- setFontRefHeightText
  - setHardwareBackup
  - setPrintPos
  - setRGB
  - `setRot90`
  - `setRot180`
  - `setRot270`
  - `setScale2x2`
  - sleepOn
  - sleepOff
  - `undoRotation`
  - `undoScale`
  - U8GLIB

begin

- **C++/Arduino:** `uint8_t U8GLIB::begin(void)`
- **C:** `uint8_t u8g_Begin(u8g_t *u8g)`
- **Description:** Reset display and put it into default state.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:** `0`, if the init procedure fails.
- **Use:** Outside picture loop.
- **Note:** Available with v1.11.
- **Example:**
- **See also:** U8GLIB

disableCursor

- **C++/Arduino:** `void U8GLIB::disableCursor(void)`
- **C:** `void u8g_DisableCursor(u8g_t *u8g)`
- **Description:** Disable the cursor. The cursor will not be visible.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**
- **Use:** Outside picture loop.
- **Note:**
- **Example:**
- **See also:** enableCursor, setCursorColor, setCursorFont, setCursorPos, setCursorStyle

drawBitmap

drawBitmapP

- **C++/Arduino:**

```
void U8GLIB::drawBitmap(u8g_uint_t x, u8g_uint_t y, u8g_uint_t cnt, u8g_uint_t
void U8GLIB::drawBitmapP(u8g_uint_t x, u8g_uint_t y, u8g_uint_t cnt, u8g_uint_
```

- **C:**

```
void u8g_DrawBitmap(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, u8g_uint_t cnt, u8
void u8g_DrawBitmapP(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, u8g_uint_t cnt, u
```

- **Description:** Draw a bitmap at the specified x/y position (upper left corner of the bitmap). Parts of the bitmap may be outside the display boundaries.The bitmap is specified by the array `bitmap`. A cleared bit means: Do not draw a pixel. A set bit inside the array means: Write pixel with the current color index.For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.

- **Arguments:**

  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x`: X-position (left position of the bitmap).
  - `y`: Y-position (upper position of the bitmap).
  - `cnt`: Number of bytes of the bitmap in horizontal direction. The width of the bitmap is `cnt*8`.
  - `h`: Height of the bitmap.

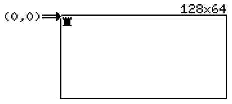
- **Returns:**

- **Use:** Inside picture loop.

- **Note:**

- **Example:**

```
U8GLIB_PCD8544 u8g(13, 11, 10, 9, 8);                    // SPI communication:

const uint8_t rook_bitmap[] U8G_PROGMEM = {
  0x00,         // 00000000
  0x55,         // 01010101
  0x7f,          // 01111111
  0x3e,         // 00111110
  0x3e,         // 00111110
  0x3e,         // 00111110
  0x3e,         // 00111110
  0x7f           // 01111111
};

void draw(void) {
  // graphic commands to redraw the complete screen should be placed here
  u8g.drawBitmapP( 0, 0, 1, 8, rook_bitmap);
}

void setup(void) {
}

void loop(void) {
  // picture loop
  u8g.firstPage();
  do {
    draw();
  } while( u8g.nextPage() );

  // rebuild the picture after some delay
  delay(1000);
}
```

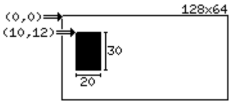


- **See also:** setColorIndex drawXBM

drawBox

- **C++/Arduino:** `void U8GLIB::drawBox(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h)`
- **C:** `void u8g_DrawBox(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h)`
- **Description:** Draw a box (filled frame), starting at x/y position (upper left edge). The box has width `w` and height `h`. Parts of the box can be outside of the display boundaries.This procedure uses the current color index to draw the box. For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x`: X-position of upper left edge.
  - `y`: Y-position of upper left edge.
  - `w`: Width of the box.
  - `h`: Height of the box.
- **Returns:**
- **Use:** Inside picture loop.
- **Note:**
- **Example:**

```
U8GLIB u8g(...)
...
u8g.drawBox(10,12,20,30);
```

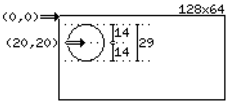


- **See also:** setColorIndex, drawFrame

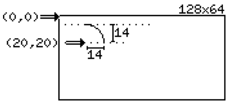
drawCircle

- **C++/Arduino:** `void U8GLIB::drawCircle(u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rad, uint8_t opt = U8G_DRAW_ALL)`
- **C:** `void u8g_DrawCircle(u8g_t *u8g, u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rad, uint8_t opt)`
- **Description:** Draw a circle with radius `rad` at position `(x0, y0)`. The diameter of the circle is `2*rad+1`Depending on `opt`, it is possible to draw only some sections of the circle. Possible values for `opt` are:`U8G_DRAW_UPPER_RIGHT`, `U8G_DRAW_UPPER_LEFT`, `U8G_DRAW_LOWER_LEFT`, `U8G_DRAW_LOWER_RIGHT`, `U8G_DRAW_ALL`.These values can be combined with the `|` operator.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x0`, `y0`: Position of the center of the circle.
  - `rad`: Defines the size of the circle: Radus = `rad`.
  - `opt`: Selects some or all sections of the circle.
    - `U8G_DRAW_UPPER_RIGHT`
    - `U8G_DRAW_UPPER_LEFT`
    - `U8G_DRAW_LOWER_LEFT`
    - `U8G_DRAW_LOWER_RIGHT`
    - `U8G_DRAW_ALL`
- **Returns:**
- **Use:** Inside picture loop.
- **Note:** Available with v1.02
- **Example:**

```
u8g.drawCircle(20, 20, 14);
```



```
u8g.drawCircle(20, 20, 14, U8G_DRAW_UPPER_RIGHT);
```



