

MUDO DDT  
Programming

---

# u8g2

## Graphical library v2.28.10

---

Savushkin Aleksander / Saransk 2021

# C++/Arduino

## Example

## Reference

<u>begin</u>	(инициализация) [3]	<u>getMenuEvent</u>	(возвр событие наж на кл) [15]
<u>clear</u>	(очищает дисплей) [4]	<u>getStrWidth</u>	(возвр шир в рх строки) [15]
<u>clearBuffer</u>	(очищает буфер дисплея) [4]	<u>getUTF8Width</u>	(возвр шир в рх строки UTF8) [15]
<u>clearDisplay</u>	(очищает буфер и дисплей) [4]	<u>home</u>	(курсор в верх-лев полож) [15]
<u>disableUTF8Print</u>	(откл.под. UTF8) [5]	<u>initDisplay</u>	(сброс и настр. дисплея) [15]
<u>drawBitmap</u>	(рис.растр. изображение) [5]	<u>nextPage</u>	(отобр.содержимого экрана) [16]
<u>drawBox</u>	(рис. заполн. прямоуго) [5]	<u>print</u>	(выводит текст и значения) [16]
<u>drawCircle</u>	(рис. круг) [6]	<u>sendBuffer</u>	(отпр.содерж.в буфер экрана) [17]
<u>drawDisc</u>	(рис. диск) [6]	<u>sendE</u>	(актив.коман. дисплея а,b,c) [17]
<u>drawEllipse</u>	(рис. не заполн. эллипс) [7]	<u>setAutoPageClear</u>	(очистка буфера экрана) [18]
<u>drawFilledEllipse</u>	(рис. заполн. эллипс) [7]	<u>setBitmapMode</u>	(прозрач содерж.экрана) [18]
<u>drawFrame</u>	(рис. рамку) [8]	<u>setBusClock</u>	(такт частота I2C) [19]
<u>drawGlyph</u>	(рис. глиф) [8]	<u>setClipWindow</u>	(огр вывод графич информ) [19]
<u>drawHLine</u>	(рис. горизонт линию) [9]	<u>setContrast</u>	(установка контраст) [19]
<u>drawLine</u>	(рис. линию по x/y) [9]	<u>setCursor</u>	(опред.кур ф-и print) [20]
<u>drawPixel</u>	(рис. пиксель) [9]	<u>setDisplayRotation</u>	(опред.ротации экрана) [20]
<u>drawRBox</u>	(рис прямоуго с закруг угл) [10]	<u>setDrawColor</u>	(знач XOR содерж.экрана) [21]
<u>drawRFrame</u>	(рис рамку с закруг угл) [10]	<u>setFlipMode</u>	(поворот содерж на 180) [22]
<u>drawStr</u>	(рис строку) [10]	<u>setFont</u>	(устан шрифта строки) [22]
<u>drawTriangle</u>	(рис треугольник) [11]	<u>setFontDirection</u>	(направ шрифта строки) [23]
<u>drawUTF8</u>	(рис строку в кодир UTF8) [11]	<u>setFontMode</u>	(прозрач шрифта строки) [23]
<u>drawVLine</u>	(рис вертикал линию) [12]	<u>setFontPosBaseline</u>	(позиция текста) [24]
<u>drawXBM</u>	(рис растровое изобр) [12]	<u>setFontPosBottom</u>	(позиция текста) [24]
<u>enableUTF8Print</u>	(UTF8 для ф-и print) [13]	<u>setFontPosTop</u>	(позиция текста) [24]
<u>firstPage</u>	(отобр.код.экрана) [13]	<u>setFontPosCenter</u>	(позиция текста) [24]
<u>getAscent</u>	(возвр выс глифов) [14]	<u>setFontRefHeightAll</u>	(подъём текста) [24]
<u>getDescent</u>	(возвр выс глифов) [14]	<u>setFontRefHeightExtendedText</u>	(подъём текста) [24]
<u>getDisplayHeight</u>	(возвр выс дисплея) [14]	<u>setFontRefHeightText</u>	(подъём текста) [24]
<u>getDisplayWidth</u>	(возвр шир дисплея) [14]	<u>setI2CAddress</u>	(измен.I2C адреса) [25]
<u>getMaxCharHeight</u>	(возвр выс растр. изобр) [14]	<u>setMaxClipWindow</u>	(удал эфф setClipWindow) [25]
<u>getMaxCharWidth</u>	(возвр шир растр. изобр) [14]	<u>setPowerSave</u>	(энергосбер экрана) [25]
		<u>updateDisplay</u>	(обнов информ экрана) [26]
		<u>updateDisplayArea</u>	(обнов информ экрана) [26]
		<u>userInterfaceInputValue</u>	(запрашив ввод) [27]
		<u>userInterfaceMessage</u>	(выв сообщ на экр) [27]
		<u>userInterfaceSelectionList</u>	(спис операц) [28]

