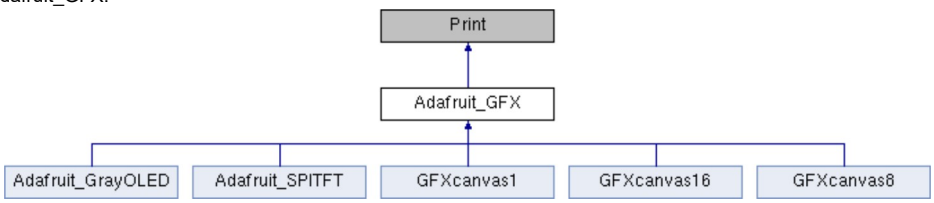
Adafruit_GFX Class Reference abstract

`#include <Adafruit_GFX.h>`

Inheritance diagram for Adafruit_GFX:



Public Member Functions

	Adafruit_GFX (int16_t w, int16_t h) Instatiate a GFX context for graphics! Can only be done by a superclass. More...
virtual void	**drawPixel** (int16_t x, int16_t y, uint16_t color)=0 Draw to the screen/framebuffer/etc. Must be overridden in subclass. More...
virtual void	**startWrite** (void) Start a display-writing routine, overwrite in subclasses.
virtual void	**writePixel** (int16_t x, int16_t y, uint16_t color) Write a pixel, overwrite in subclasses if startWrite is defined! More...
virtual void	**writeFillRect** (int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color) Write a rectangle completely with one color, overwrite in subclasses if startWrite is defined! More...
virtual void	**writeFastVLine** (int16_t x, int16_t y, int16_t h, uint16_t color) Write a perfectly vertical line, overwrite in subclasses if startWrite is defined! More...
virtual void	**writeFastHLine** (int16_t x, int16_t y, int16_t w, uint16_t color) Write a perfectly horizontal line, overwrite in subclasses if startWrite is defined! More...
virtual void	**writeLine** (int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color) Write a line. Bresenham's algorithm - thx wikpedia. More...
virtual void	**endWrite** (void) End a display-writing routine, overwrite in subclasses if startWrite is defined!
virtual void	**setRotation** (uint8_t r) Set rotation setting for display. More...
virtual void	**invertDisplay** (bool i) Invert the display (ideally using built-in hardware command) More...
virtual void	**drawFastVLine** (int16_t x, int16_t y, int16_t h, uint16_t color) Draw a perfectly vertical line (this is often optimized in a subclass!) More...
virtual void	**drawFastHLine** (int16_t x, int16_t y, int16_t w, uint16_t color) Draw a perfectly horizontal line (this is often optimized in a subclass!) More...
virtual void	**fillRect** (int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color) Fill a rectangle completely with one color. Update in subclasses if desired! More...
virtual void	**fillScreen** (uint16_t color) Fill the screen completely with one color. Update in subclasses if desired! More...
virtual void	**drawLine** (int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color) Draw a line. More...
virtual void	**drawRect** (int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color) Draw a rectangle with no fill color. More...
void	**drawCircle** (int16_t x0, int16_t y0, int16_t r, uint16_t color) Draw a circle outline. More...
void	**drawCircleHelper** (int16_t x0, int16_t y0, int16_t r, uint8_t cornername, uint16_t color) Quarter-circle drawer, used to do circles and roundrects. More...
void	**fillCircle** (int16_t x0, int16_t y0, int16_t r, uint16_t color) Draw a circle with filled color. More...
void	**fillCircleHelper** (int16_t x0, int16_t y0, int16_t r, uint8_t cornername, int16_t delta, uint16_t color) Quarter-circle drawer with fill, used for circles and roundrects. More...
void	**drawTriangle** (int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2, int16_t y2, uint16_t color) Draw a triangle with no fill color. More...
void	**fillTriangle** (int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2, int16_t y2, uint16_t color) Draw a triangle with color-fill. More...

void **drawRoundRect** (int16_t x0, int16_t y0, int16_t w, int16_t h, int16_t radius, uint16_t color)
Draw a rounded rectangle with no fill color. More...

void **fillRoundRect** (int16_t x0, int16_t y0, int16_t w, int16_t h, int16_t radius, uint16_t color)
Draw a rounded rectangle with fill color. More...

void **drawBitmap** (int16_t x, int16_t y, const uint8_t bitmap[], int16_t w, int16_t h, uint16_t color)
Draw a PROGMEM-resident 1-bit image at the specified (x,y) position, using the specified foreground color (unset bits are transparent). More...

void **drawBitmap** (int16_t x, int16_t y, const uint8_t bitmap[], int16_t w, int16_t h, uint16_t color, uint16_t bg)
Draw a PROGMEM-resident 1-bit image at the specified (x,y) position, using the specified foreground (for set bits) and background (unset bits) colors. More...