- **See also:** drawDisc

drawDisc

- **C++/Arduino:** `void U8GLIB::drawDisc(u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rad, uint8_t opt = U8G_DRAW_ALL)`
- **C:** `void u8g_DrawDisc(u8g_t *u8g, u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rad, uint8_t opt)`
- **Description:** Draw a filled circle with radius `rad` at position `(x0, y0)`. The diameter of the circle is `2*rad+1`Depending on `opt`, it is possible to draw only some sections of the disc. Possible values for `opt` are:`U8G_DRAW_UPPER_RIGHT`, `U8G_DRAW_UPPER_LEFT`, `U8G_DRAW_LOWER_LEFT`, `U8G_DRAW_LOWER_RIGHT`, `U8G_DRAW_ALL`.These values can be combined with the `|` operator.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x0`, `y0`: Position of the center of the filled circle.
  - `rad`: Defines the size of the disc: Radus = `rad`.
  - `opt`: Selects some or all sections of the circle.
    - `U8G_DRAW_UPPER_RIGHT`
    - `U8G_DRAW_UPPER_LEFT`
    - `U8G_DRAW_LOWER_LEFT`
    - `U8G_DRAW_LOWER_RIGHT`
    - `U8G_DRAW_ALL`
- **Returns:**
- **Use:** Inside picture loop.
- **Note:** Available with v1.02
- **Example:** See drawCircle
- **See also:** drawCircle

drawEllipse

- **C++/Arduino:** `void U8GLIB::drawEllipse(u8g_t *u8g, u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rx, u8g_uint_t ry, uint8_t opt)`
- **C:** `void u8g_DrawEllipse(u8g_t *u8g, u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rx, u8g_uint_t ry, uint8_t opt)`
- **Description:** Draw ellipse with radius `rx` and 'ry' at position `(x0, y0)`. `rx*ry` must be lower than 1024 in 8 Bit mode of u8glib.Depending on `opt`, it is possible to draw only some sections of the disc. Possible values for `opt` are:`U8G_DRAW_UPPER_RIGHT`, `U8G_DRAW_UPPER_LEFT`, `U8G_DRAW_LOWER_LEFT`, `U8G_DRAW_LOWER_RIGHT`, `U8G_DRAW_ALL`.These values can be combined with the `|` operator.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x0`, `y0`: Position of the center of the filled circle.
  - `rx`, `rx`: Defines the size of the ellipse.
  - `opt`: Selects some or all sections of the ellipse.
    - `U8G_DRAW_UPPER_RIGHT`
    - `U8G_DRAW_UPPER_LEFT`
    - `U8G_DRAW_LOWER_LEFT`
    - `U8G_DRAW_LOWER_RIGHT`
    - `U8G_DRAW_ALL`
- **Returns:**
- **Use:** Inside picture loop.
- **Note:** Available with v1.14
- **See also:** drawCircle

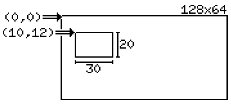
drawFilledEllipse

- **C++/Arduino:** `void U8GLIB::drawFilledEllipse(u8g_t *u8g, u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rx, u8g_uint_t ry, uint8_t opt)`
- **C:** `void u8g_DrawFilledEllipse(u8g_t *u8g, u8g_uint_t x0, u8g_uint_t y0, u8g_uint_t rx, u8g_uint_t ry, uint8_t opt)`
- **Description:** Draw a filled ellipse with radius `rx` and 'ry' at position `(x0, y0)`. `rx*ry` must be lower than 1024 in 8 Bit mode of u8glib.Depending on `opt`, it is possible to draw only some sections of the disc. Possible values for `opt` are:`U8G_DRAW_UPPER_RIGHT`, `U8G_DRAW_UPPER_LEFT`, `U8G_DRAW_LOWER_LEFT`, `U8G_DRAW_LOWER_RIGHT`, `U8G_DRAW_ALL`.These values can be combined with the `|` operator.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x0`, `y0`: Position of the center of the filled circle.
  - `rx`, `rx`: Defines the size of the ellipse.
  - `opt`: Selects some or all sections of the ellipse.
    - `U8G_DRAW_UPPER_RIGHT`
    - `U8G_DRAW_UPPER_LEFT`
    - `U8G_DRAW_LOWER_LEFT`
    - `U8G_DRAW_LOWER_RIGHT`
    - `U8G_DRAW_ALL`
- **Returns:**
- **Use:** Inside picture loop.
- **Note:** Available with v1.14
- **See also:** drawCircle

drawFrame

- **C++/Arduino:** `void U8GLIB::drawFrame(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h)`
- **C:** `void u8g_DrawFrame(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h)`
- **Description:** Draw a frame, starting at x/y position (upper left edge). The frame has width `w` and height `h`. Parts of the frame can be outside of the display boundaries.This procedure uses the current color index to draw the lines of the frame. For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x`: X-position of upper left edge.
  - `y`: Y-position of upper left edge.
  - `w`: Width of the frame.
  - `h`: Height of the frame.
- **Returns:**
- **Use:** Inside picture loop.
- **Note:**
- **Example:**

```
U8GLIB u8g(...)
...
u8g.drawFrame(10,12,30,20);
```



- **See also:** setColorIndex, drawBox

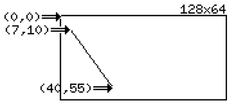
drawHLine

- **C++/Arduino:** `void U8GLIB::drawHLine(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w)`
- **C:** `void u8g_DrawHLine(u8g_t *u8g, uint8_t x, uint8_t y, u8g_uint_t w)`
- **Description:** Draw a horizontal line, starting at x/y position (left edge). The width of the line is `w` pixels. Parts of the line can be outside of the display boundaries.This procedure uses the current color index to draw the line. For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x`: X-position.
  - `y`: Y-position.
  - `w`: Width of the horizontal line.
- **Returns:**
- **Use:** Inside picture loop.
- **Note:**
- **Example:**
- **See also:** setColorIndex, drawVLine

drawLine

- **C++/Arduino:** `void U8GLIB::drawLine(u8g_uint_t x1, u8g_uint_t y1, u8g_uint_t x2, u8g_uint_t y2)`
- **C:** `void u8g_DrawLine(u8g_t *u8g, u8g_uint_t x1, u8g_uint_t y1, u8g_uint_t x2, u8g_uint_t y2)`
- **Description:** Draw a line from (x1, y1) to (x2, y2). There are no restrictions on the start end end position.This procedure uses the current color index to draw the line. For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x1`, `y1`: Start position.
  - `x2`, `y2`: End position.
- **Returns:**
- **Use:** Inside picture loop.
- **Note:** Available in v1.03.
- **Example:**

```
  u8g.drawLine(7, 10, 40, 55);
```

