colour of the scrolling text characters. A 32-bit GlowBit colour value
bgColour	The colour of the background (ie: all pixels in the 8-row high area the text is drawn to which aren't part of a character). A 32-bit GlowBit colour value.
update	Passing update = True causes updateTextScroll() to call pixelsShow(). Otherwise pixelsShow() must be called manually, allowing synchronisation of scrolling text with other animated features.
blocking	Passing blocking = True will draw the scrolling text to the screen immediately and this method will not return until the text has scrolled off the display.

3.10.3.2 drawChar()

```
def glowbit.matrix8x8.drawChar (
              self,
              char,
              Px,
              Py,
              colour )
```

Draw a single character to the display.

For increased performance on Micropython boards this method uses the Micropython Viper code emitter so all arguments are necessary.

See also: addTextScroll() / updateTextScroll() for built-in scrolling text and printTextWrap() for printing static text with automatic line feeds.

Parameters

char	A single character string. This character will be drawn to the internal buffer.
Px	The x coordinate of the upper left corner of the character. Characters occupy an 8x8 pixel area.
Py	The y coordinate of the upper left corner of the character. Characters occupy an 8x8 pixel area.
colour	A 32-bit GlowBit colour value

3.10.3.3 printTextWrap()

```
def glowbit.matrix8x8.printTextWrap (
              self,
              string,
              x = 0,
              y = 0,
              colour = 0xFFFFFF )
```

Prints a string of text to a tiled GlowBit Matrix 8x8 display, automatically wrapping to new lines as required.

Each character occupies an 8x8 pixel area.

Characters which do not fit on the display are truncated.

Parameters

string	The string to print to the display.
x	The x coordinate of the upper left corner of the first character
y	The y coordinate of the upper left corner of the first character
colour	A 32-bit GlowBit colour value. All pixels in every character will be drawn in this colour.

3.10.3.4 remap8x8()

```
def glowbit.matrix8x8.remap8x8 (
              self,
              x,
              y )
```

Maps an (x,y) coordinate on a tiled GlowBit Matrix 8x8 array to an internal buffer array index.

It is recommended to use pixelSetXY() (and variants) instead of this function.

The return value can be passed to pixelSet(i, colour) (and its variants pixelSetNow() etc) in place of the paramter "i".

The (x,y) coordinates assume (0,0) in the upper left corner of the display with x increasing to the right and y increasing down

This function does not do boundary checking and may return a value which is outside the array, causing an IndexError exception to be raised.

Parameters

x	The x coordinate of the pixel to index
y	The y coordinate of the pixel to index

3.10.3.5 updateRateLimitCharactersPerSecond()

```
def glowbit.matrix8x8.updateRateLimitCharactersPerSecond (
              self,
              rateLimitCharactersPerSecond )
```

Changes the 8x8 matrix display's update rate in units of "characters of scrolling text per second".

For example, a value of 2 would scroll 2 charcters per second; leaving each character at least partly visible for 0.5 seconds.

3.10.3.6 updateTextScroll()

```
def glowbit.matrix8x8.updateTextScroll (
              self )
```

Update a scrolling text animation.

addTextScroll() must be called at least once for scrolling text to be drawn to the display.

The documentation for this class was generated from the following file:

- glowbit.py

3.11 glowbit.micropython Class Reference

Public Member Functions

- def **viper** (func)

The documentation for this class was generated from the following file:

- glowbit.py

3.12 glowbit.rp2.PIO Class Reference

Static Public Attributes

- **OUT_LOW** = None
- **SHIFT_LEFT** = None

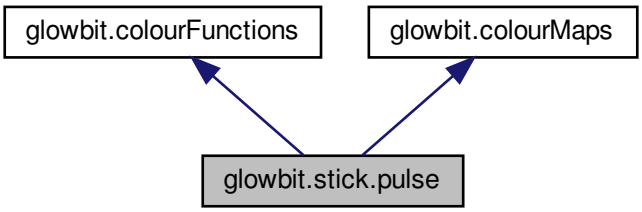
The documentation for this class was generated from the following file:

- glowbit.py

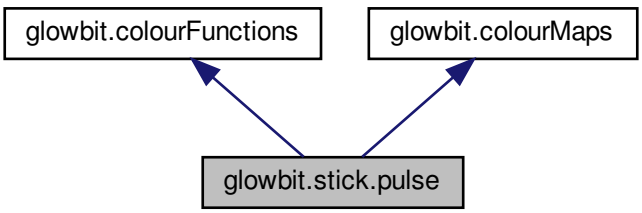
3.13 glowbit.stick.pulse Class Reference

A class for animating "pulses" which move down a GlowBit stick.