void **drawBitmap** (int16_t x, int16_t y, uint8_t *bitmap, int16_t w, int16_t h, uint16_t color)
Draw a RAM-resident 1-bit image at the specified (x,y) position, using the specified foreground color (unset bits are transparent). More...

void **drawBitmap** (int16_t x, int16_t y, uint8_t *bitmap, int16_t w, int16_t h, uint16_t color, uint16_t bg)
Draw a RAM-resident 1-bit image at the specified (x,y) position, using the specified foreground (for set bits) and background (unset bits) colors. More...

void **drawXBitmap** (int16_t x, int16_t y, const uint8_t bitmap[], int16_t w, int16_t h, uint16_t color)
Draw PROGMEM-resident XBitMap Files (*.xbm), exported from GIMP. Usage: Export from GIMP to *.xbm, rename *.xbm to *.c and open in editor. C Array can be directly used with this function. There is no RAM-resident version of this function; if generating bitmaps in RAM, use the format defined by **drawBitmap()** and call that instead. More...

void **drawGrayscaleBitmap** (int16_t x, int16_t y, const uint8_t bitmap[], int16_t w, int16_t h)
Draw a PROGMEM-resident 8-bit image (grayscale) at the specified (x,y) pos. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed. More...

void **drawGrayscaleBitmap** (int16_t x, int16_t y, uint8_t *bitmap, int16_t w, int16_t h)
Draw a RAM-resident 8-bit image (grayscale) at the specified (x,y) pos. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed. More...

void **drawGrayscaleBitmap** (int16_t x, int16_t y, const uint8_t bitmap[], const uint8_t mask[], int16_t w, int16_t h)
Draw a PROGMEM-resident 8-bit image (grayscale) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (grayscale and mask) must be PROGMEM-resident. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed. More...

void **drawGrayscaleBitmap** (int16_t x, int16_t y, uint8_t *bitmap, uint8_t *mask, int16_t w, int16_t h)
Draw a RAM-resident 8-bit image (grayscale) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (grayscale and mask) must be RAM-residentt, no mix-and-match Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed. More...

void **drawRGBBitmap** (int16_t x, int16_t y, const uint16_t bitmap[], int16_t w, int16_t h)
Draw a PROGMEM-resident 16-bit image (RGB 5/6/5) at the specified (x,y) position. For 16-bit display devices; no color reduction performed. More...

void **drawRGBBitmap** (int16_t x, int16_t y, uint16_t *bitmap, int16_t w, int16_t h)
Draw a RAM-resident 16-bit image (RGB 5/6/5) at the specified (x,y) position. For 16-bit display devices; no color reduction performed. More...

void **drawRGBBitmap** (int16_t x, int16_t y, const uint16_t bitmap[], const uint8_t mask[], int16_t w, int16_t h)
Draw a PROGMEM-resident 16-bit image (RGB 5/6/5) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (color and mask) must be PROGMEM-resident. For 16-bit display devices; no color reduction performed. More...

void **drawRGBBitmap** (int16_t x, int16_t y, uint16_t *bitmap, uint8_t *mask, int16_t w, int16_t h)
Draw a RAM-resident 16-bit image (RGB 5/6/5) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (color and mask) must be RAM-resident. For 16-bit display devices; no color reduction performed. More...

void **drawChar** (int16_t x, int16_t y, unsigned char c, uint16_t color, uint16_t bg, uint8_t size)
Draw a single character. More...

void **drawChar** (int16_t x, int16_t y, unsigned char c, uint16_t color, uint16_t bg, uint8_t size_x, uint8_t size_y)
Draw a single character. More...

void **getTextBounds** (const char *string, int16_t x, int16_t y, int16_t *x1, int16_t *y1, uint16_t *w, uint16_t *h)
Helper to determine size of a string with current font/size. Pass string and a cursor position, returns UL corner and W,H. More...

void **getTextBounds** (const __FlashStringHelper *s, int16_t x, int16_t y, int16_t *x1, int16_t *y1, uint16_t *w, uint16_t *h)
Helper to determine size of a PROGMEM string with current font/size. Pass string and a cursor position, returns UL corner and W,H. More...

void **getTextBounds** (const String &str, int16_t x, int16_t y, int16_t *x1, int16_t *y1, uint16_t *w, uint16_t *h)
Helper to determine size of a string with current font/size. Pass string and a cursor position, returns UL corner and W,H. More...

void **setTextSize** (uint8_t s)
Set text 'magnification' size. Each increase in s makes 1 pixel that much bigger. More...

void **setTextSize** (uint8_t sx, uint8_t sy)
Set text 'magnification' size. Each increase in s makes 1 pixel that much bigger. More...

void **setFont** (const **GFXfont** *f=NULL)
Set the font to display when print()ing, either custom or default. More...