# C++/Arduino Example

```
#include <Arduino.h>
#include <SPI.h>
#include <U8g2lib.h>

U8G2_SSD1306_128X64_NONAME_1_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10,
/* dc=*/ 9, /* reset=*/ 8);

void setup(void)
{
  u8g2.begin();
}

void loop(void)
{
  u8g2.firstPage();

  do {
    u8g2.setFont(u8g2_font_ncenB14_tr);
    u8g2.drawStr(0, 15, "Hello World!");
  }

  while ( u8g2.nextPage() );
  delay(1000);
}
```

# Begin

`begin(void)`

`begin(uint8_t menu_select_pin, uint8_t menu_next_pin, uint8_t menu_prev_pin, uint8_t menu_up_pin = U8X8_PIN_NONE, uint8_t menu_down_pin = U8X8_PIN_NONE, uint8_t menu_home_pin = U8X8_PIN_NONE)`

**Description:** Упрощенная процедура настройки дисплея для среды Arduino. См. Руководство по установке для выбора подходящего конструктора U8g2. Эта функция сбрасывает, настраивает, очищает и отключает режим энергосбережения дисплея. U8g2 также может обнаруживать события нажатия клавиш. Можно наблюдать до шести кнопок. Здесь можно присвоить код Arduino. `begin()` вызывает: [initDisplay](#) [setPowerSave](#) [clearDisplay](#)

**Arguments:** -

**Returns:** всегда 1/true

**See also:** [initDisplay](#) [setPowerSave](#) [clearDisplay](#)

**Example:**

```
void setup(void) {
  u8g2.begin();
}

void loop(void) {
  u8g2.firstPage();
  do
  {
    u8g2.setFont(u8g2_font_ncenB14_tr);
    u8g2.drawStr(0,15,"Hello World!");
  }

  while ( u8g2.nextPage() );
  delay(1000);
}
```



# Clear

`clear(void)`

**Description:** Очищает все пиксели на дисплее и в буфере. Помещает курсор функции `print()` в верхний левый угол.

`clear()` вызывает: `home`, `clearDisplay`, `clearBuffer`

**Arguments:**

**Returns:** -

**See also:** [print](#) [home](#) [clearBuffer](#)

# clearBuffer

`clearBuffer(void)`

**Description:** Очищает все пиксели в буфере кадра памяти. Используйте `sendBuffer` для передачи очищенного буфера кадра на дисплей. В большинстве случаев эта процедура полезна только с полным буфером кадра в ОЗУ микроконтроллера (конструктор с опцией буфера «f», см. [here](#)). Эта процедура также отправит сообщение об обновлении (`refreshDisplay`) на e-Paper/e-Ink устройства.

**Arguments:**

**Returns:** -

**See also:** [sendBuffer](#)

**Example:**

```
void loop() {
  u8g2.clearBuffer();

  // ... напишите что-нибудь в буфер

  u8g2.sendBuffer();
  delay(1000);
}
```

# clearDisplay

`clearDisplay(void)`

**Description:** Очищает все пиксели во внутреннем буфере и на подключенном дисплее. Эта процедура также вызывается с `begin()`. Обычно эту функцию вызывать не нужно, за исключением процедуры инициализации. Другие процедуры, такие как `sendBuffer` и `nextPage`, также перезаписывают (и очищают) отображение.

**Arguments:**

**Returns:** -

**Notes:** - Эта команда может быть использована со всеми конструкторами (`_F_`, `_1_`, `_2_`).

- Не используйте эту команду в цикле изображения (между `firstPage` и `nextPage`).