Inheritance diagram for glowbit.stick.pulse:



Collaboration diagram for glowbit.stick.pulse:



Public Member Functions

- def __init__ (self, speed=100, colour=[0xFFFFFF], index=0, colourMap=None)

  Initialisation routine for the GlowBit Stick pulse object.

Public Attributes

- speed

  Speed of the pulse.
- index

  Initial index of the pulse.
- colour

  A list of 32-bit GlowBit colour values.
- colourMap

  Either the string "Solid" or "Rainbow" or a function pointer to a custom colourmap.

3.13.1 Detailed Description

A class for animating "pulses" which move down a GlowBit stick.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 __init__()

```
def glowbit.stick.pulse.__init__ (
              self,
              speed = 100,
              colour = [0xFFFFFF],
              index = 0,
              colourMap = None )
```

Initialisation routine for the GlowBit Stick pulse object.

This function uses the pixelSaturatingAdd() method so multiple pulses can be drawn without colour values corrupting due to addition overflow.

Parameters

speed	The speed of the pulse in units of (pixels moved per frame) $*$ 100. A value of 100 means the pulse will move 1 pixels per frame. A speed of 1 will move a pulse 1 pixel every 100 frames. Speed can be positive or negative to allow pulses to move in either direction.
colour	A list of 32-bit GlowBit colours for the pulse. The pulse will have a width equal to the number of elements in this list. A list entry of -1 will have the colour set by a colour map function.
index	The initial index of the pulse. Generally recommended to set to 0 if speed $>$ 0 and numLEDs if speed $<$ 0.
colourMap	Either the string "Solid" or "Rainbow" or a custom function pointer. Custom functions must take the positional arguments: colourMapFunction(self, index, minIndex, maxIndex). When calling colour map functions updatePulses() sets minIndex to 0 and maxIndex to numLEDs.

3.13.3 Member Data Documentation

3.13.3.1 colour

`glowbit.stick.pulse.colour`

A list of 32-bit GlowBit colour values.

Each one is drawn to a pixel; a -1 indicates the use of the colourMap function

3.13.3.2 colourMap

`glowbit.stick.pulse.colourMap`

Either the string "Solid" or "Rainbow" or a function pointer to a custom colourmap.

Only sets pixel colour for pixels with a colour of -1.

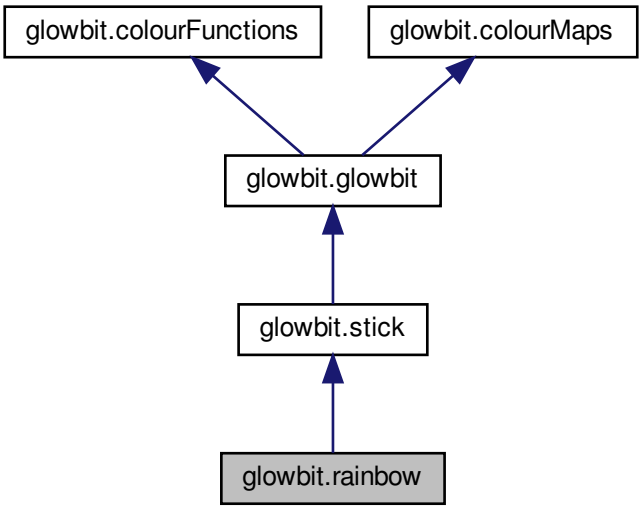
The documentation for this class was generated from the following file:

- glowbit.py

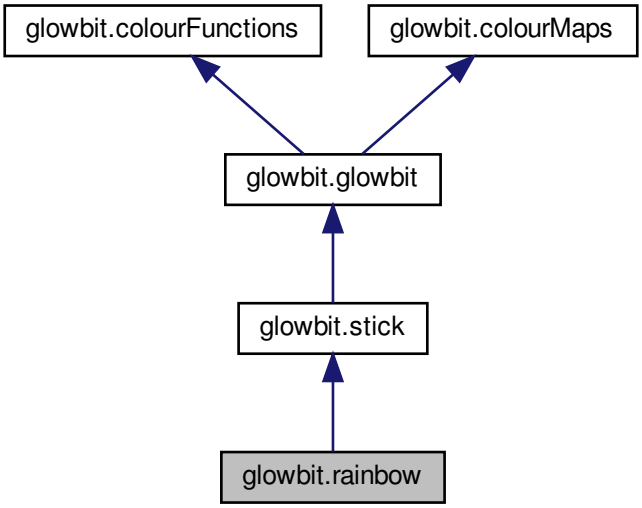
3.14 glowbit.rainbow Class Reference

The class specific to the GlowBit Rainbow.

