Table of Contents

- drawRFrame
- drawString
- drawTetragon
- drawTriangle
- drawVLine
- getFontAscent
- getFontDescent
- getHeight
- getStrWidth
- getUcg
- getWidth
- print
- setClipRange
- setColor
- setFont
- setFontMode
- setFontPosBaseline
- setFontPosBottom
- setFontPosCenter
- setFontPosTop
- setMaxClipRange
- setPrintDir

Introduction

This is the “Hello World” example for Ucglib:

```
#include <SPI.h>
#include "Ucglib.h"

Ucglib_ST7735_18x128x160_SWSPI ucg(/*sclk=*/ 13, /*data=*/ 11, /*cd=*/ 9 , /*cs=*/ 10, /*res

void setup(void){
  delay(1000);
  ucg.begin(UCG_FONT_MODE_TRANSPARENT);
  ucg.clearScreen();
}

void loop(void){
  ucg.setRotate90();
  ucg.setFont(ucg_font_ncenR14_tr);
  ucg.setPrintPos(0,25);
  ucg.setColor(255, 255, 255);
  ucg.print("Hello World!");

  delay(500);
}
```

The constructor in line

```
Ucglib_ST7735_18x128x160_SWSPI ucg(/*sclk=*/ 13, /*data=*/ 11, /*cd=*/ 9 , /*cs=*/ 10, /*res
```

will setup Ucglib for a display with ST7735 controller. A complete list of available constructors is here.

Migration from v1.01 to v1.02

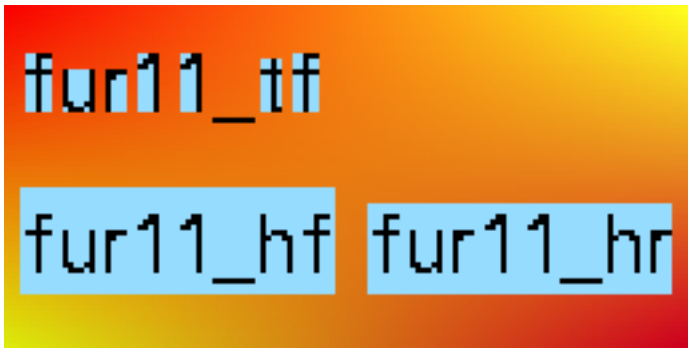
Rename Fonts

All existing fonts have to be renamed:

- If font ends in `r`, replace `r` with `_tr`
- If font ends in `n`, replace `n` with `_tn`
- For all other fonts: add `_tf`

Font Mode UCG_FONT_MODE_SOLID

This mode is still supported, however, this mode requires special fonts. `_tf`, `_tr` and `_tn` will not look very nicely, use `_hf`, `_hr`, `_hn` or `_mf`, `_mr` and `_mn` (whatever is avialable).



Functions setFontMode() and begin()

`UCG_FONT_MODE_NONE` is not supported any more. It is still there, but behaves like
`UCG_FONT_MODE_TRANSPARENT`

User Reference Manual

begin

- **C++ Prototype:**

```
uint8_t Ucglib::begin(ucg_font_mode_fnptr fontmode)
```

- **Description:** Send init code to the display, setup internal data structures and define the text output mode. Use this command in the `setup()` of the Arduino ".ino" file. The font mode can be changed later with the setFontMode command. However, using both methods will increase the code size of the library. If text output is not required at all, then using `UCG_FONT_MODE_NONE` will greatly reduce code size.

- **Arguments:**
  - `fontmode`: `UCG_FONT_MODE_TRANSPARENT`, `UCG_FONT_MODE_SOLID` or `UCG_FONT_MODE_NONE`

- **Returns:** 0, if the init procedure fails.

- **See also:** setFontMode

- **Note:** `UCG_FONT_MODE_NONE` is obsolete in v1.02

- **Example:**

```
Ucglib4WireSWSPI ucg(ucg_dev_ssd1351_18x128x128_ilsoft, ucg_ext_ssd1351_18, /*sclk=*/ 7
void setup(void) {
  ucg.begin(UCG_FONT_MODE_TRANSPARENT);
  ucg.clearScreen();
}
void loop(void) {
  ucg.setFont(ucg_font_ncenR14_tr);
  ucg.setPrintPos(0,25);
  ucg.setColor(255, 255, 255);
  ucg.print("Hello World!");
  delay(500);
}
```

clearScreen

- **C++ Prototype:**

```
void Ucglib::clearScreen(void)
```

- **Description:** Clear the screen and reset the clip range to maximum.

- **Arguments:** None.

- **Returns:** -

- **See also:** begin

- **Example:**

```
Ucglib4WireSWSPI ucg(ucg_dev_ssd1351_18x128x128_ilsoft, ucg_ext_ssd1351_18, /*sclk=*/ 7
void setup(void) {
  ucg.begin(UCG_FONT_MODE_TRANSPARENT);
  ucg.clearScreen();
}
void loop(void) {
  ucg.setFont(ucg_font_ncenR14_tr);
  ucg.setPrintPos(0,25);
  ucg.setColor(255, 255, 255);
  ucg.print("Hello World!");
  delay(500);
}
```

drawBox

- **C++ Prototype:**

```
void Ucglib::drawBox(ucg_int_t x, ucg_int_t y, ucg_int_t w, ucg_int_t h)
```

- **Description:** Draw a filled box. Use current color from index 0. The top-left pixel is at `x`,`y`. The box has width of `w` and the height is `h` pixel.