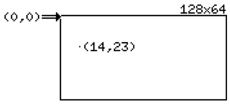


- **See also:** setColorIndex, drawVLine drawHLine

drawPixel

- **C++/Arduino:** `void U8GLIB::drawPixel(uint8_t x, uint8_t y)`
- **C:** `void u8g_DrawPixel(u8g_t *u8g, uint8_t x, uint8_t y)`
- **Description:** Draw a pixel at the specified x/y position. Position (0,0) is at the upper left corner of the display. The position may be outside the display boundaries.This procedure uses the current color index to draw the pixel. For a monochrome display, the color index 0 will usually clear the pixel and the color index 1 will set a pixel.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x`: X-position.
  - `y`: Y-position.
- **Returns:**
- **Use:** Inside picture loop.
- **Note:**
- **Example:**

```
U8GLIB u8g(...)
...
u8g.drawPixel(14,23);
```



- **See also:** setColorIndex

drawRBox

drawRFrame

- **C++/Arduino:**

```
void U8GLIB::drawRBox(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h,
```

```
void U8GLIB::drawRFrame(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h
```

- **C:**

```
void u8g_DrawRBox(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_ui
void u8g_DrawRFrame(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_
```

- **Description:** Draw a box/frame with round edges, starting at x/y position (upper left edge). The box/frame has width `w` and height `h`. Parts of the box can be outside of the display boundaries.Edges have radius `r`. It is required that `w >= 2*(r+1)` and `h >= 2*(r+1)`. This condition is not checked. Behavior is undefined if `w` or `h` is smaller than `2*(r+1)`.This procedure uses the current color index to draw the box. For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x`: X-position of upper left edge.
  - `y`: Y-position of upper left edge.
  - `w`: Width of the box.
  - `h`: Height of the box.
  - `r`: Radius for the four edges.
- **Returns:**
- **Use:** Inside picture loop.
- **Note:** Available with v1.09
- **Note:**
- **See also:** setColorIndex, drawFrame drawBox

drawStr

drawStr90

drawStr180

drawStr270

drawStrP

drawStr90P

drawStr180P

drawStr270P

- **C++/Arduino:**

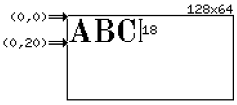
```
u8g_uint_t U8GLIB::drawStr(u8g_uint_t x, u8g_uint_t y, const char *s)
u8g_uint_t U8GLIB::drawStr90(u8g_uint_t x, u8g_uint_t y, const char *s)
u8g_uint_t U8GLIB::drawStr180(u8g_uint_t x, u8g_uint_t y, const char *s)
u8g_uint_t U8GLIB::drawStr270(u8g_uint_t x, u8g_uint_t y, const char *s)
u8g_uint_t U8GLIB::drawStrP(u8g_uint_t x, u8g_uint_t y, const u8g_pgm_uint8_t
u8g_uint_t U8GLIB::drawStr90P(u8g_uint_t x, u8g_uint_t y, const u8g_pgm_uint8_
u8g_uint_t U8GLIB::drawStr180P(u8g_uint_t x, u8g_uint_t y, const u8g_pgm_uint8
u8g_uint_t U8GLIB::drawStr270P(u8g_uint_t x, u8g_uint_t y, const u8g_pgm_uint8
```

- **C:**

```
u8g_uint_t u8g_DrawStr(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, const char *s);
u8g_uint_t u8g_DrawStr90(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, const char *s
u8g_uint_t u8g_DrawStr180(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, const char *
u8g_uint_t u8g_DrawStr270(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, const char *
u8g_uint_t u8g_DrawStrP(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, const u8g_pgm_
u8g_uint_t u8g_DrawStr90P(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, const u8g_pg
u8g_uint_t u8g_DrawStr180P(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, const u8g_p
u8g_uint_t u8g_DrawStr270P(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, const u8g_p
```

- **Description:** Draws a string at the specified x/y position. The x/y position is the lower left corner of the first character of the string.It is required to assign a font with the setFont procedure before the first call to this procedure.This procedure also uses the current color index to draw the characters. For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.The (x,y) arguments are influenced by the reference point calculation mode (`setFontPosBaseline`).'P' variant: `s` is assumed to point to a string in PROGMEM area.'90', '180', '270' variants: Rotate string output by 90, 180 or 270 degree.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x`: X-position.
  - `y`: Y-position.
  - `s`: A pointer to a C-string (terminated with `\0`).
- **Returns:**The width of the string `s` in pixel.
- **Use:** Inside picture loop.
- **Note:**The C++ Arduino environment also offers the more powerful print procedure.
- **Example:**

```
U8GLIB u8g(...)
...
u8g.setFont(u8g_font_osb18);
u8g.drawStr(0, 20, "ABC");
```



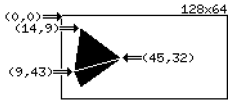
The reference point (0,20) for the origin of the text string usually is one pixel below the lower left edge of the first character. The height of the uppercase letters is shown in the font overview bitmap (in this example 18, see here). In some cases the size of the uppercase letters is also part of the font name.

- **See also:** setColorIndex, setFont, setFontPosBaseline, print

drawTriangle

- **C++/Arduino:** `void U8GLIB::drawTriangle(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2)`
- **C:** `void u8g_DrawTriangle(u8g_t *u8g, uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2)`
- **Description:** Draw a triangle (filled polygon). Arguments are 16 bit and the polygon is clipped to the size of the display. Multiple polygons are drawn so that they exactly match without overlap:The left side of a polygon is drawn, the right side is not draw. The upper side is only draw if it is flat. In theexample picture below, the pixel at (9,43) is drawn by the polygon procedures, but pixels (14,9) and (45,32) are not drawn.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x0`: X-position point 0.
  - `y0`: Y-position point 0.
  - `x1`: X-position point 1.
  - `y1`: Y-position point 1.
  - `x2`: X-position point 2.
  - `y2`: Y-position point 2.
- **Returns:**
- **Use:** Inside picture loop.
- **Note:** Available with v1.15
- **Example:**

```
U8GLIB u8g(...)
...
u8g.drawTriangle(14,9, 45,32, 9,42);
u8g.drawTriangle(14,55, 45,33, 9,43);
```