void **setCursor** (int16_t x, int16_t y)

Set text cursor location. More...

void **setTextColor** (uint16_t c)
Set text font color with transparant background. More...

void **setTextColor** (uint16_t c, uint16_t bg)
Set text font color with custom background color. More...

void **setTextWrap** (bool w)
Set whether text that is too long for the screen width should automatically wrap around to the next line (else clip right). More...

void **cp437** (bool x=true)
Enable (or disable) Code Page 437-compatible charset. There was an error in glcdfont.c for the longest time – one character (#176, the 'light shade' block) was missing – this threw off the index of every character that followed it. But a TON of code has been written with the erroneous character indices. By default, the library uses the original 'wrong' behavior and old sketches will still work. Pass 'true' to this function to use correct CP437 character values in your code. More...

virtual void **write** (uint8_t)
Print one byte/character of data, used to support print() More...

int16_t **width** (void) const
Get width of the display, accounting for current rotation. More...

int16_t **height** (void) const
Get height of the display, accounting for current rotation. More...

uint8_t **getRotation** (void) const
Get rotation setting for display. More...

int16_t **getCursorX** (void) const
Get text cursor X location. More...

int16_t **getCursorY** (void) const
Get text cursor Y location. More...

Protected Member Functions

void **charBounds** (unsigned char c, int16_t *x, int16_t *y, int16_t *minx, int16_t *miny, int16_t *maxx, int16_t *maxy)
Helper to determine size of a character with current font/size. Broke this out as it's used by both the PROGMEM- and RAM-resident **getTextBounds()** functions. More...

Protected Attributes

int16_t **WIDTH**
This is the 'raw' display width - never changes.

int16_t **HEIGHT**
This is the 'raw' display height - never changes.

int16_t **_width**
Display width as modified by current rotation.

int16_t **_height**
Display height as modified by current rotation.

int16_t **cursor_x**
x location to start print()ing text

int16_t **cursor_y**
y location to start print()ing text

uint16_t **textcolor**
16-bit background color for print()

uint16_t **textbgcolor**
16-bit text color for print()

uint8_t **textsize_x**
Desired magnification in X-axis of text to print()

uint8_t **textsize_y**
Desired magnification in Y-axis of text to print()

uint8_t **rotation**
Display rotation (0 thru 3)

bool **wrap**
If set, 'wrap' text at right edge of display.

bool **_cp437**
If set, use correct CP437 charset (default is off)

GFXfont * **gfxFont**
Pointer to special font.

Detailed Description

A generic graphics superclass that can handle all sorts of drawing. At a minimum you can subclass and provide **drawPixel()**. At a maximum you can do a ton of overriding to optimize. Used for any/all Adafruit displays!

Constructor & Destructor Documentation

◆ Adafruit_GFX()

```
Adafruit_GFX::Adafruit_GFX ( int16_t  w,
                             int16_t  h
                           )
```

Instatiate a GFX context for graphics! Can only be done by a superclass.

Parameters

w Display width, in pixels

h Display height, in pixels

Member Function Documentation

◆ drawPixel()

```
virtual void Adafruit_GFX::drawPixel ( int16_t    x,
                                       int16_t    y,
                                       uint16_t  color
                                     )                                    pure virtual
```

Draw to the screen/framebuffer/etc. Must be overridden in subclass.

Parameters

x X coordinate in pixels

y Y coordinate in pixels

color 16-bit pixel color.

Implemented in **GFXcanvas16**, **GFXcanvas8**, **GFXcanvas1**, **Adafruit_SPITFT**, and **Adafruit_GrayOLED**.

◆ writePixel()

```
void Adafruit_GFX::writePixel ( int16_t    x,
                                int16_t    y,
                                uint16_t  color
                              )                                           virtual
```

Write a pixel, overwrite in subclasses if startWrite is defined!

Parameters

x x coordinate

y y coordinate

color 16-bit 5-6-5 Color to fill with

Reimplemented in **Adafruit_SPITFT**.

◆ writeFillRect()

```
void Adafruit_GFX::writeFillRect ( int16_t  x,
                                   int16_t  y,
                                   int16_t  w,
                                   int16_t  h,
                                   uint16_t color
                                 )                                  virtual
```

Write a rectangle completely with one color, overwrite in subclasses if startWrite is defined!

Parameters

- **x** Top left corner x coordinate
- **y** Top left corner y coordinate
- **w** Width in pixels
- **h** Height in pixels
- **color** 16-bit 5-6-5 Color to fill with

Reimplemented in **Adafruit_SPITFT**.

◆ writeFastVLine()

```
void Adafruit_GFX::writeFastVLine ( int16_t  x,
                                    int16_t  y,
                                    int16_t  h,
                                    uint16_t color
                                  )                                 virtual
```

Write a perfectly vertical line, overwrite in subclasses if startWrite is defined!