- **Arguments:**
  - `x`, `y`: Top-left position of the box.

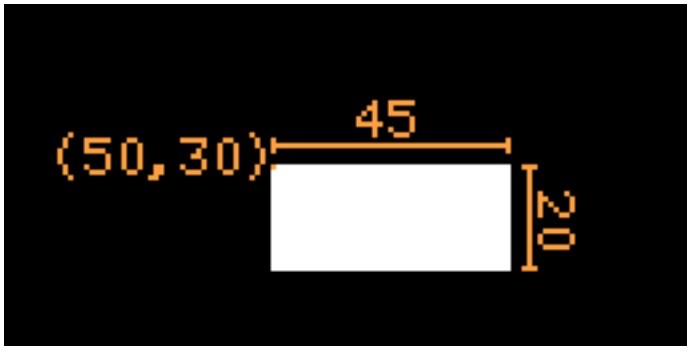
- `w`, `h`: Width and height of the box.

- **Returns:** -

- **See also:** setColor drawFrame

- **Example:**

```
ucg.setColor(255, 255, 255);
ucg.drawBox(50, 30, 45, 20);
```



drawCircle

- **C++ Prototype:**

```
void Ucglib::drawCircle(ucg_int_t x0, ucg_int_t y0, ucg_int_t rad, uint8_t option)
```

- **Description:** Draw a full circle or a quarter of a circle. Use current color from index 0. The center of the circle is at `x0`,`y0`. The circle has a diameter of `2*rad+1` pixel.

- **Arguments:**

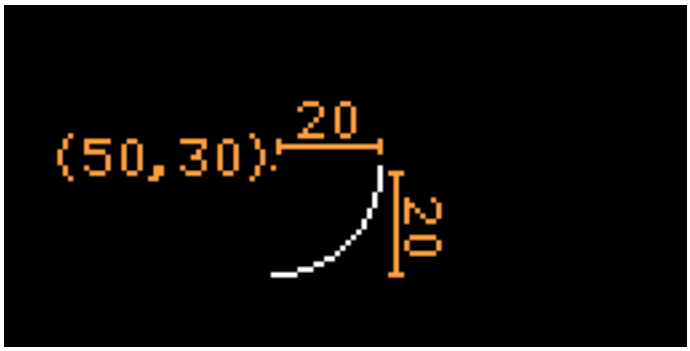
  - `x0`, `y0`: Center of the circle.

  - `rad`: Radius of the circle.

  - `option`: One of the following constants: `UCG_DRAW_ALL`, `UCG_DRAW_UPPER_RIGHT`, `UCG_DRAW_UPPER_LEFT`, `UCG_DRAW_LOWER_LEFT`, `UCG_DRAW_LOWER_RIGHT`

- **Returns:** -
- **See also:** setColor drawDisc
- **Example:**

```
ucg.setColor(255, 255, 255);
ucg.drawCircle(50, 30, 20, UCG_DRAW_LOWER_RIGHT);
```



drawDisc

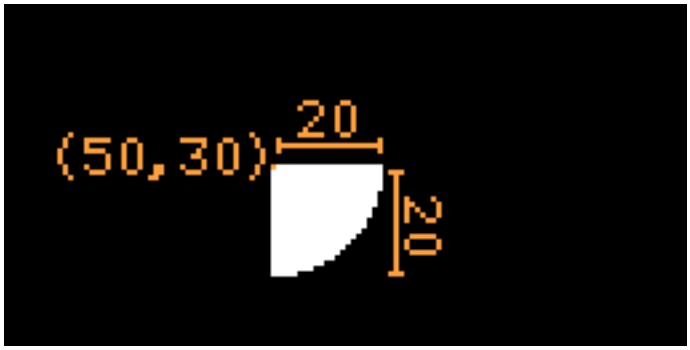
- **C++ Prototype:**

```
void Ucglib::drawDisc(ucg_int_t x0, ucg_int_t y0, ucg_int_t rad, uint8_t option)
```

- **Description:** Draw a filled full circle or a quarter of a filled circle. Use current color from index 0. The center of the filled circle is at `x0`,`y0`. The filled circle has a diameter of `2*rad+1` pixel.
- **Arguments:**
  - `x0`, `y0`: Center of the filled circle.
  - `rad`: Radius of the filled circle.
  - `option`: One of the following constants: `UCG_DRAW_ALL`, `UCG_DRAW_UPPER_RIGHT`, `UCG_DRAW_UPPER_LEFT`, `UCG_DRAW_LOWER_LEFT`, `UCG_DRAW_LOWER_RIGHT`

- **Returns:** -
- **See also:** setColor drawCircle
- **Example:**

```
ucg.setColor(255, 255, 255);
ucg.drawDisc(50, 30, 20, UCG_DRAW_LOWER_RIGHT);
```



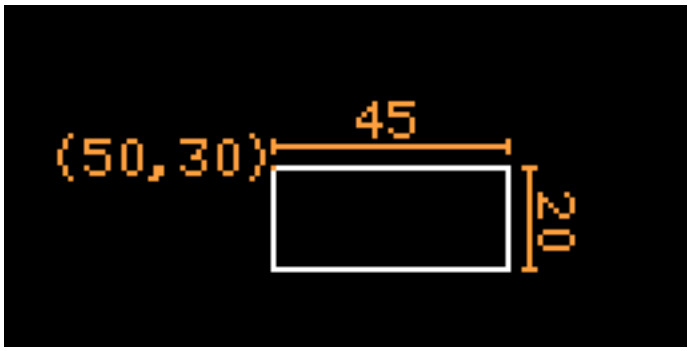
drawFrame

- **C++ Prototype:**

```
void Ucglib::drawFrame(ucg_int_t x, ucg_int_t y, ucg_int_t w, ucg_int_t h)
```