**See also:** [begin](#)

# disableUTF8Print

disableUTF8Print(void)

**Description:** Отключает поддержку UTF8 для функции print() Arduino. Это также настройка по умолчанию.

**Arguments:** -

**Returns:** -

**See also:** [print](#), [enableUTF8Print](#)

# drawBitmap

drawBitmap(u8g2\_uint\_t x, u8g2\_uint\_t y, u8g2\_uint\_t cnt, u8g2\_uint\_t h, const uint8\_t \*bitmap)

**Description:** Нарисуйте растровое изображение в указанной позиции **x / y** (верхний левый угол растрового изображения). Части растрового изображения могут выходить за границы отображения. Растровое изображение определяется растровым изображением массива - bitmap. Сброшенный бит означает: не рисовать пиксель. Установленный бит внутри массива означает: записать пиксель с текущим индексом цвета. Для монохромного дисплея индекс цвета 0 очистит пиксель (в сплошном режиме), а индекс цвета 1 установит пиксель.

**Arguments:**

- **x:** X-позиция (левая позиция растрового изображения).
- **y:** Позиция Y (верхняя позиция растрового изображения).
- **cnt:** Количество байтов растрового изображения по горизонтали. Ширина растрового изображения составляет **cnt\*8**.

**Returns:** -

**Note:** Эта функция больше не должна использоваться, используйте вместо нее drawXBM.

**See also:** [drawXBM](#) [setBitmapMode](#)

# drawBox

drawBox(u8g2\_uint\_t x, u8g2\_uint\_t y, u8g2\_uint\_t w, u8g2\_uint\_t h)

**Description:** Нарисуйте рамку (закрашенную рамку), начиная с позиции **x / y** (верхний левый край). Коробка имеет ширину **w** и высоту **h**. Части бокса могут выходить за границы отображения. Эта процедура будет использовать текущий цвет (setDrawColor) для рисования поля. Для монохромного дисплея цветовой индекс 0 очищает пиксель, а цветовой индекс 1 устанавливает пиксель.

**Arguments:**

- **x:** X-положение верхнего левого края.
- **y:** Y-положение верхнего левого края.
- **w:** Ширина коробки.
- **h:** Высота коробки.

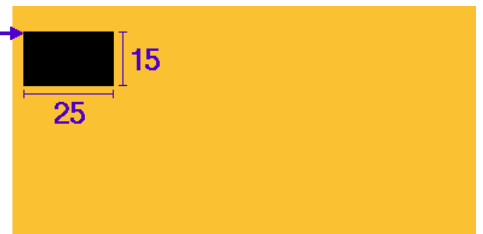
**Returns:**

**See also:** [drawFrame](#) [setDrawColor](#)

**Example:**

u8g2.drawBox(3,7,25,15);

x=3, y=7 →



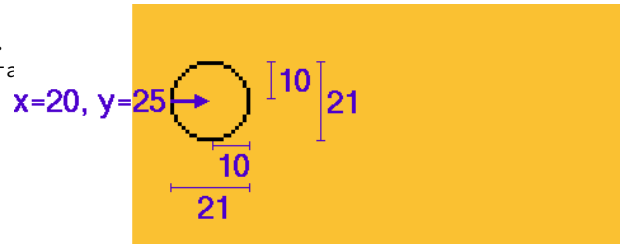
## drawCircle

```
drawCircle(u8g2_uint_t x0, u8g2_uint_t y0, u8g2_uint_t rad, uint8_t opt =
U8G_DRAW_ALL)
```

**Description:** Нарисуйте круг с радиусом `rad` в позиции `(x0, y0)`. Диаметр круга  $2*rad+1$ . В зависимости от `opt`, можно нарисовать только некоторые участки круга. Возможные значения `opt`: `U8G2_DRAW_UPPER_RIGHT`, `U8G2_DRAW_UPPER_LEFT`, `U8G2_DRAW_LOWER_LEFT`, `U8G2_DRAW_LOWER_RIGHT`, `U8G2_DRAW_ALL`. Эти значения можно комбинировать с «|» оператор. Эта процедура будет использовать текущий цвет (`setDrawColor`) для рисования.

**Arguments:**

- `x0, y0`: Положение центра круга.
- `rad`: Определяет размер круга: `Radius = rad`.
- `opt`: Выбирает некоторые или все части круга
- `U8G2_DRAW_UPPER_RIGHT`
- `U8G2_DRAW_UPPER_LEFT`
- `U8G2_DRAW_LOWER_LEFT`
- `U8G2_DRAW_LOWER_RIGHT`
- `U8G2_DRAW_ALL`