Parameters

- **x** Top-most x coordinate
- **y** Top-most y coordinate
- **h** Height in pixels
- **color** 16-bit 5-6-5 Color to fill with

Reimplemented in **Adafruit_SPITFT**.

◆ writeFastHLine()

```
void Adafruit_GFX::writeFastHLine ( int16_t  x,
                                    int16_t  y,
                                    int16_t  w,
                                    uint16_t color
                                  )                                 virtual
```

Write a perfectly horizontal line, overwrite in subclasses if startWrite is defined!

Parameters

- **x** Left-most x coordinate
- **y** Left-most y coordinate
- **w** Width in pixels
- **color** 16-bit 5-6-5 Color to fill with

Reimplemented in **Adafruit_SPITFT**.

◆ writeLine()

```
void Adafruit_GFX::writeLine ( int16_t  x0,
                               int16_t  y0,
                               int16_t  x1,
                               int16_t  y1,
                               uint16_t color
                             )                                    virtual
```

Write a line. Bresenham's algorithm - thx wikpedia.

Parameters

- **x0** Start point x coordinate
- **y0** Start point y coordinate
- **x1** End point x coordinate
- **y1** End point y coordinate
- **color** 16-bit 5-6-5 Color to draw with

◆ setRotation()

```
void Adafruit_GFX::setRotation ( uint8_t  x )                     virtual
```

Set rotation setting for display.

Parameters

- **x** 0 thru 3 corresponding to 4 cardinal rotations

◆ invertDisplay()

```
void Adafruit_GFX::invertDisplay ( bool  i )                      virtual
```

Invert the display (ideally using built-in hardware command)

Parameters

- **i** True if you want to invert, false to make 'normal'

Reimplemented in **Adafruit_SPITFT**, and **Adafruit_GrayOLED**.

◆ drawFastVLine()

```
void Adafruit_GFX::drawFastVLine ( int16_t  x,
                                   int16_t  y,
                                   int16_t  h,
                                   uint16_t color
                                 )                                virtual
```

Draw a perfectly vertical line (this is often optimized in a subclass!)

Parameters

- **x** Top-most x coordinate
- **y** Top-most y coordinate
- **h** Height in pixels
- **color** 16-bit 5-6-5 Color to fill with

Reimplemented in **GFXcanvas16**, **GFXcanvas8**, **GFXcanvas1**, and **Adafruit_SPITFT**.

◆ drawFastHLine()

```
void Adafruit_GFX::drawFastHLine ( int16_t  x,
                                   int16_t  y,
                                   int16_t  w,
                                   uint16_t color
                                 )                                  virtual
```

Draw a perfectly horizontal line (this is often optimized in a subclass!)

Parameters

- **x** Left-most x coordinate
- **y** Left-most y coordinate
- **w** Width in pixels
- **color** 16-bit 5-6-5 Color to fill with

Reimplemented in **GFXcanvas16**, **GFXcanvas8**, **GFXcanvas1**, and **Adafruit_SPITFT**.

◆ fillRect()

```
void Adafruit_GFX::fillRect ( int16_t  x,
                              int16_t  y,
                              int16_t  w,
                              int16_t  h,
                              uint16_t color
                            )                                       virtual
```

Fill a rectangle completely with one color. Update in subclasses if desired!

Parameters

- **x** Top left corner x coordinate
- **y** Top left corner y coordinate
- **w** Width in pixels
- **h** Height in pixels
- **color** 16-bit 5-6-5 Color to fill with

Reimplemented in **Adafruit_SPITFT**.

◆ fillScreen()

```
void Adafruit_GFX::fillScreen ( uint16_t  color )                   virtual
```

Fill the screen completely with one color. Update in subclasses if desired!

Parameters

- **color** 16-bit 5-6-5 Color to fill with

Reimplemented in **GFXcanvas16**, **GFXcanvas8**, and **GFXcanvas1**.

◆ drawLine()

```
void Adafruit_GFX::drawLine ( int16_t  x0,
                              int16_t  y0,
                              int16_t  x1,
                              int16_t  y1,
                              uint16_t color
                            )                                    virtual
```

Draw a line.

Parameters

x0	Start point x coordinate
y0	Start point y coordinate
x1	End point x coordinate
y1	End point y coordinate
color	16-bit 5-6-5 Color to draw with

◆ drawRect()

```
void Adafruit_GFX::drawRect ( int16_t  x,
                              int16_t  y,
                              int16_t  w,
                              int16_t  h,
                              uint16_t color
                            )                                    virtual
```

Draw a rectangle with no fill color.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
w	Width in pixels
h	Height in pixels
color	16-bit 5-6-5 Color to draw with

◆ drawCircle()

```
void Adafruit_GFX::drawCircle ( int16_t  x0,
                                int16_t  y0,
                                int16_t  r,
                                uint16_t color
                              )
```

Draw a circle outline.