- **Description:** Draw a rectangle. Use current color from index 0. The top-left pixel is at `x`,`y`. The rectangle has width of `w` and the height is `h` pixel.
- **Arguments:**
  - `x`, `y`: Top-left position of the rectangle.
  - `w`, `h`: Width and height of the rectangle.
- **Returns:** -
- **See also:** setColor drawRFrame
- **Example:**

```
ucg.setColor(255, 255, 255);
ucg.drawFrame(50, 30, 45, 20);
```



drawGlyph

- **C++ Prototype:**

  ```
  ucg_int_t Ucglib::drawGlyph(ucg_int_t x, ucg_int_t y, uint8_t dir, uint8_t encoding)
  ```

- **Description:** Draw a single character. Use current color from index 0. If the setFontMode is `UCG_FONT_MODE_SOLID`, then the background color is defined by color index 1 (`ucg.setColor(1, r, g, b)`).

- **Arguments:**
  - `x`, `y`: Reference position of the character.
  - `dir`: One of the values 0 (left to right), 1 (top down), 2 (right left) or 3 (bottom up)
  - `encoding`: Code number of the character.

- **Returns:** Width of the gylph.

- **See also:** setColor print

drawGradientBox

- **C++ Prototype:**

```
void Ucglib::drawGradientBox(ucg_int_t x, ucg_int_t y, ucg_int_t w, ucg_int_t h)
```

- **Description:** Draw a filled box. The upper left position is at `x`,`y`. Dimensions of the box are `w` (width) and `h` (height) pixel. The pixel at the upper left will have the color from index 0, upper right pixel has color from index 1, lower left from index 2 and lower right from index 3. The remaining colors will be interpolated between the four colors.

- **Arguments:**
  - `x`, `y`: Start position of the line.
  - `w`, `h`: Width and height of the box.

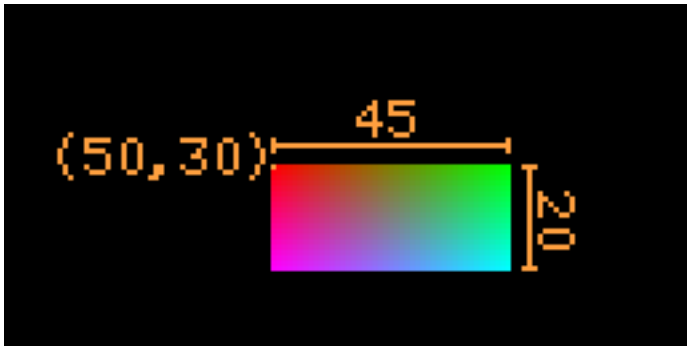
- **Returns:** -

- **See also:** setColor

- **Note**: This procedure is part of the Ucglib extension. It must be activated during library setup.

- **Example:**

```
ucg.setColor(0, 255, 0, 0);
ucg.setColor(1, 0, 255, 0);
ucg.setColor(2, 255, 0, 255);
ucg.setColor(3, 0, 255, 255);
ucg.drawGradientBox(50, 30, 45, 20);
```



drawGradientLine

- **C++ Prototype:**

```
void Ucglib::drawGradientLine(ucg_int_t x, ucg_int_t y, ucg_int_t len, ucg_int_t dir)
```

- **Description:** Draw a horizontal or vertical line. The line will start at `x`,`y` and has a total of `len` pixel. The pixel at `x`,`y` will have the color from index 0. The color will be changed until it matches the color of index 1.

- **Arguments:**
  - `x`, `y`: Start position of the line.
  - `len`: Length of the line.
  - `dir`: One of the values 0 (left to right), 1 (top down), 2 (right left) or 3 (bottom up)

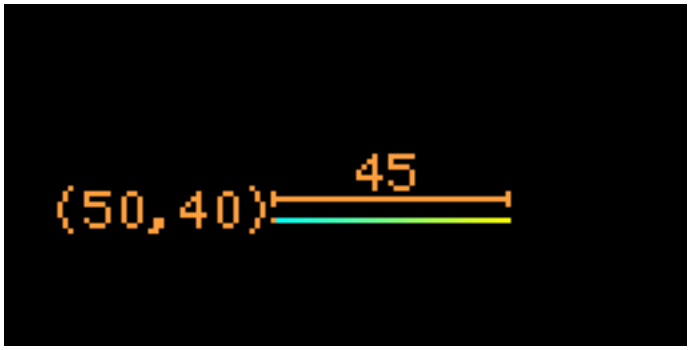
- **Returns:** -

- **See also:** setColor

- **Note**: This procedure is part of the Ucglib extension. It must be activated during library setup.