Inheritance diagram for glowbit.rainbow:



Collaboration diagram for glowbit.rainbow:



Public Member Functions

- def __init__ (self, numLEDs=13, pin=18, brightness=40, rateLimitFPS=60, sm=0)

  Initialisation routine for GlowBit rainbow modules.
- def pixelSetAngle (self, angle, colour)

  Sets the colour of a pixel on the GlowBit Rainbow, addressed by its angle label.
- def drawRainbow (self, offset=0)

  Colours each pixel to display a rainbow spectrum.
- def demo (self)

  Displays a rainbow animation in an infinite loop.

Additional Inherited Members

3.14.1 Detailed Description

The class specific to the GlowBit Rainbow.

This class inherits all the functionality of the GlowBit Stick and extends it with Rainbow-specific methods.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 __init__()

```
def glowbit.rainbow.__init__ (
              self,
              numLEDs = 13,
              pin = 18,
              brightness = 40,
              rateLimitFPS = 60,
              sm = 0 )
```

Initialisation routine for GlowBit rainbow modules.

Parameters

numLEDs	The total number of LEDs. Should be set to 13 ∗ (the number of tiled modules).
pin	The GPIO pin connected to the GlowBit Rainbow module. Defaults to 18 as that pin is compatible with the Raspberry Pi and Raspberry Pi Pico. Any pin can be used on the Raspberry Pi Pico, only pins 18 and 12 are valid on the Raspberry Pi.
brightness	The relative brightness of the LEDs. Colours drawn to the internal buffer should be in the range [0,255] and the brightness parameter scales this value before drawing to the physical display. If brightness is an integer it should be in the range [0,255]. If brightness is floating point it is assumed to be in the range [0,1.0].
rateLimitFPS	The maximum frame rate of the display in frames per second. The pixelsShow() function blocks to enforce this limit.

3.14.3 Member Function Documentation

3.14.3.1 demo()

```
def glowbit.rainbow.demo (
              self )
```

Displays a rainbow animation in an infinite loop.

This method demonstrates the use of drawRainbow().

3.14.3.2 drawRainbow()

```
def glowbit.rainbow.drawRainbow (
              self,
              offset = 0 )
```

Colours each pixel to display a rainbow spectrum.

This method calls pixelsShow().

Parameters

offset	A "phase" offset mapping [0,360] degrees to [0,255]. A value of 0 displays red at angle 0 and purple at angle 180. A modulo-255 operation is performed, allowing this value to be any integer. The rainLoop() method varies this value to display an animation.

3.14.3.3 pixelSetAngle()

```
def glowbit.rainbow.pixelSetAngle (
              self,
              angle,
              colour )
```

Sets the colour of a pixel on the GlowBit Rainbow, addressed by its angle label.

Parameters

angle	An integer number in degrees equal to an angle label on the GlowBit Rainbow PCB.
colour	A 32-bit GlowBit colour value

The documentation for this class was generated from the following file:

- glowbit.py

3.15 glowbit.glowbitMatrix.raindrop Class Reference

A class used by the rain() demonstration.

Public Member Functions

- def **__init__** (self, x, speed)
- def **update** (self)
- def **getY** (self)

Public Attributes

- **x**
- **speed**
- **y**

3.15.1 Detailed Description

A class used by the rain() demonstration.