- **See also:** setColorIndex

drawVLine

- **C++/Arduino:** `void U8GLIB::drawVLine(u8g_uint_t x, u8g_uint_t y, u8g_uint_t h)`
- **C:** `void u8g_DrawVLine(u8g_t *u8g, uint8_t x, uint8_t y, u8g_uint_t h)`
- **Description:** Draw a vertical line, starting at x/y position (upper edge). The height of the line is `h` pixels. Parts of the line can be outside of the display boundaries.This procedure uses the current color index to draw the line. For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x`: X-position.
  - `y`: Y-position.
  - `h`: Height of the horizontal line.
- **Returns:**
- **Use:** Inside picture loop.
- **Note:**
- **Example:**
- **See also:** setColorIndex, drawHLine

drawXBM

drawXBMP

- **C++/Arduino:**

```
void U8GLIB::drawXBM(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h, c
void U8GLIB::drawXBMP(u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uint_t h,
```

- **C:**

```
void u8g_DrawXBM(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_uir
void u8g_DrawXBMP(u8g_t *u8g, u8g_uint_t x, u8g_uint_t y, u8g_uint_t w, u8g_ui
```

- **Description:** Draw a XBM Bitmap. Position (x,y) is the upper left corner of the bitmap. XBM contains monochrome, 1-bit bitmaps. This procedure only draws pixel values 1. The current color index is used for drawing (see setColorIndex). Pixel with value 0 are not drawn (transparent). Many tools can save a bitmap as XBM. The result will look like this **Example:**

```
#define u8g_logo_width 38
#define u8g_logo_height 24
static unsigned char u8g_logo_bits[] = {
   0xff, 0xff, 0xff, 0xff, 0x3f, 0xff, 0xff, 0xff, 0xff, 0x3f, 0xe0, 0xe0,
...
   0xff, 0x3f, 0xff, 0xff, 0xff, 0xff, 0x3f, 0xff, 0xff, 0xff, 0xff, 0x3f };
```

This could can be copied directly into your code. Use `drawXBM` to draw this bitmap at (0,0):

```
u8g.drawXBM( 0, 0, u8g_logo_width, u8g_logo_height, u8g_logo_bits);
```

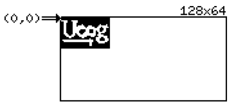
In most cases it is better to place the bitmap into AVR PROGMEM area. Add the `U8G_PROGMEM` after the array definition before the init sequence:

```
static unsigned char u8g_logo_bits[] U8G_PROGMEM = {
```

With this modification call the `drawXBMP` variant:

```
u8g.drawXBMP( 0, 0, u8g_logo_width, u8g_logo_height, u8g_logo_bits);
```

- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `x`: X-position.
  - `y`: Y-position.
  - `w`: Width of the bitmap.
  - `h`: Height of the bitmap.
  - `bitmap`: Pointer to the start of the bitmap.
- **Returns:**
- **Use:** Inside picture loop.
- **Note:**
- **Example:**



- **See also:** setColorIndex, drawBitmap

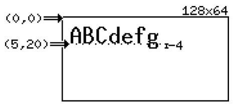
enableCursor

- **C++/Arduino:** `void U8GLIB::enableCursor(void)`
- **C:** `void u8g_EnableCursor(u8g_t *u8g)`
- **Description:** Enable the cursor at the specified position.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**
- **Use:** Outside picture loop.

getFontDescent

- **C++/Arduino:** `u8g_int_t U8GLIB::getFontDescent(void)`
- **C:** `u8g_int_t u8g_GetFontDescent(u8g_t *u8g)`
- **Description:** Returns the reference height of the glyphs below the baseline (descent).
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**The descent of the current font.
- **Use:** Inside and outside picture loop.
- **Note:**
- **Example:**For `u8g_font_10x20` the descent is -4.

```
u8g_SetFont(u8g, u8g_font_10x20);
u8g_DrawStr(u8g, 5, 20, "ABCdefg");
d = u8g_GetFontDescent(u8g);
```

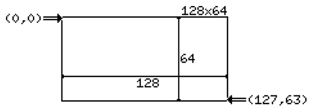


The dotted line shows the baseline of the string. The string itself is above the baseline. The reference point for the string (5, 20) is exactly on the baseline. The ascent is the number of pixels of the highest glyph above baseline. To calculate the y position which is below the glyph with the highest descent, use `baseline_y_pos-u8g_GetFontDescent(u8g)`.

- **See also:** setFont getFontAscent

getFontLineSpacing

- **C++/Arduino:** `u8g_int_t U8GLIB::getFontLineSpacing(void)`



- **See also:** getHeight

getStrWidth

- **C++/Arduino:**

```
u8g_uint_t U8GLIB::getStrWidth(const char *s)
u8g_uint_t U8GLIB::getStrWidthP(const u8g_pgm_uint8_t *s)
```

- **C:**

```
u8g_uint_t u8g_GetStrWidth(u8g_t *u8g, const char *s)
u8g_uint_t u8g_GetStrWidthP(u8g_t *u8g, const u8g_pgm_uint8_t *s)
```

- **Description:** Returns the width of the string "s", based on the current font.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
  - `s` : Pointer to a string.
- **Returns:**The width of the string.
- **Use:** Inside the picture loop.
- **Note:**
- **Example:** See Tutorial Font and String Handling
- **See also:** setFont

`InitSPI, InitHWSPI, Init8Bit InitComFn`

- **C:**

- **Note:**
- **Example:**
- **See also:** disableCursor, setCursorColor, setCursorFont, setCursorPos, setCursorStyle

firstPage

- **C++/Arduino:** `void U8GLIB::firstPage(void)`
- **C:** `void u8g_FirstPage(u8g_t *u8g)`
- **Description:** A call to this procedure, marks the beginning of the picture loop.
- **Arguments:**
- **Returns:**
- **Use:** This procedure call starts the picture loop; it cannot be used inside the picture loop. Picture loops cannot be nested.
- **Note:**
- **Example:**
- **See also:** nextPage

getColorIndex

- **C++/Arduino:** `uint8_t U8GLIB::getColorIndex(void)`
- **C:** `uint8_t u8g_GetColorIndex(u8g_t *u8g)`
- **Description:** The current "color index" is used by all "draw" procedures to set a pixel value on the display.This procedure returns the current value, which has been set as current color index.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**Value, which is used by the "draw" procedures as a pixel value.
- **Use:** Inside and outside picture loop.
- **Note:**
- **Example:**