Parameters

x0	Center-point x coordinate
y0	Center-point y coordinate
r	Radius of circle
color	16-bit 5-6-5 Color to draw with

◆ drawCircleHelper()

```
void Adafruit_GFX::drawCircleHelper ( int16_t  x0,
                                      int16_t  y0,
                                      int16_t  r,
                                      uint8_t  cornername,
                                      uint16_t color
                                    )
```

Quarter-circle drawer, used to do circles and roundrects.

Parameters

x0	Center-point x coordinate
y0	Center-point y coordinate
r	Radius of circle
cornername	Mask bit #1 or bit #2 to indicate which quarters of the circle we're doing
color	16-bit 5-6-5 Color to draw with

◆ fillCircle()

```
void Adafruit_GFX::fillCircle ( int16_t  x0,
                                int16_t  y0,
                                int16_t  r,
                                uint16_t color
                              )
```

Draw a circle with filled color.

Parameters

x0	Center-point x coordinate
y0	Center-point y coordinate
r	Radius of circle
color	16-bit 5-6-5 Color to fill with

◆ fillCircleHelper()

```
void Adafruit_GFX::fillCircleHelper ( int16_t  x0,
                                      int16_t  y0,
                                      int16_t  r,
                                      uint8_t  corners,
                                      int16_t  delta,
                                      uint16_t color
                                    )
```

Quarter-circle drawer with fill, used for circles and roundrects.

Parameters

x0	Center-point x coordinate
y0	Center-point y coordinate
r	Radius of circle
corners	Mask bits indicating which quarters we're doing
delta	Offset from center-point, used for round-rects
color	16-bit 5-6-5 Color to fill with

◆ drawTriangle()

```
void Adafruit_GFX::drawTriangle ( int16_t  x0,
                                  int16_t  y0,
                                  int16_t  x1,
                                  int16_t  y1,
                                  int16_t  x2,
                                  int16_t  y2,
                                  uint16_t color
                                )
```

Draw a triangle with no fill color.

Parameters

- **x0** Vertex #0 x coordinate
- **y0** Vertex #0 y coordinate
- **x1** Vertex #1 x coordinate
- **y1** Vertex #1 y coordinate
- **x2** Vertex #2 x coordinate
- **y2** Vertex #2 y coordinate
- **color** 16-bit 5-6-5 Color to draw with

◆ fillTriangle()

```
void Adafruit_GFX::fillTriangle ( int16_t  x0,
                                  int16_t  y0,
                                  int16_t  x1,
                                  int16_t  y1,
                                  int16_t  x2,
                                  int16_t  y2,
                                  uint16_t color
                                )
```

Draw a triangle with color-fill.

Parameters

- **x0** Vertex #0 x coordinate
- **y0** Vertex #0 y coordinate
- **x1** Vertex #1 x coordinate
- **y1** Vertex #1 y coordinate
- **x2** Vertex #2 x coordinate
- **y2** Vertex #2 y coordinate
- **color** 16-bit 5-6-5 Color to fill/draw with

◆ drawRoundRect()

```
void Adafruit_GFX::drawRoundRect ( int16_t  x,
                                   int16_t  y,
                                   int16_t  w,
                                   int16_t  h,
                                   int16_t  r,
                                   uint16_t color
                                 )
```

Draw a rounded rectangle with no fill color.

Parameters

- **x** Top left corner x coordinate
- **y** Top left corner y coordinate
- **w** Width in pixels
- **h** Height in pixels
- **r** Radius of corner rounding
- **color** 16-bit 5-6-5 Color to draw with

◆ fillRoundRect()

```
void Adafruit_GFX::fillRoundRect ( int16_t  x,
                                   int16_t  y,
                                   int16_t  w,
                                   int16_t  h,
                                   int16_t  r,
                                   uint16_t color
                                 )
```

Draw a rounded rectangle with fill color.

Parameters

- **x** Top left corner x coordinate
- **y** Top left corner y coordinate
- **w** Width in pixels
- **h** Height in pixels
- **r** Radius of corner rounding
- **color** 16-bit 5-6-5 Color to draw/fill with

◆ drawBitmap() [1/4]

```
void Adafruit_GFX::drawBitmap ( int16_t        x,
                                int16_t        y,
                                const uint8_t  bitmap[],
                                int16_t        w,
                                int16_t        h,
                                uint16_t       color
                              )
```

Draw a PROGMEM-resident 1-bit image at the specified (x,y) position, using the specified foreground color (unset bits are transparent).