The documentation for this class was generated from the following file:

- glowbit.py

3.16 glowbit.rp2 Class Reference

Classes

- class PIO

Public Member Functions

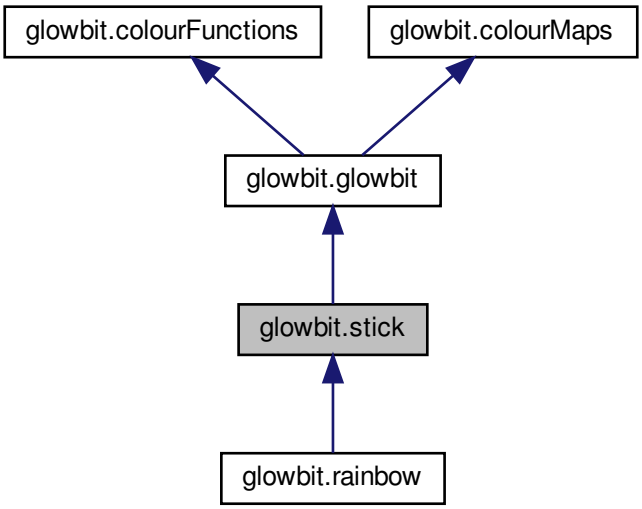
- def **asm_pio** (sideset_init, out_shiftdir, autopull, pull_thresh)

The documentation for this class was generated from the following file:

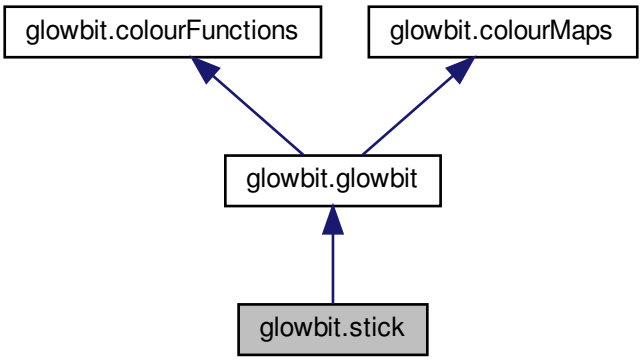
- glowbit.py

3.17 glowbit.stick Class Reference

Inheritance diagram for glowbit.stick:



Collaboration diagram for glowbit.stick:



Classes

- class graph1D

  One dimensional graph ofject for drawing a graph bar on a GlowBit Stick display.
- class pulse

  A class for animating "pulses" which move down a GlowBit stick.

Public Member Functions

- def __init__ (self, numLEDs=8, pin=18, brightness=20, rateLimitFPS=30, sm=0)

  Initialisation routine for GlowBit stick modules and tiled arrays thereof.
- def addPulse (self, speed=100, colour=[0xFFFFFF], index=0, colourMap=None)

  Add a pulse to the list of pulses.
- def updatePulses (self)

  Update the position of all pulses in self.pulses[] and draw them to the internal buffer.
- def newGraph1D (self, minIndex=0, maxIndex=7, minValue=0, maxValue=255, colour=0xFFFFFF, colour↩ Map="Solid", update=False)

  Wrapper function to create graph1D objects.
- def updateGraph1D (self, graph, value)

  Updates a graph1D object, drawing it to the display.
- def fillSlice (self, i=0, j=-1, colour=0xFFFFFF)

  Fill a "slice" of the GlowBit stick's pixels with a solid colour.
- def pulseDemo (self, iters=100)

  A demonstration of the use of "pulse" objects.
- def graphDemo (self, iters=3)

  A demonstration of the use of "graph1D" objects.
- def sliceDemo (self)

  A Demonstration of the use of the "fillSlice" method.
- def rainbowDemo (self, iters=5)

  Uses the colourMapRainbow() colour map to display a colourful animation.
- def demo (self)

  Runs several demo patterns.

Public Attributes

- **sm**
- **pixelsShow**
- **ticks_ms**
- **numLEDs**
- **strip**
- **lastFrame_ms**
- **ar**
- **dimmer_ar**
- **rateLimit**
- **brightness**
- **pulses**

Additional Inherited Members

3.17.1 Constructor & Destructor Documentation

3.17.1.1 __init__()

```
def glowbit.stick.__init__ (
              self,
              numLEDs = 8,
              pin = 18,
              brightness = 20,
              rateLimitFPS = 30,
              sm = 0 )
```

Initialisation routine for GlowBit stick modules and tiled arrays thereof.

Parameters

numLEDs	The total number of LEDs. Should be set to 8 ∗ (the number of tiled modules).
pin	The GPIO pin connected to the GlowBit stick module. Defaults to 18 as that pin is compatible with the Raspberry Pi and Raspberry Pi Pico. Any pin can be used on the Raspberry Pi Pico, only pins 18 and 12 are valid on the Raspberry Pi.
brightness	The relative brightness of the LEDs. Colours drawn to the internal buffer should be in the range [0,255] and the brightness parameter scales this value before drawing to the physical display. If brightness is an integer it should be in the range [0,255]. If brightness is floating point it is assumed to be in the range [0,1.0].
rateLimitFPS	The maximum frame rate of the display in frames per second. The pixelsShow() function blocks to enforce this limit.
sm	(Raspberry Pi Pico only) The PIO state machine to generate the GlowBit data stream. Each connected GlowBit display chain requires a unique state machine. Valid values are in the range [0,7].