- **Example:**

```
ucg.setColor(0, 0, 255, 255);
ucg.setColor(1, 255, 255, 0);
ucg.drawGradientLine(50, 40, 45, 0);
```



drawHLine

- **C++ Prototype:**

  ```
  void Ucglib::drawHLine(ucg_int_t x, ucg_int_t y, ucg_int_t len)
  ```

- **Description:** Draw a horizontal line. Use current color from index 0. The leftmost pixel is at `x`,`y` and the rightmost pixel is at `x+len-1`,`y`,

- **Arguments:**

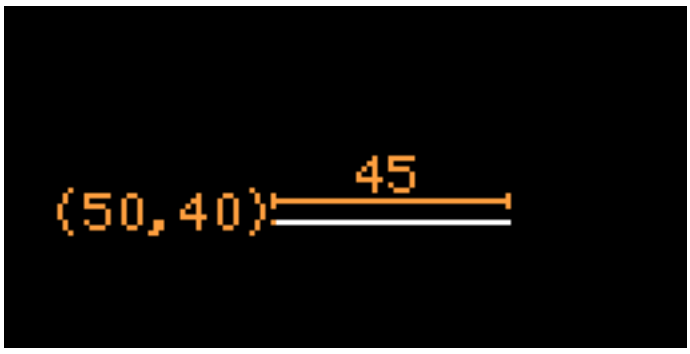
  - `x`, `y`: Left position of the horizontal line.
  - `len`: Length of the line.

- **Returns:** -

- **See also:** setColor drawVLine

- **Example:**

  ```
  ucg_SetColor(ucg, 0, 255, 255, 255);
  ucg_DrawHLine(ucg, 50, 40, 45);
  ```



drawLine

- **C++ Prototype:**

  ```
  void Ucglib::drawLine(ucg_int_t x1, ucg_int_t y1, ucg_int_t x2, ucg_int_t y2)
  ```

- **Description:** Draw a line from pixel at x1,y1 to pixel x2,y2. Use current color from index 0.

- **Arguments:**
  - `x1`, `y1`: Start pixel of the line.
  - `x2`, `y2`: End pixel of the line.

- **Returns:** -

- **See also:** setColor drawVLine

drawPixel

- **C++ Prototype:**

```
void Ucglib::drawPixelucg_int_t x, ucg_int_t y)
```

- **Description:** Draw a pixel at position x,y. Use current color from index 0.

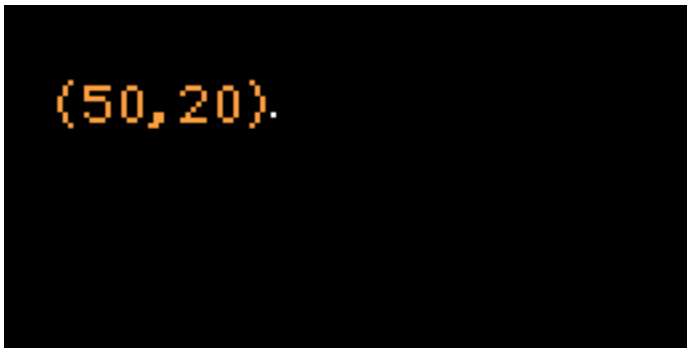
- **Arguments:**
  - `x`, `y`: Position of pixel.

- **Returns:** -

- **See also:** setColor

- **Example:**

```
ucg_SetColor(ucg, 0, 255, 255, 255);
ucg_DrawPixel(ucg, 50, 20);
```



drawRBox

drawRFrame

- **C++ Prototype:**

```
void Ucglib::drawRBox(ucg_int_t x, ucg_int_t y, ucg_int_t w, ucg_int_t h, ucg_int_t r)
void Ucglib::drawRFrame(ucg_int_t x, ucg_int_t y, ucg_int_t w, ucg_int_t h, ucg_int_t r
```

- **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)`. Use current color from index 0.

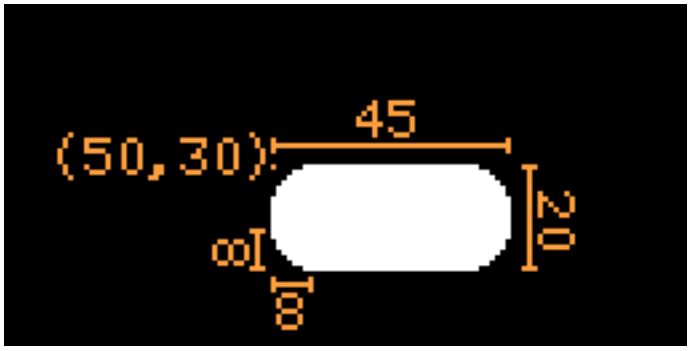
- **Arguments:**
  - `x`, `y`: Position of upper left edge.
  - `w`: Width of the box or frame.
  - `h`: Height of the box or frame.
  - `r`: Radius for the four edges.

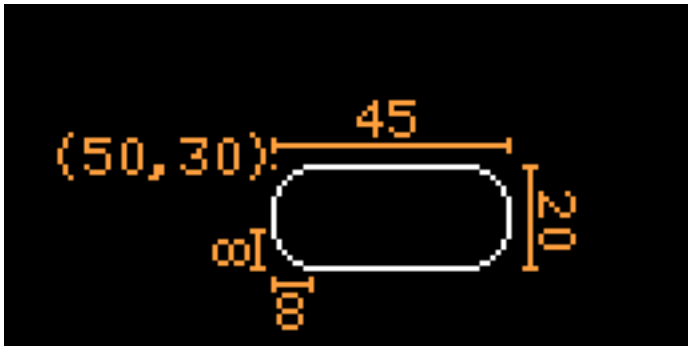
- **Returns:** -

- **See also:** setColor drawBox drawFrame

- **Example:**

```
ucg.setColor(255, 255, 255);
ucg.drawRBox(50,30, 45,20, 8);  // left picture
ucg.drawRFrame(50,30, 45,20, 8);  // right picture
```





drawString

- **C++ Prototype:**

```
ucg_int_t drawString(ucg_int_t x, ucg_int_t y, uint8_t dir, const char *str)
```

- **Description:** Draw a string. Use current color from index 0. If the setFontMode is `UCG_FONT_MODE_SOLID`, then the background color is defined by color index 1 (`ucg.setColor(1, r, g, b)`).