Parameters

- **x** Top left corner x coordinate
- **y** Top left corner y coordinate
- **bitmap** byte array with monochrome bitmap
- **w** Width of bitmap in pixels
- **h** Height of bitmap in pixels
- **color** 16-bit 5-6-5 Color to draw with

◆ drawBitmap() [2/4]

```
void Adafruit_GFX::drawBitmap ( int16_t        x,
                                int16_t        y,
                                const uint8_t  bitmap[],
                                int16_t        w,
                                int16_t        h,
                                uint16_t       color,
                                uint16_t       bg
                              )
```

Draw a PROGMEM-resident 1-bit image at the specified (x,y) position, using the specified foreground (for set bits) and backg

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with monochrome bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels
color	16-bit 5-6-5 Color to draw pixels with
bg	16-bit 5-6-5 Color to draw background with

◆ drawBitmap() [3/4]

```
void Adafruit_GFX::drawBitmap ( int16_t    x,
                                int16_t    y,
                                uint8_t *  bitmap,
                                int16_t    w,
                                int16_t    h,
                                uint16_t   color
                              )
```

Draw a RAM-resident 1-bit image at the specified (x,y) position, using the specified foreground color (unset bits are transparent).

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with monochrome bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels
color	16-bit 5-6-5 Color to draw with

◆ drawBitmap() [4/4]

```
void Adafruit_GFX::drawBitmap ( int16_t    x,
                                int16_t    y,
                                uint8_t *  bitmap,
                                int16_t    w,
                                int16_t    h,
                                uint16_t   color,
                                uint16_t   bg
                              )
```

Draw a RAM-resident 1-bit image at the specified (x,y) position, using the specified foreground (for set bits) and background (unset bits) colors.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with monochrome bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels
color	16-bit 5-6-5 Color to draw pixels with
bg	16-bit 5-6-5 Color to draw background with

◆ drawXBitmap()

```
void Adafruit_GFX::drawXBitmap ( int16_t         x,
                                 int16_t         y,
                                 const uint8_t   bitmap[],
                                 int16_t         w,
                                 int16_t         h,
                                 uint16_t        color
                               )
```

Draw PROGMEM-resident XBitMap Files (*.xbm), exported from GIMP. Usage: Export from GIMP to *.xbm, rename *.xbm to *.c and open in editor. C Array can be directly used with this function. There is no RAM-resident version of this function; if generating bitmaps in RAM, use the format defined by **drawBitmap()** and call that instead.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with monochrome bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels
color	16-bit 5-6-5 Color to draw pixels with

◆ drawGrayscaleBitmap() [1/4]

```
void Adafruit_GFX::drawGrayscaleBitmap ( int16_t        x,
                                         int16_t        y,
                                         const uint8_t  bitmap[],
                                         int16_t        w,
                                         int16_t        h
                                       )
```

Draw a PROGMEM-resident 8-bit image (grayscale) at the specified (x,y) pos. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/ expansion is performed.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with grayscale bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels

◆ drawGrayscaleBitmap() [2/4]

```
void Adafruit_GFX::drawGrayscaleBitmap ( int16_t    x,
                                         int16_t    y,
                                         uint8_t *  bitmap,
                                         int16_t    w,
                                         int16_t    h
                                       )
```

Draw a RAM-resident 8-bit image (grayscale) at the specified (x,y) pos. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/ expansion is performed.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with grayscale bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels

◆ drawGrayscaleBitmap() [3/4]

```
void Adafruit_GFX::drawGrayscaleBitmap ( int16_t        x,
                                         int16_t        y,
                                         const uint8_t  bitmap[],
                                         const uint8_t  mask[],
                                         int16_t        w,
                                         int16_t        h
                                       )
```

Draw a PROGMEM-resident 8-bit image (grayscale) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (grayscale and mask) must be PROGMEM-resident. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with grayscale bitmap
mask	byte array with mask bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels

◆ drawGrayscaleBitmap() [4/4]

```
void Adafruit_GFX::drawGrayscaleBitmap ( int16_t    x,
                                         int16_t    y,
                                         uint8_t *  bitmap,
                                         uint8_t *  mask,
                                         int16_t    w,
                                         int16_t    h
                                       )
```

Draw a RAM-resident 8-bit image (grayscale) with a 1-bit mask (set bits = opaque, unset bits = clear) at the sp and mask) must be RAM-residentt, no mix-and-match Specifically for 8-bit display devices such as IS31FL37

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with grayscale bitmap
mask	byte array with mask bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels

◆ drawRGBBitmap() [1/4]

```
void Adafruit_GFX::drawRGBBitmap ( int16_t          x,
                                   int16_t          y,
                                   const uint16_t   bitmap[],
                                   int16_t          w,
                                   int16_t          h
                                 )
```