- **C:** `u8g_int_t u8g_getFontLineSpacing(u8g_t *u8g)`
- **Description:** Returns the vertical distance of two lines of text, written with the current font. This value is derived from the ascent and descent value and multiplied with the current `LineSpacingFactor`. The returned value is influenced by the current font, the "Reference Height" and the `LineSpacingFactor`.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**The distance of two lines (pixel).
- **Use:** Inside and outside picture loop.
- **Note:**
- **See also:** setFont getFontAscent getFontDescent setFontRefHeightAll setLineSpacingFactor

getHeight

- **C++/Arduino:** `u8g_uint_t U8GLIB::getHeight(void)`
- **C:** `u8g_uint_t u8g_GetHeight(u8g_t *u8g)`
- **Description:** Returns the height of the display.
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**The height of the display.
- **Use:** Inside and outside picture loop.
- **Note:**
- **Example:**
- **See also:** getWidth

getMode

- **C++/Arduino:** `uint8_t U8GLIB::getMode(void)`
- **C:** `uint8_t u8g_GetMode(u8g_t *u8g)`
- **Description:** Returns information about the display (display mode).The result of this procedure can be used to extract the number of

```
uint8_t u8g_InitSPI(u8g_t *u8g, u8g_dev_t *dev, uint8_t sck, uint8_t mosi, uir
uint8_t u8g_InitHWSPI(u8g_t *u8g, u8g_dev_t *dev, uint8_t cs, uint8_t a0, uint
uint8_t u8g_Init8Bit(u8g_t *u8g, u8g_dev_t *dev, uint8_t d0, uint8_t d1, uint8
  uint8_t en, uint8_t cs1, uint8_t cs2, uint8_t di, uint8_t rw, uint8_t reset)
uint8_t u8g_InitComFn(u8g_t *u8g, u8g_dev_t *dev, u8g_com_fnptr com_fn);
```

- **Description:** For the C-Interface only: Create a new interface to a graphics display. This procedure must be called before calling any other C-procedure.The `dev` argument describes the type of the display. See here for a complete list of available devices. `u8g_InitComFn` will be the defaultinit precedure for the ARM plattfrom. It requires a specific low level procedure for the ARM controller. An examples for this procedure ishere for the LPC1114 (end of u8g_arm.c). A more detailed **Description:** of the communication procedure can be found here: INSTALL
- **Arguments:**
  - `dev`: A pointer to a device structure.
  - Arduino pins: Required pins to connect the display depending on the communication interface.
  - `reset`: The reset pin is optional and can be `U8G_PIN_NONE`
  - `com_fn`: A procedure that handles low level access to the display.
- **Returns:**
- **Use:** Outside picture loop.
- **Note:** `u8g_InitComFn` is avialble with v1.14.
- **Example:**
- **See also:** List of supported devices, U8GLIB C++ Constructor

nextPage

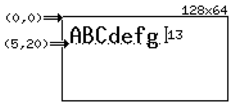
- **C++/Arduino:** `uint8_t U8GLIB::nextPage(void)`
- **C:** `uint8_t u8g_NextPage(u8g_t *u8g)`
- **Description:** A call to this procedure, marks the end of the body of the picture loop.

- **See also:** drawPixel setColorIndex

getFontAscent

- **C++/Arduino:** `u8g_int_t U8GLIB::getFontAscent(void)`
- **C:** `u8g_int_t u8g_GetFontAscentu8g_t *u8g)`
- **Description:** Returns the reference height of the glyphs above the baseline (ascent). This value depends on the current reference height (see setFontRefHeightAll).
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**The ascent of the current font.
- **Use:** Inside and outside picture loop.
- **Note:**
- **Example:**For `u8g_font_10x20` the ascent is 13.

```
u8g_SetFont(u8g, u8g_font_10x20);
u8g_DrawStr(u8g, 5, 20, "ABCdefg");
a = u8g_GetFontAscent(u8g);
```



The dotted line shows the baseline of the string. The string itself is above the baseline. The reference point for the string (5, 20) is exactly on the baseline. The ascent is the number of pixels of the highest glyph above baseline. To calculate the y position which is above the largest glyph, use `baseline_y_pos-u8g_GetFontAscent(u8g)-1`.

- **See also:** setFont getFontDescent setFontRefHeightAll

bits per pixel: `U8G_MODE_GET_BITS_PER_PIXEL(mode)`. Predefined modes are:
  - `U8G_MODE_BW`: black/white monochrome mode with 1 bit per pixel
  - `U8G_MODE_GRAY2BIT`: Graylevel mode with 2 bit per pixel
- **Arguments:**
  - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**The current display mode.
- **Use:** Inside and outside picture loop.
- **Note:**
- **Example:**
- **See also:**

getWidth

- **C++/Arduino:** `u8g_uint_t U8GLIB::getWidth(void)`

- **C:** `u8g_uint_t u8g_GetWidth(u8g_t *u8g)`

- **Description:** Returns the width of the display.

- **Arguments:**

  - `u8g` : Pointer to the `u8g` structure (C interface only).

- **Returns:**The width of the display.

- **Use:** Inside and outside picture loop.

- **Note:**

- **Example:**This procedure will return 128 for an attached display with the dimensions 128x64.

- **Arguments:**
- **Returns:** `0`, if the picture loop has been finished, `1` if another redraw of the picture is required.
- **Use:** This procedure call marks the body of the picture loop, it can not be used inside the picture loop (Picture loops can not be nested).
- **Note:**This procedure will not reset or modify any internal values (like the draw color or the current font). The font settings and draw properties at the end of the body of the picture loop are still the same when the body of the picture loop is started again. Usually it is a good idea to set such properties at the beginning of the body of the picture loop.
- **Example:**
- **See also:** firstPage Picture Loop