- **Arguments:**
  - `x`, `y`: Reference position of the string.

  - `dir`: One of the values 0 (left to right), 1 (top down), 2 (right left) or 3 (bottom up)
  - `str`: String, which will be printed on the screen.

- **Returns:** The width of the string.
- **See also:** setColor print

drawTetragon

- **C++ Prototype:**

```
void Ucglib::drawTetragon(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2, i
```

- **Description:** Draw a filled tetragon (a shape with four corners), defined by four edge points. Use current color from index 0.
- **Arguments:**
  - `x0`, `y0`: Point 0 of the tetragon.
  - `x1`, `y1`: Point 1 of the tetragon.
  - `x2`, `y2`: Point 2 of the tetragon.
  - `x3`, `y3`: Point 3 of the tetragon.
- **Returns:** -
- **See also:** setColor
- **Note 1:** `int16_t` argument type might be changed to `ucg_int_t` in the future.
- **Note 2:** This procedure will only draw "simple"/convex tetragons. The result will be undefined, if the tetragon is not convex.

drawTriangle

- **C++ Prototype:**

```
void Ucglib::drawTriangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2, i
```

- **Description:** Draw a filled triangle, defined by three edge points. Use current color from index 0.

- **Arguments:**
  - `x0`, `y0`: Point 0 of the triangle.
  - `x1`, `y1`: Point 1 of the triangle.
  - `x2`, `y2`: Point 2 of the triangle.

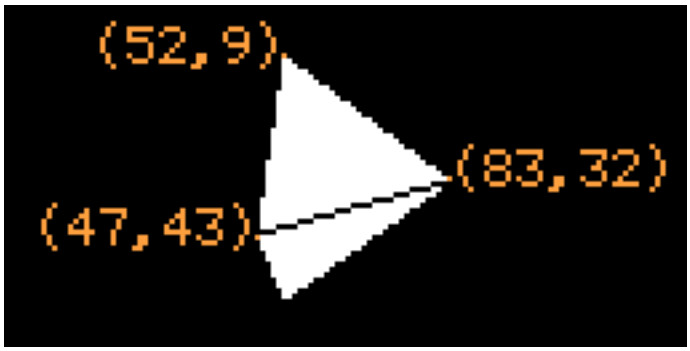
- **Returns:** -

- **See also:** setColor

- **Note:** `int16_t` argument type might be changed to `ucg_int_t` in the future.

- **Example:**

```
ucg.setColor(0, 255, 255, 255);
ucg.drawTriangle(52, 9, 83,32,   47,42);
ucg.drawTriangle(52,55, 83,32+1, 47,42+1);
```



drawVLine

- **C++ Prototype:**

```
void Ucglib::drawVLine(ucg_int_t x, ucg_int_t y, ucg_int_t len)
```

- **Description:** Draw a vertical line. Use current color from index 0. The topmost pixel is at `x`,`y`. The bottom pixel is at `x`,`y+len-1`,

- **Arguments:**
  - `x`, `y`: Top position of the vertical line.
  - `len`: Length of the line.

- **Returns:** -

- **See also:** setColor drawHLine

- **Example:**

```
ucg_SetColor(ucg, 0, 255, 255, 255);
ucg_DrawVLine(ucg, 50, 20, 25);
![ref/draw_vline.png](ref/draw_vline.png)
```

getFontAscent

- **C++ Prototype:**

```
void Ucglib::getFontAscent(void)
```

- **Description:** Returns the height of the character 'A' or the number '1' above the baseline. For the font in the example below, getFontAscent returns the value 24.

- **Arguments:** None.

- **Returns:** The height of the font.

- **See also:** getFontDescent

- **Example:**

getFontDescent

- **C++ Prototype:**

```
void Ucglib::getFontDescent(void)
```

- **Description:** Some glphys of a font might are extended below the baseline (‘g’ or ‘)’). This procedure returns the extension of these characters above baseline. If the extension is below the baseline (which is usually the case) then a negative number is returned. In the example below, the returned descent value is “-7”.

- **Arguments:** None.

- **Returns:** The extension of the characters below the baseline.

- **See also:** getFontAscent

- **Example:**

getHeight

- **C++ Prototype:**

  ```
  ucg_int_t Ucglib::getHeight(void)
  ```

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

- **Arguments:** None.

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

- **See also:** getWidth

getStrWidth

- **C++ Prototype:**

  ```
  ucg_int_t getStrWidth(const char *s)
  ```

- **Description:** Returns the number of pixels, required for the text in `*s` with the current font settings. Some extra pixels are added in front and after the text as defined in the current font.

- **Arguments:** None.
  - `s`: A pointer to a memory location with the text.

- **Returns:** Width of the text in pixel.

- **See also:** setFont

getUcg

- **C++ Prototype:**

  `ucg_t *Ucglib::getUcg(void)`

- **Description:** Returns a pointer to the internal c-structure.

- **Arguments:** None.

- **Returns:** Pointer to ucg_t structure.

- **See also:** -

getWidth

- **C++ Prototype:**

  `ucg_int_t Ucglib::getWidth(void)`

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

- **Arguments:** None.