3.17.2 Member Function Documentation

3.17.2.1 addPulse()

```
def glowbit.stick.addPulse (
              self,
              speed = 100,
              colour = [0xFFFFFF],
              index = 0,
              colourMap = None )
```

Add a pulse to the list of pulses.

Parameters

speed	The speed of the pulse in units of (pixels moved per frame) $*$ 100. A value of 100 means the pulse will move 1 pixels per frame. A speed of 1 will move a pulse 1 pixel every 100 frames. Speed can be positive or negative to allow pulses to move in either direction.
colour	A list of 32-bit GlowBit colours for the pulse. The pulse will have a width equal to the number of elements in this list. A list entry of -1 will have the colour set by a colour map function.
index	The initial index of the pulse. Generally recommended to set to 0 if speed $>$ 0 and numLEDs if speed $<$ 0.
colourMap	Either the string "Solid" or "Rainbow" or a custom function pointer. Custom functions must take the positional arguments: colourMapFunction(self, index, minIndex, maxIndex). When calling colour map functions updatePulses() sets minIndex to 0 and maxIndex to numLEDs.

3.17.2.2 fillSlice()

```
def glowbit.stick.fillSlice (
              self,
              i = 0,
              j = -1,
              colour = 0xFFFFFF )
```

Fill a "slice" of the GlowBit stick's pixels with a solid colour.

By default it will fill the entire display with a solid colour.

Parameters

i	The minimum index to fill
j	The maximum index to fill
colour	A 32-bit GlowBit colour value

3.17.2.3 graphDemo()

```
def glowbit.stick.graphDemo (
              self,
              iters = 3 )
```

A demonstration of the use of "graph1D" objects.

This demonstration alternates between drawing two graphs with different colour maps; one with the "Rainbow" map, covering the full colour wheel, and another of solid white.

Parameters

iters	The number of times both graphs are drawn.

3.17.2.4 newGraph1D()

```
def glowbit.stick.newGraph1D (
              self,
              minIndex = 0,
              maxIndex = 7,
              minValue = 0,
              maxValue = 255,
              colour = 0xFFFFFF,
              colourMap = "Solid",
              update = False )
```

Wrapper function to create graph1D objects.

Returns a new stick.graph1D() object.

See also stick.graph1D()

Parameters

minIndex	The pixel index for the start of the graph
maxIndex	The pixel index for the end of the graph
minValue	The numerical value of the start of the graph
maxValue	The numerical value of the end of the graph
colour	The graph's colour if using the Solid colourmap
colourMap	Either the string "Solid" or "Rainbow" or a function pointer to a custom colour map. Custom colour maps must take the parameters colourMap(Self, index, minIndex, maxIndex).
update	If this is set to True then a call to updateGraph1D() will automatically call pixelsShow() to update the physical display.

3.17.2.5 pulseDemo()

```
def glowbit.stick.pulseDemo (
              self,
              iters = 100 )
```

A demonstration of the use of "pulse" objects.

The pulse traveling "up" the stick is drawn with default arguments: a single white pixel

The pulse returning "down" the stick is drawn with a 3-pixel list of colours. The first and last are coloured with the "Rainbow" colour map, changing colour with pixel index, while the middle is white.

Parameters

iters	The number of frames which are drawn before returning.

3.17.2.6 sliceDemo()

```
def glowbit.stick.sliceDemo (
              self )
```

A Demonstration of the use of the "fillSlice" method.

Animates a red, green, and blue slice "moving" down the GlowBit Stick display.

The number of iterations is fixed due to the bit shift operation being used to change colour.

3.17.2.7 updateGraph1D()

```
def glowbit.stick.updateGraph1D (
              self,
              graph,
              value )
```

Updates a graph1D object, drawing it to the display.

If the graph1D object was created with "update = True" this function will call pixelsShow() to update the physical display before returning.

Parameters

graph	A graph1D object as returned by stick.graph1D
value	The numerical value to plot on the graph

3.17.2.8 updatePulses()

```
def glowbit.stick.updatePulses (
              self )
```

Update the position of all pulses in self.pulses[] and draw them to the internal buffer.