**Returns:**

**Note:** `Draw_color 2` (XOR Mode) не поддерживается.

**See also:** [drawDisc](#) [setDrawColor](#)

**Example:**

```
u8g2.drawCircle(20, 25, 10, U8G2_DRAW_ALL);
```

## drawDisc

```
drawDisc(u8g2_uint_t x0, u8g2_uint_t y0, u8g2_uint_t rad, uint8_t opt =
U8G_DRAW_ALL)
```

**Description:** Нарисуйте закрашенный круг с радиусом `rad` в позиции `(x0, y0)`. Диаметр круга  $2*rad+1$ . В зависимости от опции можно рисовать только некоторые разделы диска. В зависимости от `opt` можно рисовать только некоторые разделы диска. Возможные значения `opt`: `U8G2_DRAW_UPPER_RIGHT`, `U8G2_DRAW_UPPER_LEFT`, `U8G2_DRAW_LOWER_LEFT`, `U8G2_DRAW_LOWER_RIGHT`, `U8G2_DRAW_ALL`. Эти значения можно комбинировать с «|» оператор. Эта процедура будет использовать текущий цвет (`setDrawColor`) для рисования.

**Arguments:**

- `x0, y0`: Положение центра диска.
- `rad`: Определяет размер круга: `Radius = rad`.
- `opt`: Выбирает некоторые или все части круга.
- `U8G2_DRAW_UPPER_RIGHT`
- `U8G2_DRAW_UPPER_LEFT`
- `U8G2_DRAW_LOWER_LEFT`
- `U8G2_DRAW_LOWER_RIGHT`
- `U8G2_DRAW_ALL`

**Returns:**

**Note:** `Draw_color 2` (XOR Mode) не поддерживается.

**See also:** [drawCircle](#) [setDrawColor](#)

**Example:** Смотрите [drawCircle](#)

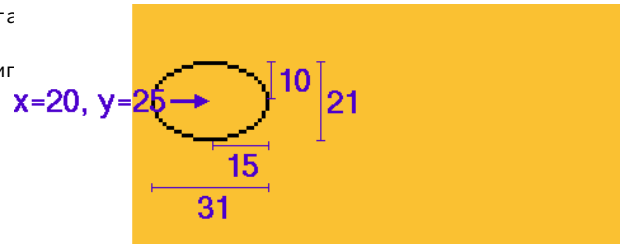
## drawEllipse

`drawEllipse(u8g2_uint_t x0, u8g2_uint_t y0, u8g2_uint_t rx, u8g2_uint_t ry, uint8_t opt)`

**Description:** Нарисуйте эллипс с радиусом **rx** и **'ry'** в позиции (**x0**, **y0**). **rx\*ry** должен быть меньше 512 в 8-битном режиме u8g2. В зависимости от **opt** можно рисовать только некоторые разделы диска. Возможные значения **opt**: U8G\_DRAW\_UPPER\_RIGHT, U8G\_DRAW\_UPPER\_LEFT, U8G\_DRAW\_LOWER\_LEFT, U8G\_DRAW\_LOWER\_RIGHT, U8G\_DRAW\_ALL. Эти значения можно комбинировать с «|» оператор. Диаметр в два раза больше радиуса плюс один.

**Arguments:**

- x0, y0: Положение центра заполненного круга
- rx, ry: Определяет размер эллипса.
- opt: Выбирает некоторые или все части эллиг
- U8G2\_DRAW\_UPPER\_RIGHT
- U8G2\_DRAW\_UPPER\_LEFT
- U8G2\_DRAW\_LOWER\_LEFT
- U8G2\_DRAW\_LOWER\_RIGHT
- U8G2\_DRAW\_ALL



**Returns:**

**Note:** `Draw_color 2` (XOR Mode) не поддерживается.

**See also:** [drawCircle](#)

**Example:**

```
u8g2.drawEllipse(20, 25, 15, 10, U8G2_DRAW_ALL);
```

## drawFilledEllipse

`drawFilledEllipse(u8g2_uint_t x0, u8g2_uint_t y0, u8g2_uint_t rx, u8g2_uint_t ry, uint8_t opt)`

**Description:** Нарисуйте закрашенный эллипс с радиусом **rx** и **'ry'** в позиции (**x0**, **y0**). **rx\*ry** должен быть меньше 512 в 8-битном режиме u8g2. В зависимости от опции можно рисовать только некоторые разделы диска. Возможные значения **opt**: U8G\_DRAW\_UPPER\_RIGHT, U8G\_DRAW\_UPPER\_LEFT, U8G\_DRAW\_LOWER\_LEFT, U8G\_DRAW\_LOWER\_RIGHT, U8G\_DRAW\_ALL. Эти значения можно комбинировать с «|» оператор.