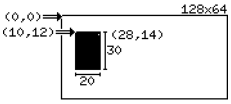
print

- **C++/Arduino:** `U8GLIB::print(...)`
- **C:**
- **Description:** A call to the print procedure of the `Print` base class. See the documentation on the Arduino web page: http://arduino.cc/en/Serial/Print.`print()` behaves similar to `drawStr`. All font settings also apply to this procedure. All strings and values passed to the `print` procedureare written to the "print position". The "print position" can be set via setPrintPos.
- **Arguments:** See http://arduino.cc/en/Serial/Print
- **Returns:** See http://arduino.cc/en/Serial/Print
- **Use:** Inside the picture loop.
- **Note:**
- **Example:**
- **See also:** setPrintPos drawStr

setColorIndex

- **C++/Arduino:** `void U8GLIB::setColorIndex(uint8_t color_index)`
- **C:** `void u8g_SetColorIndex(u8g_t *u8g, uint8_t color_index)`
- **Description:** The current "color index" is used by all "draw" procedures to set a pixel value on the display.For a monochrome display, the color index 0 will usually clear a pixel and the color index 1 will set a pixel.For a display which supports gray levels, this procedure sets the gray level for drawing.
- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
    - `color_index`: Value, which is used by the "draw" procedures as a pixel value.
- **Returns:**
- **Use:** Inside and outside picture loop. It is a good practice to use this procedure at the beginning of the body of the picture loop.
- **Note:**
- **Example:**

```
U8GLIB u8g(...)
...
u8g.setColorIndex(1);
u8g.drawBox(10, 12, 20, 30);
u8g.setColorIndex(0);
u8g.drawPixel(28, 14);          // clear pixel at (28, 14)
```



- **See also:** drawPixel getColorIndex setDefaultBackgroundColor

setContrast

- **C++/Arduino:** `uint8_t U8GLIB::setContrast(uint8_t contast)`
- **C:** `uint8_t u8g_SetContrast(u8g_t *u8g, uint8_t contast)`
- **Description:** Assigns a new contrast value (0..255) to the display. Not all displays or driver support the setting of the contrast value (see devices table).
- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
    - `contrast`: New contrast value (0..255).
- **Returns:**The value 1, if the contrast value has been assigned.
- **Use:** Inside and outside picture loop. It is a good practice to use this procedure not inside the picture loop.
- **Note:** Available with v1.02
- **Example:**
- **See also:** Device Table

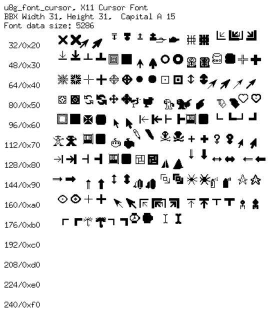
setCursorColor

- **C++/Arduino:** `void U8GLIB::setCursorColor(uint8_t fg, uint8_t bg)`
- **C:** `void u8g_SetCursorColor(u8g_t *u8g, uint8_t fg, uint8_t bg)`
- **Description:** Assign the foreground and background color index for the cursor.
- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
    - `fg`: Foreground color index.
    - `bg`: Background color index.
- **Returns:**
- **Use:** Outside picture loop.
- **Note:**
- **Example:**
- **See also:** enableCursor

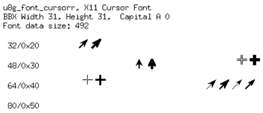
setCursorFont

- **C++/Arduino:** `void U8GLIB::setCursorFont(const u8g_pgm_uint8_t *font)`
- **C:** `void u8g_SetCursorFont(u8g_t *u8g, const u8g_pgm_uint8_t *font)`
- **Description:** Set the cursor font (see note below). The cursor shape can be selected from this font.
- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
    - `font`: A pointer to the font data with cursor shapes.
- **Returns:**
- **Use:** Outside picture loop.
- **Note:**The following cursor fonts are available:

`u8g_font_cursor`



`u8g_font_cursorr`, reduced number of cursor shapes, uses less memory.



u8g_font_cursorr, X11 Cursor Font
BBX Width 31, Height 31, Capital A 0
Font data size: 492
32/0x20
48/0x30
64/0x40
80/0x50

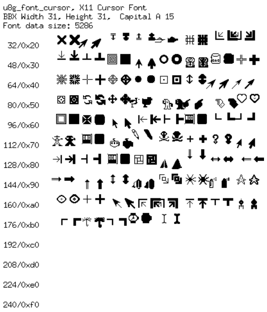
- **See also:** setCursorStyle, enableCursor

setCursorPos

- **C++/Arduino:** `void U8GLIB::setCursorPos(uint8_t x, uint8_t y)`
- **C:** `void u8g_SetCursorPos(u8g_t *u8g, uint8_t x, uint8_t y)`
- **Description:** Draw the enabled cursor at the specified x/y position.
- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
    - `x`: X-position.
    - `y`: Y-position.
- **Returns:**
- **Use:** Outside picture loop.
- **Note:**
- **Example:**
- **See also:** enableCursor

setCursorStyle

- **C++/Arduino:** `void U8GLIB::setCursorStyle(uint8_t encoding)`
- **C:** `void u8g_SetCursorStyle(u8g_t *u8g, uint8_t encoding)`
- **Description:** Set the cursor shape. The cursor shape is defined by two bitmaps of a cursor font. The encoding 32 will select the bitmaps 32 and 33 of a cursor font.In the font `u8g_font_cursor`, this would select the x cursor in the upper left edge:



- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
    - `encoding`: A character position within the cursor font.
- **Returns:**
- **Use:** Outside picture loop.
- **Note:**
- **Example:**
- **See also:** setCursorFont, enableCursor

setDefaultBackgroundColor

setDefaultForegroundColor

setDefaultMidColor

- **C++/Arduino:**

```
void U8GLIB::setDefaultBackgroundColor(void)
void U8GLIB::setDefaultForegroundColor(void)
void U8GLIB::setDefaultMidColor(void)
```

- **C:**

```
void u8g_SetDefaultBackgroundColor(u8g_t *u8g)
void u8g_SetDefaultForegroundColor(u8g_t *u8g)
void u8g_SetDefaultMidColor(u8g_t *u8g)
```

- **Description:** Assign one of the default colors as current color index. On a monochrom display, `setDefaultBackgroundColor` will assign 0 to the current color indexand `setDefaultForegroundColor` will assign 1 to the current color index. For all display types, it is ensured, that `setDefaultBackgroundColor` and`setDefaultForegroundColor` will assign different values.
- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**
- **Use:** Inside and outside picture loop.
- **Note:**
- **Example:**
- **See also:** setColorIndex

setFont

- **C++/Arduino:** `void U8GLIB::setFont(const u8g_fntpgm_uint8_t *font)`
- **C:** `void u8g_SetFont(u8g_t *u8g, const u8g_pgm_uint8_t *font)`
- **Description:** Set the current font and reset the font reference position to "Baseline" (setFontPosBaseline). This font will be used for any further font and draw procedures. U8glib has a lot of built-in fonts which can be used as argument. See here for an overview.
- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
    - `font`: A pointer to the font data.
- **Returns:**
- **Use:** Inside and outside picture loop. It is a good practice to use this procedure at the beginning of the body of the picture loop.
- **Note:**The fonts are loaded into memory as needed. The more fonts you use, the larger your program will be.
- **Example:**
- **See also:** drawStr setFontPosBaseline

setFontLineSpacingFactor

- **C++/Arduino:** `void U8GLIB::setFontLineSpacingFactor(uint8_t factor)`
- **C:** `void u8g_SetFontLineSpacingFactor(u8g_t *u8g,uint8_t factor)`
- **Description:** Assign the factor for the `LineSpacing` calclation.