Draw a PROGMEM-resident 16-bit image (RGB 5/6/5) at the specified (x,y) position. For 16-bit display devices; no color reduction performed.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with 16-bit color bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels

◆ drawRGBBitmap() [2/4]

```
void Adafruit_GFX::drawRGBBitmap ( int16_t    x,
                                   int16_t    y,
                                   uint16_t *  bitmap,
                                   int16_t    w,
                                   int16_t    h
                                 )
```

Draw a RAM-resident 16-bit image (RGB 5/6/5) at the specified (x,y) position. For 16-bit display devices; no color reduction performed.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with 16-bit color bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels

◆ drawRGBBitmap() [3/4]

```
void Adafruit_GFX::drawRGBBitmap ( int16_t        x,
                                   int16_t        y,
                                   const uint16_t bitmap[],
                                   const uint8_t  mask[],
                                   int16_t        w,
                                   int16_t        h
                                 )
```

Draw a PROGMEM-resident 16-bit image (RGB 5/6/5) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (color and mask) must be PROGMEM-resident. For 16-bit display devices; no color reduction performed.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with 16-bit color bitmap
mask	byte array with monochrome mask bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels

◆ drawRGBBitmap() [4/4]

```
void Adafruit_GFX::drawRGBBitmap ( int16_t    x,
                                   int16_t    y,
                                   uint16_t * bitmap,
                                   uint8_t *  mask,
                                   int16_t    w,
                                   int16_t    h
                                 )
```

Draw a RAM-resident 16-bit image (RGB 5/6/5) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (color and mask) must be RAM-resident. For 16-bit display devices; no color reduction performed.

Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with 16-bit color bitmap
mask	byte array with monochrome mask bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels

◆ drawChar() [1/2]

```
void Adafruit_GFX::drawChar ( int16_t       x,
                              int16_t       y,
                              unsigned char c,
                              uint16_t      color,
                              uint16_t      bg,
                              uint8_t       size
                            )
```

Draw a single character.

Parameters

x	Bottom left corner x coordinate
y	Bottom left corner y coordinate
c	The 8-bit font-indexed character (likely ascii)
color	16-bit 5-6-5 Color to draw chraracter with
bg	16-bit 5-6-5 Color to fill background with (if same as color, no background)
size	Font magnification level, 1 is 'original' size

◆ drawChar() [2/2]

```
void Adafruit_GFX::drawChar ( int16_t       x,
                              int16_t       y,
                              unsigned char c,
                              uint16_t      color,
                              uint16_t      bg,
                              uint8_t       size_x,
                              uint8_t       size_y
                            )
```

Draw a single character.

Parameters

x	Bottom left corner x coordinate
y	Bottom left corner y coordinate
c	The 8-bit font-indexed character (likely ascii)
color	16-bit 5-6-5 Color to draw chraracter with
bg	16-bit 5-6-5 Color to fill background with (if same as color, no background)
size_x	Font magnification level in X-axis, 1 is 'original' size
size_y	Font magnification level in Y-axis, 1 is 'original' size

◆ getTextBounds() [1/3]

```
void Adafruit_GFX::getTextBounds ( const char *  str,
                                   int16_t       x,
                                   int16_t       y,
                                   int16_t *     x1,
                                   int16_t *     y1,
                                   uint16_t *    w,
                                   uint16_t *    h
                                 )
```

Helper to determine size of a string with current font/size. Pass string and a cursor position, returns UL corner and W,H.

Parameters

- **str** The ASCII string to measure
- **x** The current cursor X
- **y** The current cursor Y
- **x1** The boundary X coordinate, returned by function
- **y1** The boundary Y coordinate, returned by function
- **w** The boundary width, returned by function
- **h** The boundary height, returned by function

◆ getTextBounds() [2/3]

```
void Adafruit_GFX::getTextBounds ( const __FlashStringHelper *  str,
                                   int16_t                      x,
                                   int16_t                      y,
                                   int16_t *                    x1,
                                   int16_t *                    y1,
                                   uint16_t *                   w,
                                   uint16_t *                   h
                                 )
```

Helper to determine size of a PROGMEM string with current font/size. Pass string and a cursor position, returns UL corner and W,H.

Parameters

- **str** The flash-memory ascii string to measure
- **x** The current cursor X
- **y** The current cursor Y
- **x1** The boundary X coordinate, set by function
- **y1** The boundary Y coordinate, set by function
- **w** The boundary width, set by function
- **h** The boundary height, set by function

◆ getTextBounds() [3/3]

```
void Adafruit_GFX::getTextBounds ( const String &  str,
                                   int16_t          x,
                                   int16_t          y,
                                   int16_t *        x1,
                                   int16_t *        y1,
                                   uint16_t *       w,
                                   uint16_t *       h
                                 )
```

Helper to determine size of a string with current font/size. Pass string and a cursor position, returns UL corner and W,H.