**Arguments:**

- x0, y0: Положение центра заполненного круга.
- rx, ry: Определяет размер эллипса.
- opt: Выбирает некоторые или все части эллипса.
- U8G2\_DRAW\_UPPER\_RIGHT
- U8G2\_DRAW\_UPPER\_LEFT
- U8G2\_DRAW\_LOWER\_LEFT
- U8G2\_DRAW\_LOWER\_RIGHT
- U8G2\_DRAW\_ALL

**Returns:**

**Note:** `Draw_color 2` (XOR Mode) не поддерживается.

**See also:** [drawCircle](#)

**Example:** [drawEllipse](#)



## drawFrame

`drawFrame(u8g2_uint_t x, u8g2_uint_t y, u8g2_uint_t w, u8g2_uint_t h)`

**Description:** Нарисуйте рамку (пустое поле), начиная с позиции **x / y** (верхний левый край). Коробка имеет ширину **w** и высоту **h**. Части кадра могут выходить за границы дисплея. Эта процедура будет использовать текущий цвет (`setDrawColor`) для рисования поля. Для монохромного дисплея цветовой индекс **0** - очищает пиксель, а цветовой индекс **1** - устанавливает пиксель.

**Arguments:**

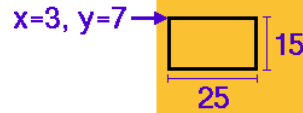
- **x:** X-положение верхнего левого края.
- **y:** Y-положение верхнего левого края.
- **w:** Ширина рамы.
- **h:** Высота рамы.

**Returns:**

**See also:** [drawBox](#) [setDrawColor](#)

**Example:**

```
u8g2.drawFrame(3,7,25,15);
```



## drawGlyph

`drawGlyph(u8g2_uint_t x, u8g2_uint_t y, uint16_t encoding)`

**Description:** Нарисуйте одного персонажа. Символ помещается в указанную позицию пикселя **x** и **y**. U8g2 поддерживает младшие 16 бит диапазона символов Юникода (plane 0/Basic Multilingual Plane): кодировка может быть любым значением от 0 до 65535. Глиф может быть нарисован только в том случае, если кодировка существует в активном шрифте.

**Arguments:**

- **x, y:** Положение персонажа на дисплее.
- **encoding:** Значение Юникода персонажа.

**Returns:** -

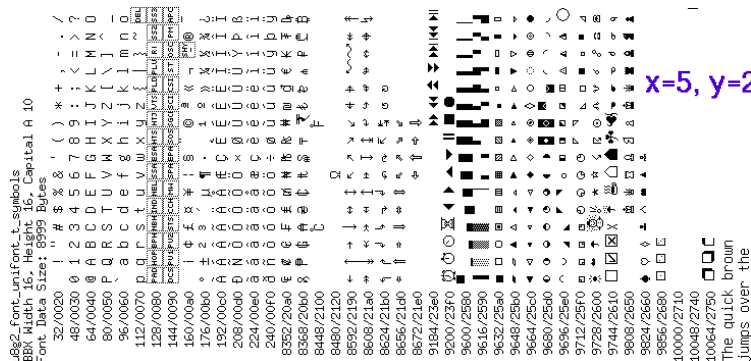
**Note:** Эта функция рисования зависит от текущего [font mode](#) и [drawing color](#).

**See also:** [setFont](#)

**Example:** Глиф «снеговик» является частью символов погоды в формате Unicode и имеет кодировку Unicode 9731 (в десятичной системе - decimal) / 2603 (в шестнадцатеричной системе - hex): «☹». "Снеговик" также является частью шрифта `u8g2 u8g2_font_unifont_t_symbols` (см. Ниже).

```
u8g2.setFont(u8g2_font_unifont_t_symbols);
```

```
u8g2.drawGlyph(5, 20, 0x2603); /* dec 9731/hex 2603 Snowman */
```



## drawHLine

`drawHLine(u8g2_uint_t x, u8g2_uint_t y, u8g2_uint_t w)`

**Description:** Нарисуйте горизонтальную линию, начиная с позиции **x / y** (левый край). Ширина (длина) линии составляет **w** пикселей. Части линии могут выходить за границы отображения. Эта процедура использует текущий цветовой индекс для рисования линии. Индекс цвета **0** – **очищает пиксель**, а индекс цвета **1** – **устанавливает пиксель**.