Line stretch	0.5	0.8	1.0	1.2	1.5	2.0
factor	32	51	64	77	96	128

- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
    - `factor`: See above.
- **Returns:**
- **Use:** Inside and outside picture loop.
- **Note:**
- **See also:** getFontLineSpacing

setFontPosBaseline

setFontPosBottom

setFontPosCenter

setFontPosTop

- **C++/Arduino:**

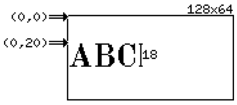
```
void U8GLIB::setFontPosBaseline(void)
void U8GLIB::setFontPosBottom(void)
void U8GLIB::setFontPosCenter(void)
void U8GLIB::setFontPosTop(void)
```

- **C:**

```
void u8g_SetFontPosBaseline(u8g_t *u8g);
void u8g_SetFontPosBottom(u8g_t *u8g);
void u8g_SetFontPosCenter(u8g_t *u8g);
void u8g_SetFontPosTop(u8g_t *u8g);
```

- **Description:** Set the reference position for the character and string draw procedure. In the command `u8g_DrawStr(u8g, 5, 20, "ABCdefg");` the string is placed at (5,20), where (5,20) defines the left start of the baseline if `setFontPosBaseline` has been called (which also is the default).
    - `setFontPosBottom`: Reference position is `getFontDescent()` below baseline.
    - `setFontPosTop`: Reference position is `getFontAscent()+1` above baseline (one pixel above the highest refrence character).
    - `setFontPosCenter`: Reference position centered with respect to `getFontAscent()` and `getFontDescent()`.
- **Arguments:**
    - `u8g` : Pointer to the `u8g` structure (C interface only).
- **Returns:**
- **Use:** Inside and outside picture loop.
- **Note:**
- **Example:**`setFontPosTop` will move the reference point (0,20) for the origin of the text string to the upper left corner of the string.

```
U8GLIB u8g(...)
...
u8g.setFont(u8g_font_osb18);
u8g.setFontPosTop();
u8g.drawStr(0, 20, "ABC");
```



- **See also:** drawStr getFontAscent getFontDescent

setFontRefHeightAll

setFontRefHeightExtendedText

setFontRefHeightText

- **C++/Arduino:**

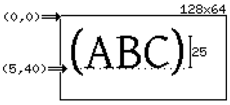
```
void U8GLIB::setFontRefHeightAll(void)
void U8GLIB::setFontRefExtendedHeightText(void)
void U8GLIB::setFontRefHeightText(void)
```

- **C:**

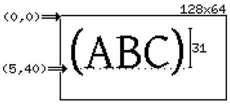
```
void u8g_SetFontRefHeightAll(u8g_t *u8g)
void u8g_SetFontRefHeightExtendedText(u8g_t *u8g)
void u8g_SetFontRefHeightText(u8g_t *u8g)
```

- **Description:** A call to one of these procedure will define the calculation method for the ascent and descent of the current font. This method will be used for the current and all other fonts, which will be set with setFont(). Changing this calculation method has an effect on getFontAscent() and getFontDescent(). It has also an effect on the text position methods other than setFontPosBaseline()

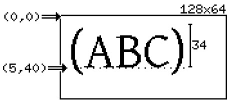
  - setFontRefHeightAll: Ascent will be the highest ascent of all glyphs of the current font. Descent will be the highest descent of all glyphs of the current font.
  - setFontRefHeightExtendedText: Ascent will be the largest ascent of "A", "1" or "(" of the current font. Descent will be the descent of "g" or "(" of the current font (this is the default after startup).
  - setFontRefHeightText: Ascent will be the ascent of "A" or "1" of the current font. Descent will be the descent "g" of the current font.
- **Arguments:**
  - u8g : Pointer to the u8g structure (C interface only).
- **Returns:**
- **Use:** Inside and outside picture loop.
- **Note:**
- **Example:**setFontRefHeightText: Ascent value of u8g_font_gdr25 is 25.



setFontRefHeightExtendedText: Ascent value of u8g_font_gdr25 is 31.



setFontRefHeightAll: Ascent value of u8g_font_gdr25 is 35.



- **See also:** setFontPosBaseline getFontAscent

setHardwareBackup

- **C++/Arduino:** void U8GLIB::setHardwareBackup(u8g_state_cb backup_cb)
- **C:** void u8g_SetHardwareBackup(u8g_t *u8g, u8g_state_cb backup_cb)
- **Description:** The display can be connected to I/O pins which are also shared with other external devices. Examples are SPI, TWI or UART interfaces. **Example:** SD card and SPI displayshare Clock and Data pins (but have different chip select lines). The SD Card software uses the SPI hardware of the microcontroller to access the SD card, but U8glibshould use a software SPI mode. In such a case, the hardware state of the microcontroller SPI subsystem must be modified before access to SD card and display.This modification is activated by this procedure. Usage is: (1) Init u8glib, (2) call this procedure and (3) init other libraries.Available backup procedures:
  - u8g_backup_avr_spi: Backup SPI hardware state of an AVR microcontroller.
- **Arguments:**
  - u8g : Pointer to the u8g structure (C interface only).
  - backup_cb: Hardware state backup procedure (specific to controller and hardware subsystem).
- **Returns:**
- **Use:** Outside picture loop.
- **Note:** Available in v1.05.
- **Example:**U8glib uses software SPI and SD library hardware SPI:

```
U8GLIB_DOGM128 u8g(7, 5, 1, 2);                    // SPI Com: SCK = 7, MOSI =
...
void setup()  {
  ...
  // SPI backup: Avoid conflict between SW-SPI (u8glib) and HW-SPI (SD)
  u8g.setHardwareBackup(u8g_backup_avr_spi);
  ...
  // Setup Arduino SD library
  pinMode(SS, OUTPUT);
  if (SD.begin(23)) {
    mas_Init(mas_device_sd, NULL);
  }
  ...
}
```