A call to pixelsShow() must be done manually to update the physical LEDs.

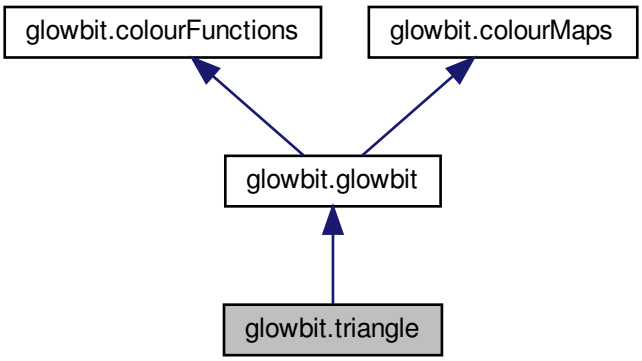
The documentation for this class was generated from the following file:

- glowbit.py

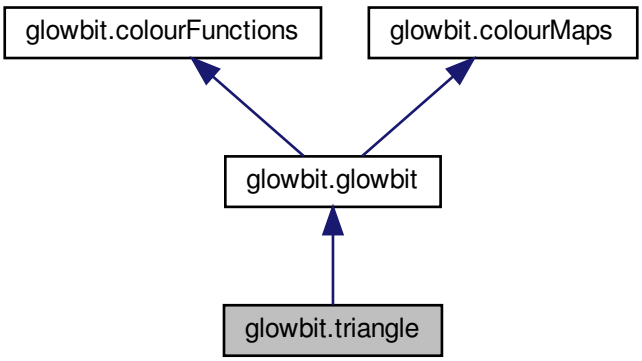
3.18 glowbit.triangle Class Reference

Class for driving triangular GlowBit modules.

Inheritance diagram for glowbit.triangle:



Collaboration diagram for glowbit.triangle:



Public Member Functions

- def __init__ (self, numTris=1, LEDsPerTri=6, pin=18, brightness=20, rateLimitFPS=20, sm=0)

  Initialisation routine for triangular GlowBit modules and tiled arrays thereof.
- def fillTri (self, tri, colour)

  Fills all LEDs on a given triangle with the same colour.
- def demo (self)

  Displays a simple demo pattern.

Public Attributes

- **sm**
- **pixelsShow**
- **ticks_ms**
- **LEDsPerTri**
- **numLEDs**
- **numTris**
- **strip**
- **ar**
- **dimmer_ar**
- **rateLimit**
- **brightness**
- **lastFrame_ms**

Additional Inherited Members

3.18.1 Detailed Description

Class for driving triangular GlowBit modules.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 __init__()

```
def glowbit.triangle.__init__ (
              self,
              numTris = 1,
              LEDsPerTri = 6,
              pin = 18,
              brightness = 20,
              rateLimitFPS = 20,
              sm = 0 )
```

Initialisation routine for triangular GlowBit modules and tiled arrays thereof.

Parameters

numTris	The number of triangle modules in the tiled array.
LEDsPerTri	The number of LEDs on each triangular module.
pin	The GPIO pin connected to the GlowBit stick module. Defaults to 18 as that pin is compatible with the Raspberry Pi and Raspberry Pi Pico. Any pin can be used on the Raspberry Pi Pico, only pins 18 and 12 are valid on the Raspberry Pi.
brightness	The relative brightness of the LEDs. Colours drawn to the internal buffer should be in the range [0,255] and the brightness parameter scales this value before drawing to the physical display. If brightness is an integer it should be in the range [0,255]. If brightness is floating point it is assumed to be in the range [0,1.0].
rateLimitFPS	The maximum frame rate of the display in frames per second. The pixelsShow() function blocks to enforce this limit.
sm	(Raspberry Pi Pico only) The PIO state machine to generate the GlowBit data stream. Each connected GlowBit display chain requires a unique state machine. Valid values are in the range [0,7].

3.18.3 Member Function Documentation

3.18.3.1 fillTri()

```
def glowbit.triangle.fillTri (
              self,
              tri,
              colour )
```

Fills all LEDs on a given triangle with the same colour.

Parameters

tri	The triangle to fill. The first triangle is addressed with 0.
colour	A 32-bit GlowBit colour value

The documentation for this class was generated from the following file:

- glowbit.py

Index