**Arguments:**

- x: X-позиция.
- y: Y-позиция.
- w: Длина горизонтальной линии.

**Returns:** -

**See also:** [setDrawColor](#), [drawVLine](#)

## drawLine

`drawLine(u8g2_uint_t x0, u8g2_uint_t y0, u8g2_uint_t x1, u8g2_uint_t y1)`

**Description:** Проведите линию между двумя точками. Эта процедура будет использовать текущий цвет ([setDrawColor](#)).

**Arguments:**

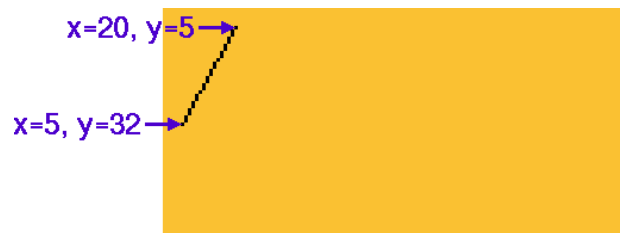
- x0: X-положение первой точки.
- y0: Y-положение первой точки.
- x1: X-положение второй точки.
- y1: Y-положение второй точки.

**Returns:**

**See also:** [drawPixel](#) [setDrawColor](#)

**Example:**

`u8g2.drawLine(20, 5, 5, 32);`



## drawPixel

`drawPixel(u8g2_uint_t x, u8g2_uint_t y)`

**Description:** Нарисуйте пиксель в указанной позиции **x / y**. Позиция **(0,0)** находится в верхнем левом углу дисплея. Положение может выходить за границы отображения. Эта процедура использует текущий индекс цвета для рисования пикселя. Индекс цвета **0** – **очищает пиксель**, а индекс цвета **1** – **устанавливает пиксель**.

**Arguments:**

- x: X-позиция.
- y: Y-позиция.

**Returns:**

**See also:** [setDrawColor](#)

# drawRBox

## drawRFrame

```
drawRBox(u8g2_uint_t x, u8g2_uint_t y, u8g2_uint_t w, u8g2_uint_t h,
u8g2_uint_t r)
drawRFrame(u8g2_uint_t x, u8g2_uint_t y, u8g2_uint_t w, u8g2_uint_t h,
u8g2_uint_t r)
```

**Description:** Нарисуйте прямоугольник / рамку с закругленными краями, начиная с позиции **x / y** (верхний левый край). Коробка / рама имеет ширину **w** и высоту **h**. Части бокса могут выходить за границы отображения. Края имеют радиус **r**. Требуется, чтобы  $w \geq 2 * (r + 1)$  и  $h \geq 2 * (r + 1)$ . Это условие не проверяется. Поведение не определено, если **w** или **h** меньше  $2 * (r + 1)$ . Эта процедура использует текущий цветовой индекс для рисования рамки. Для монохромного дисплея цветовой индекс **0** – очищает пиксель, а цветовой индекс **1** – устанавливает пиксель.

**Arguments:**

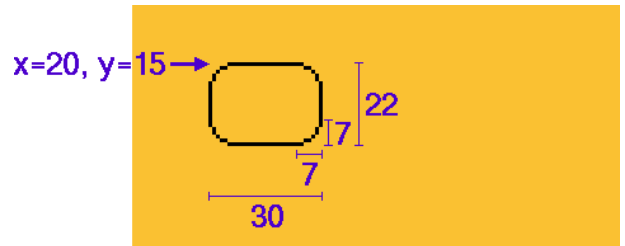
- **x**: X-положение верхнего левого края.
- **y**: Y-положение верхнего левого края.
- **w**: Ширина коробки.
- **h**: Высота коробки.
- **r**: Радиус для четырех краев.

**Returns:** -

**See also:** [setDrawColor](#), [drawFrame](#), [drawBox](#)

**Example:**

```
u8g2.drawRFrame(20,15,30,22,7);
```



# drawStr

```
drawStr(u8g2_uint_t x, u8g2_uint_t y, const char *s)
```

**Description:** Нарисуйте строку. Первый символ помещается в позиции **x** и **y**. Используйте [setFont](#), чтобы назначить шрифт перед рисованием строки на дисплее. Чтобы нарисовать символ с кодировкой от 127 до 255, используйте escape-последовательность C / C++ / Arduino «\xab» (шестнадцатеричное значение ab) или «\xyz» (восьмеричное значение xyz). Эта функция не может рисовать глифы с кодировкой больше или равной 256. Используйте [drawUTF8](#) или [drawGlyph](#) для доступа к глифам с кодировкой больше или равной 256.