- **See also:**

setPrintPos

- **C++/Arduino:** void U8GLIB::setPrintPos(u8g_uint_t x, u8g_uint_t y)
- **C:**
- **Description:** Assignes the (x,y) position for the next call of the print procedure.
- **Arguments:**
  - x: X-position.
  - y: Y-position.
- **Returns:**
- **Use:** Inside picture loop.
- **Example:**
- **See also:** print

setRGB

- **C++/Arduino:** void U8GLIB::setRGB(uint8_t r, uint8_t g, uint8_t b)
- **C:** void u8g_SetRGB(u8g_t *u8g, uint8_t r, uint8_t g, uint8_t b)
- **Description:** Assignes RGB color for one of the color devices.
- **Arguments:**
  - u8g : Pointer to the u8g structure (C interface only).
  - r: Red part of the color, range: 0..255.
  - g: Green part of the color, range: 0..255.
  - b: Blue part of the color, range: 0..255.
- **Returns:**
- **Note:** Available with v1.13
- **Use:** Inside picture loop.
- **Example:**
- **See also:** setColorIndex

setRot90

setRot180

setRot270

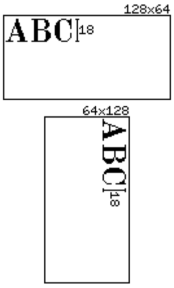
- **C++/Arduino:**

```
void U8GLIB::setRot90()
void U8GLIB::setRot180()
void U8GLIB::setRot270()
```

- **C:**

```
void u8g_SetRot90(u8g_t *u8g)
void u8g_SetRot180(u8g_t *u8g)
void u8g_SetRot270(u8g_t *u8g)
```

- **Description:** Clockwise rotates the display screen by 90, 180 or 270 degree. For most display devices, landscape view is the default mode.Rotation by 90 or 270 degree will put the display into portrait mode.
- **Arguments:**
  - u8g : Pointer to the u8g structure (C interface only).
- **Returns:**
- **Use:** Outside picture loop. Arduino environment: It is a good practice to use this procedure in the setup() procedure.
- **Example:**Left: Default landscape mode. Right: Portrait mode with setRot90.

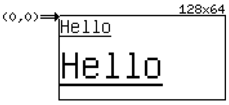


- **See also:** undoRotation

setScale2x2

- **C++/Arduino:** void U8GLIB::setScale2x2()
- **C:** void u8g_SetScale2x2(u8g_t *u8g)
- **Description:** This command halfs x and y dimension of the display. After calling this command, graphics commands outputblocks of size 2x2 pixel until a call to u8g:undoScale().getHeight() and getWidth() only return half of the original display values.All graphic commands between "u8g::setScale2x2()" and "u8g:undoScale()" are scaled up (line draw, pixel set, font, bitmaps, ...).
- **Arguments:**
  - u8g : Pointer to the u8g structure (C interface only).
- **Returns:**
- **Note:** Available with v1.09
- **Use:** Outside or inside picture loop. If used inside the picture loop, also call u8g:undoScale() inside the picture loop.
- **Example:**

```
void draw(void) {
  u8g.setFont(u8g_font_unifont);
  u8g.setFontPosTop();
  u8g.drawStr(0, 1, "Hello");
  u8g.drawHLine(0, 1+14, 40);
  u8g.setScale2x2();                    // Scale up all draw procedures
  u8g.drawStr(0, 12, "Hello");          // actual display position is (0,24)
  u8g.drawHLine(0, 12+14, 40);          // All other procedures are also affec
  u8g.undoScale();                      // IMPORTANT: Switch back to normal mc
}
```



- **See also:** undoScale

sleepOn

sleepOff

- **C++/Arduino:** ``

```
void U8GLIB::sleepOn(void)
void U8GLIB::sleepOff(void)
```

- **C:** ``

```
void u8g_SleepOn(u8g_t *u8g)
void u8g_SleepOff(u8g_t *u8g)
```

- **Description:** Enable/disable sleep mode for the display (if possible).
- **Arguments:**
  - u8g : Pointer to the u8g structure (C interface only).
- **Returns:**
- **Use:** Outside picture loop.
- **Note:**Available with v1.11.Supported for ST7565 and SSD13xx controller.
- **Example:**

undoRotation

- **C++/Arduino:** void U8GLIB::undoRotation()
- **C:** void u8g_UndoRotation(u8g_t *u8g)
- **Description:** Remove an applied rotation done by the "setRotXY" commands. After calling this command, the display will have its default orientation. Nothing happens if the display has default orientation.
- **Arguments:**
  - u8g : Pointer to the u8g structure (C interface only).
- **Returns:**
- **Use:** Outside picture loop.
- **Example:**
- **See also:** setRot90

undoScale

- **C++/Arduino:** void U8GLIB::undoScale()
- **C:** void u8g_UndoScale(u8g_t *u8g)
- **Description:** Remove an applied scaling. If the scaling has been applied within the body of the picture loop, then this command should be called within the body of the picture loop.
- **Arguments:**
  - u8g : Pointer to the u8g structure (C interface only).
- **Returns:**
- **Note:** Available with v1.09
- **Example:**
- **See also:** setScale2x2

U8GLIB

- **C++/Arduino:** ``

```
void U8GLIB::U8GLIB(u8g_dev_t *dev)
void U8GLIB::U8GLIB(u8g_dev_t *dev, uint8_t sck, uint8_t mosi, uint8_t cs, uir
void U8GLIB::U8GLIB(u8g_dev_t *dev, uint8_t d0, uint8_t d1, uint8_t d2, uint8_
        uint8_t en, uint8_t cs1, uint8_t cs2, uint8_t di, uint8_t rw, uint8_t
```

- **Description:** Create a new interface to a graphics display. The dev argument describes the type of the display. See here for a complete list of available devices.Usually this constructor is not called directly. Instead there are drived classes for each available device. See also the last column of the device listfor a complete list of available constructor calls.
- **Arguments:**
  - dev: A pointer to a device structure.
  - Arduino pins: Required pins to connect the display depending on the communication interface.
  - reset: The reset pin is optional and can be U8G_PIN_NONE
- **Returns:**
- **Use:** Outside picture loop.
- **Note:**
- **Example:**
- **See also:** List of supported devices