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

- **See also:** getHeight

print

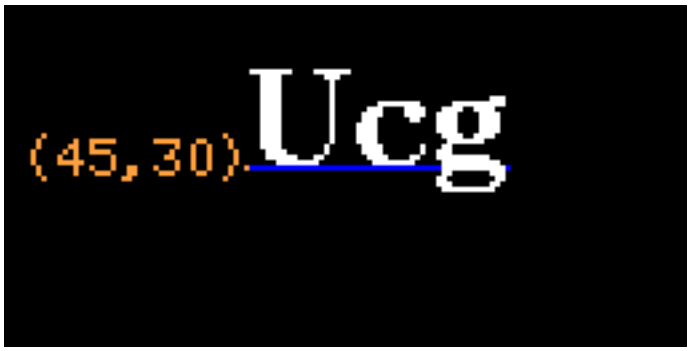
- **C++ Prototype:**

  `size_t Ucglib::print()`

- **Description:** Print text or values. The print function is identical to `Serial.print` from the Arduino Environment (see Serial.print). The position of the output is defined by `setPrintPos()`, the writing direction is defined by `setPrintDir()`. Values and text will have the current color from index 0. If the setFontMode is `UCG_FONT_MODE_SOLID`, then the background color is defined by color index 1 (`ucg.setColor(1, r, g, b)`)

- **Arguments:** None.
- **Returns:** Number of characters.
- **See also:** setPrintDir setPrintPos Serial.print
- **Example:**

```
ucg.setFontPosBaseline();
ucg.setFont(ucg_font_ncenB18_tf);
ucg.setColor(255, 255, 255);
ucg.setPrintPos(45,30);
ucg.setPrintDir(0);
ucg.print("Ucg");
```



setClipRange

- **C++ Prototype:**

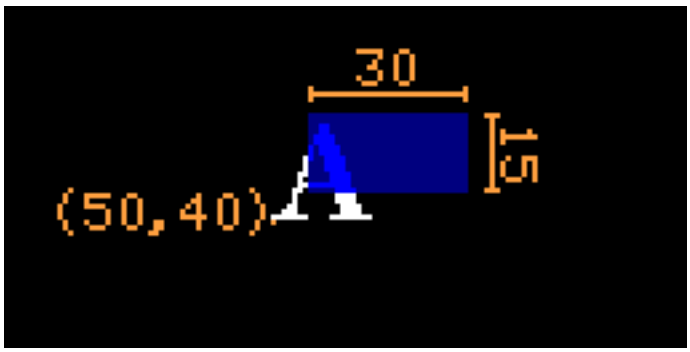
```
void Ucglib::setClipRange(ucg_int_t x, ucg_int_t y, ucg_int_t w, ucg_int_t h)
```

- **Description:** Defines the drawing area for all other commands. Graphics commands are "clipped" against this area. By default this is the complete visible area of the screen. The drawing area must not be larger than the physical display size. Some other commands reset the drawing area to the size of the display (begin, setRotate90).
- **Arguments:**
  - `x`, `y`: Upper left position of the drawing area.

– `w`, `h`: Width and height of the drawing area. Behavior is undefined for `w=0` or `h=0`.

- **Returns:** -
- **See also:** undoClipRange
- **Example:**

```
ucg_SetFontPosBaseline(ucg);
ucg_SetFont(ucg, ucg_font_ncenB18_tf);
ucg_SetColor(ucg, 0, 255, 255, 255);        /* draw white A */
ucg_DrawGlyph(ucg, 50, 40, 0, 'A');
ucg_SetClipRange(ucg, 57, 20, 30, 15);      /* restrict area */
ucg_SetColor(ucg, 0, 0, 0, 127);
ucg_DrawBox(ucg, 0, 0, 128, 64);        /* fill the restricted area with dark blue */
ucg_SetColor(ucg, 0, 0, 0, 255);
ucg_DrawGlyph(ucg, 50, 40, 0, 'A');     /* draw light blue A */
```



setColor

- **C++ Prototype:**

```
void Ucglib::setColor(uint8_t idx, uint8_t r, uint8_t g, uint8_t b)
void Ucglib::setColor(uint8_t r, uint8_t g, uint8_t b)
```

- **Description:** Defines up to four different colors for the graphics primitives. Most commands will use the color at index position 0. If the index (first argument) is skipped, then the color is stored as index 0.

- **Arguments:**
  - `idx`, `y`: Upper left position of the drawing area.
  - `r`, `g`, `b`: Red, green and blue component of the color. Color range is always from 0 to 255 for each of the color components.
- **Returns:** -
- **See also:**
- **Example:**

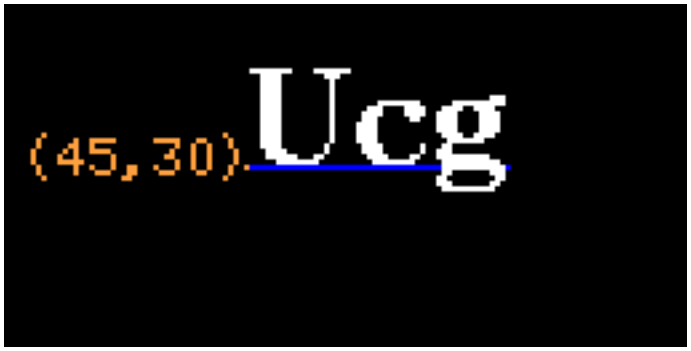
setFont

- **C++ Prototype:**

```
void Ucglib::setFont(const ucg_fntpgm_uint8_t *font)
```