**Arguments:**

- **x, y**: Положение первого персонажа на дисплее.
- **s**: Текст.

**Returns:** Ширина строки.

**Note 1:** Эта функция рисования зависит от текущего режима [font\\_mode](#) и [drawing\\_color](#).

**Note 2:** Используйте функцию [print](#), чтобы распечатать значение числовой переменной.

**See also:** [setFont](#) [drawUTF8](#) [drawGlyph](#) [print](#)

**Example:**

```
u8g2.setFont(u8g2_font_ncenB14_tr);
u8g2.drawStr(0,15,"Hello World!");
```



## drawTriangle

```
drawTriangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2,
int16_t y2)
```

**Description:** Нарисуйте треугольник (закрашенный многоугольник). Аргументы 16-битные, а многоугольник обрезается по размеру дисплея. Несколько многоугольников нарисованы так, чтобы они точно совпадали без перекрытия: левая сторона многоугольника рисуется, правая не рисуется. Верхняя сторона рисуется только в том случае, если она плоская.

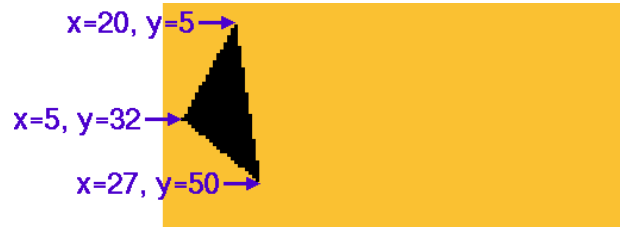
**Arguments:**

- x0: X-точка позиции 0.
- y0: Y-точка позиции 0.
- x1: X-точка позиции 1.
- y1: Y-точка позиции 1.
- x2: X-точка позиции 2.
- y2: Y-точка позиции 2.

**Returns:** -

**Example:**

```
u8g2.drawTriangle(20,5, 27,50, 5,32);
```



## drawUTF8

```
drawUTF8(u8g2_uint_t x, u8g2_uint_t y, const char *s)
```

**Description:** Нарисуйте строку в кодировке UTF-8. Есть два предварительных условия для использования этой функции: (A) компилятор C / C ++ / Arduino должен поддерживать (это значение по умолчанию для компилятора GNU, который также используется для большинства плат Arduino) и (B) редактор кода / IDE должен поддерживать и хранить код C / C ++ / Arduino как UTF-8 (верно для IDE Arduino). Если эти условия соблюдены, вы можете использовать символ с кодовым значением больше 127 непосредственно в строке (конечно, символ должен существовать в файле шрифта, смотрите также [setFont](#)). Преимущество: не требуются эскап-коды, а исходный код более читабелен. Глиф можно скопировать и вставить в редактор с помощью инструмента "набор символов". Недостаток: код менее переносим, и функция strlen не возвращает количество видимых символов. Используйте [getUTF8Len](#) вместо strlen.

**Arguments:**

- x, y: Положение первого персонажа на дисплее.
- s: UTF-8 закодированный текст.

**Returns:** Ширина строки.

**Note 1:** Эта функция чертежа зависит от текущего [font\\_mode](#) и [drawing\\_color](#).

**Note 2:** Используйте функцию [print](#), чтобы распечатать значение числовой переменной.

**See also:** [setFont](#) [drawStr](#) [print](#)

**Example:**

```
u8g2.setFont(u8g2_font_unifont_t_symbols);
u8g2.drawUTF8(5, 20, "Snowman: ☺");
```



## drawVLine

`drawVLine(u8g2_uint_t x, u8g2_uint_t y, u8g2_uint_t h)`

**Description:** Нарисуйте вертикальную линию, начиная с позиции **x / y** (верхний конец). Высота (длина) линии составляет **h** пикселей. Части линии могут выходить за границы отображения. Эта процедура использует текущий цветовой индекс для рисования линии. Индекс цвета **0** – **очищает пиксель**, а индекс цвета **1** – **устанавливает пиксель**.

**Arguments:**

- x: X-позиция.
- y: Y-позиция.
- h: Длина вертикальной линии.