Parameters

- **str** The ascii string to measure (as an arduino String() class)
- **x** The current cursor X
- **y** The current cursor Y
- **x1** The boundary X coordinate, set by function
- **y1** The boundary Y coordinate, set by function
- **w** The boundary width, set by function
- **h** The boundary height, set by function

◆ setTextSize() [1/2]

```
void Adafruit_GFX::setTextSize ( uint8_t  s )
```

Set text 'magnification' size. Each increase in s makes 1 pixel that much bigger.

Parameters

- **s** Desired text size. 1 is default 6x8, 2 is 12x16, 3 is 18x24, etc

◆ setTextSize() [2/2]

```
void Adafruit_GFX::setTextSize ( uint8_t  s_x,
                                 uint8_t  s_y
                               )
```

Set text 'magnification' size. Each increase in s makes 1 pixel that much bigger.

Parameters

- **s_x** Desired text width magnification level in X-axis. 1 is default
- **s_y** Desired text width magnification level in Y-axis. 1 is default

◆ setFont()

```
void Adafruit_GFX::setFont ( const GFXfont *  f = NULL )
```

Set the font to display when print()ing, either custom or default.

Parameters

- **f** The **GFXfont** object, if NULL use built in 6x8 font

◆ setCursor()

void Adafruit_GFX::setCursor (int16_t x,
int16_t y
) inline

Set text cursor location.

Parameters

x X coordinate in pixels

y Y coordinate in pixels

◆ setTextColor() [1/2]

void Adafruit_GFX::setTextColor (uint16_t c) inline

Set text font color with transparant background.

Parameters

c 16-bit 5-6-5 Color to draw text with

Note

For 'transparent' background, background and foreground are set to same color rather than using a separate flag.

◆ setTextColor() [2/2]

void Adafruit_GFX::setTextColor (uint16_t c,
uint16_t bg
) inline

Set text font color with custom background color.

Parameters

c 16-bit 5-6-5 Color to draw text with

bg 16-bit 5-6-5 Color to draw background/fill with

◆ setTextWrap()

void Adafruit_GFX::setTextWrap (bool w) inline

Set whether text that is too long for the screen width should automatically wrap around to the next line (else clip right).

Parameters

w true for wrapping, false for clipping

◆ cp437()

void Adafruit_GFX::cp437 (bool x = true) inline

Enable (or disable) Code Page 437-compatible charset. There was an error in glcdfont.c for the longest time – one character (#176, the 'light shade' block) was missing – this threw off the index of every character that followed it. But a TON of code has been written with the erroneous character indices. By default, the library uses the original 'wrong' behavior and old sketches will still work. Pass 'true' to this function to use correct CP437 character values in your code.

Parameters

x true = enable (new behavior), false = disable (old behavior)

◆ write()

size_t Adafruit_GFX::write (uint8_t c) virtual

Print one byte/character of data, used to support print()

Parameters

c The 8-bit ascii character to write

◆ width()

int16_t Adafruit_GFX::width (void) const inline

Get width of the display, accounting for current rotation.

Returns

Width in pixels

◆ height()

int16_t Adafruit_GFX::height (void) const inline

Get height of the display, accounting for current rotation.

Returns

Height in pixels

◆ getRotation()

uint8_t Adafruit_GFX::getRotation (void) const inline

Get rotation setting for display.

Returns

0 thru 3 corresponding to 4 cardinal rotations

◆ getCursorX()

int16_t Adafruit_GFX::getCursorX (void) const inline

Get text cursor X location.

Returns

X coordinate in pixels

◆ getCursorY()

int16_t Adafruit_GFX::getCursorY (void) const inline

Get text cursor Y location.

Returns

Y coordinate in pixels

◆ charBounds()

```
void Adafruit_GFX::charBounds ( unsigned char  c,
                                int16_t *      x,
                                int16_t *      y,
                                int16_t *      minx,
                                int16_t *      miny,
                                int16_t *      maxx,
                                int16_t *      maxy
                              )
```

protected

Helper to determine size of a character with current font/size. Broke this out as it's used by both the PROGMEM- and RAM-resident **getTextBounds()** functions.

Parameters

c	The ASCII character in question
x	Pointer to x location of character. Value is modified by this function to advance to next character.
y	Pointer to y location of character. Value is modified by this function to advance to next character.
minx	Pointer to minimum X coordinate, passed in to AND returned by this function – this is used to incrementally build a bounding rectangle for a string.
miny	Pointer to minimum Y coord, passed in AND returned.
maxx	Pointer to maximum X coord, passed in AND returned.
maxy	Pointer to maximum Y coord, passed in AND returned.

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

- **Adafruit_GFX.h**
- Adafruit_GFX.cpp

Generated by doxygen 1.8.13