- **Description:** Set the current font for all text and print procedures.
  All fonts will start with `ucg_font_` and will have a two character postfix (since v1.02).
  These two characters
  indicate (1) the purpose of the font (`t` transparent, `m` monospace, see below for
  a complete list) and (2) the number of glyphs in the font (`f` full, `r` reduced, see below).
- **Arguments:**
  - `font`: Pointer to the ucg font. Fonts, which are part of Ucglib start `ucg_font_`. A list of all available fonts is here.
- **Returns:** 0, if the init procedure fails.
- **See also:** List of all available Ucg fonts print
- **Note:** See also Arduino example "HowToUseFonts"
- **Example:**

```
ucg.setFont(ucg_font_ncenB18_tf);
ucg.setPrintPos(45,10);
ucg.print("Ucg");
```



Version v1.02 has a new font naming convention. Second to last character of the font
name is the “Font Purpose” and the last character indicates the “Font Char Set”.

Font Purpose	Description
t	Transparent font, use together with UCG_FONT_MODE_TRANSPARENT. t is identical to h but font
h	All glyphs have common height, use together with UCG_FONT_MODE_TRANSPARENT and UCG_FO
m	All glyphs have common height and width (monospace), use together with UCG_FONT_MODE_T
8	All glyphs have common height and width, use together with UCG_FONT_MODE_TRANSPARENT a

Font Char Set	Description
f	The font includes all glyphs of the original font (up to 256).
r	Only glyphs on the range of the ASCII codes 32 to 127 are included in the font.
n	Only numbers and extra glyphs for writing date and time strings are included in the font.

Example	Description
ucg_font_helvB08_tf	Helvetica bold font for UCG_FONT_MODE_TRANSPARENT which includes ASCII coo
ucg_font_inr53_mr	Monospace font rendered from Inconsolata true type font with restricted glyph
ucg_font_amstrad_cpc_8f	A monospace font, where glyphs fit into 8x8 pixel block

setFontMode

- **C++ Prototype:**

  ```
  void Ucglib::setFontMode(ucg_font_mode_fnptr fontmode)
  ```

- **Description:** Define the text output mode.
  See examples below for the difference between `UCG_FONT_MODE_TRANSPARENT` and `UCG_FONT_MODE_SOLID`.
  Do not use transparent fonts with `UCG_FONT_MODE_SOLID`.

- **Arguments:**

  - `fontmode`: `UCG_FONT_MODE_TRANSPARENT`, `UCG_FONT_MODE_SOLID` or `UCG_FONT_MODE_NONE` (obsolete since v1.02)

- **Returns:** 0, if the init procedure fails.

- **See also:** begin setFont

- **Note:** `UCG_FONT_MODE_NONE` is obsolete in v1.02

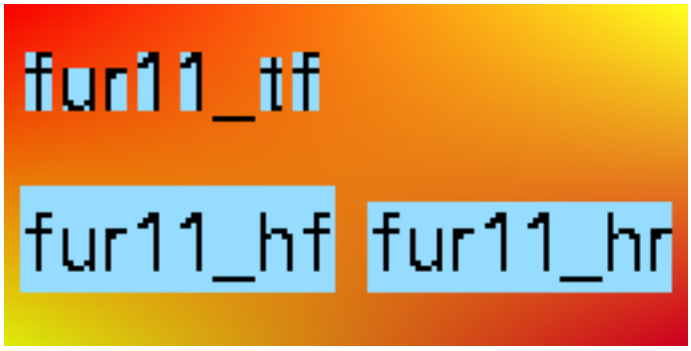
- **Example 1:** `UCG_FONT_MODE_TRANSPARENT`

```
ucg.setFont(ucg_font_ncenB18_tf);
ucg.setFontMode(UCG_FONT_MODE_TRANSPARENT);
ucg.setPrintPos(42, 40);
ucg.setColor(0, 0, 0, 0);       // black color for the text
ucg.print("Ucg");
```

- **Example 2:** `UCG_FONT_MODE_SOLID`

```
ucg.setFont(ucg_font_ncenB18_hf); // do not use _tf for UCG_FONT_MODE_SOLID
ucg.setFontMode(UCG_FONT_MODE_SOLID);
ucg.setPrintPos(42, 40);
ucg.setColor(0, 0, 0, 0);       // black color for the text
ucg.setColor(1, 150, 220, 255); // light blue for the background
ucg.print("Ucg");
```

- **Example 3:** The following picture shows, that transparent (`_tf`) fonts will not look good
with the `UCG_FONT_MODE_SOLID` mode (upper left). Also note, that the common height of the glyphs depend
on the number of glyphs in the font (lower left vs. lower right).



setFontPosBaseline

setFontPosBottom

setFontPosCenter

setFontPosTop

- **C++ Prototype:**

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

- **Description:** Change the reference position for the character output procedures print, drawString and drawGlyph.

- **Arguments:** -

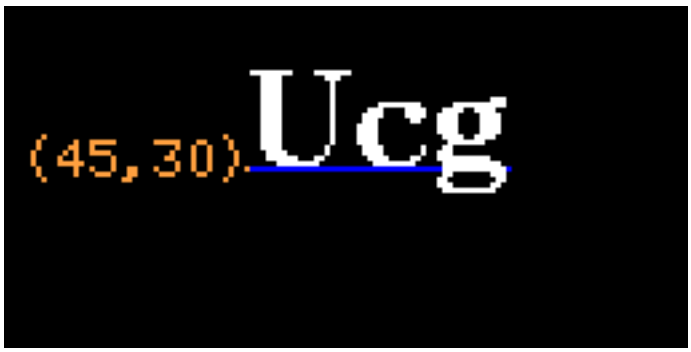
- **Returns:** -

- **See also:** print

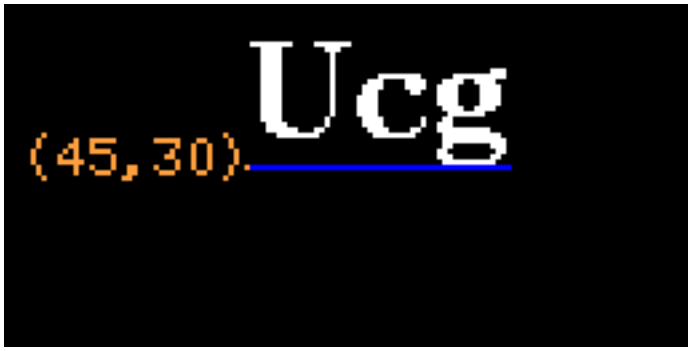
- **Examples:**

Pictures below demonstrate the position of the text "Ucg" with respect to the reference position.

```
ucg.setFontPosBaseline();
```



```
ucg.setFontPosBottom();
```




```
ucg.setFontPosCenter();
```


```
ucg.setFontPosTop();
```

setMaxClipRange

- **C++ Prototype:**

```
void Ucglib::setMaxClipRange(void)
```

- **Description:** This command will reset the clip area to the full display size. It will undo any settings from setClipRange.

- **Arguments:** -

- **Returns:** -

- **See also:** setClipRange

setPrintDir

- **C++ Prototype:**

```
void Ucglib::setPrintDir(uint8_t dir)
```

- **Description:** This command will set the print direction for the following "print" commands.

- **Arguments:**

  - `dir`: One of the values 0 (left to right), 1 (top down), 2 (right left) or 3 (bottom up).

- **Returns:** -

- **See also:** print setPrintPos setFont

- **Example:**

```
ucg.setFontPosBaseline();
ucg.setFont(ucg_font_ncenB18_tf);
ucg.setColor(255, 255, 255);
ucg.setPrintPos(45,10);
ucg.setPrintDir(1);
ucg.print("Ucg");
```

setPrintPos

- **C++ Prototype:**

  ```
  void Ucglib::setPrintPos(ucg_int_t x, ucg_int_t y)
  ```

- **Description:** This command will set the position for next "print" command.

- **Arguments:**

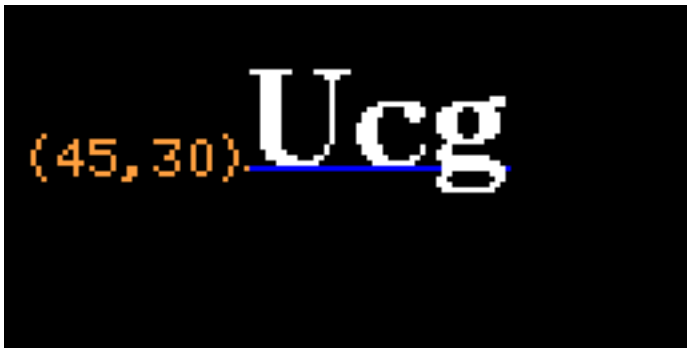
  - `x`, `y`: Reference position for the characters of the next print command.

- **Returns:** -

- **See also:** print

- **Example:**

  ```
  ucg.setFontPosBaseline();
  ucg.setFont(ucg_font_ncenB18_tf);
  ucg.setColor(255, 255, 255);
  ucg.setPrintPos(45,30);
  ucg.setPrintDir(0);
  ucg.print("Ucg");
  ```



setRotate90

setRotate180

setRotate270

- **C++ Prototype:**

  ```
  void Ucglib::setRotate90(void)
  ```

- **Description:** Rotate the screen by 90, 180 or 270 degree. Depending on the aspect ratio of the display, this will put the display in portrait or landscape mode.

- **Arguments:** -

- **Returns:** -

- **See also:** undoRotate

- **Example:**

setScale2x2

- **C++ Prototype:**

  ```
  void Ucglib::setScale2x2(void)
  ```

- **Description:** Scale everything by 2. This includes position values, lines, fonts, circles, etc.
- **Arguments:** -
- **Returns:** -
- **Note**: As long as scaling is active, the screen rotation commands must not be used.
- **See also:** undoScale
- **Example:**

```
ucg.setFontPosBaseline();
ucg.setFont(ucg_font_ncenB18_tf);
ucg.setColor(255, 255, 255);
ucg.setScale2x2();
ucg.setPrintPos(20,20);  // This is position (40,40) on the screen
ucg.setPrintDir(0);
ucg.print("Ucg");
ucg.undoScale();
```

undoClipRange

- **C++ Prototype:**

```
void Ucglib::undoClipRange(void)
```

- **Description:** Removes the clip window. All graphics commands can now write to the entire screen.
- **Arguments:** None.
- **Returns:** None.
- **See also:** setClipRange
- **Example:**

undoRotate

- **C++ Prototype:**

```
void Ucglib::undoRotate(void)
```

- **Description:** Restore the original display orientation.
- **Arguments:** -
- **Returns:** -
- **See also:** setRotate90
- **Example:**

undoScale

- **C++ Prototype:**

```
void Ucglib::undoScale(void)
```

- **Description:** Return to none scaling mode.
- **Arguments:** -

- **Returns:** -
- **See also:** setScale2x2
- **Example:**