**Returns:** -

**See also:** [setDrawColor](#), [drawHLine](#)

## drawXBM

`drawXBM(u8g2_uint_t x, u8g2_uint_t y, u8g2_uint_t w, u8g2_uint_t h, const uint8_t *bitmap)`  
`drawXBMP(u8g2_uint_t x, u8g2_uint_t y, u8g2_uint_t w, u8g2_uint_t h, const uint8_t *bitmap)`

**Description:** Нарисуйте растровое изображение [XBM Bitmap](#). Позиция (**x, y**) – это, верхний левый угол растрового изображения. XBM содержит монохромные однобитовые растровые изображения. Текущий индекс цвета используется для рисования (см. [setColorIndex](#)) значений пикселей 1. Версия 2.15.x U8g2 вводит сплошной и прозрачный режим для растровых изображений. По умолчанию drawXBM будет рисовать сплошные растровые изображения. Это отличается от предыдущих версий: используйте `setBitmapMode (1)`, чтобы переключиться на предыдущее поведение. Версия этой процедуры для XBMP ожидает, что растровое изображение будет в области PROGMEM (только AVR). Многие инструменты (включая GIMP) могут сохранять растровое изображение как XBM. Хорошая пошаговая инструкция здесь [here \(external link\)](#). Результат будет таким:

**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 };
```

Это можно скопировать прямо в ваш код. Используйте drawXBM, чтобы нарисовать это растровое изображение в точке (0,0):

```
u8g2.drawXBM( 0, 0, u8g_logo_width, u8g_logo_height, u8g_logo_bits);
```

**Arguments:**

- x: X-позиция.
- y: Y-позиция.
- w: Ширина бит-карты.
- h: Высота бит-карты.
- bitmap: Указатель к началу бит-карты.

**Returns:**

**See also:** [setBitmapMode](#)

**Note:** Версия XBMP требует, чтобы массив bitmap определялся таким образом:

```
static const unsigned char u8g_logo_bits[] PROGMEM = { ...
```

# enableUTF8Print

enableUTF8Print(void)

**Description:** Активирует поддержку UTF8 для функции print() Arduino. При активации символы Unicode разрешены для строк, передаваемых в функцию печати. Обычно эта функция вызывается после begin():

```
void setup() {
    u8g2.begin();
    u8g2.enableUTF8Print(); // включить поддержку UTF8 для print()
}
```

**Arguments:** -

**Returns:** -

**See also:** [print](#), [disableUTF8Print](#)

**Example:**

```
void setup() {
    u8g2.begin();
    u8g2.enableUTF8Print(); // включить поддержку UTF8 для функции print()
}
void loop() {
    u8g2.setFont(u8g2_font_unifont_t_chinese2); // использовать chinese2 для всех глифов "你好世界"
    u8g2.firstPage();
    do {
        u8g2.setCursor(0, 40);
        u8g2.print("你好世界"); // Китайский "Здравствуй, мир"
    }
    while ( u8g2.nextPage() );
    delay(1000);
}
```

# firstPage

firstPage(void)

**Description:** Эта команда является частью цикла (изображения), который отображает содержимое дисплея. Эта команда должна использоваться вместе с [nextPage](#). Есть некоторые ограничения: не изменяйте содержимое при выполнении этого цикла. Всегда все перерисовывайте. Невозможно перерисовать только часть контента. Преимущество заключается в меньшем потреблении ОЗУ по сравнению с буфером полного кадра в ОЗУ, см. [sendBuffer](#).

**Arguments:**

**Returns:** -

**Note:** Эта процедура устанавливает текущую позицию страницы на ноль.

**See also:** [nextPage](#)

**Example:**

```
u8g2.firstPage();
do {

    /* все графические команды должны отображаться в корпусе цикла. */

    u8g2.setFont(u8g2_font_ncenB14_tr);
    u8g2.drawStr(0,20,"Hello World!");
}
while ( u8g2.nextPage() );
```

# getAscent

getAscent(void)

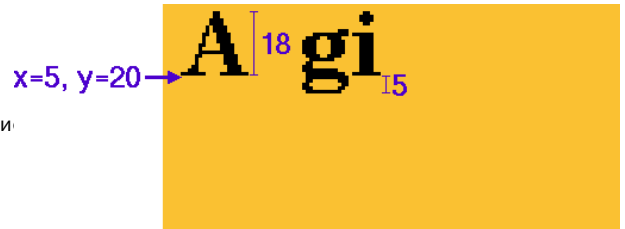
**Description:** Возвращает эталонную высоту глифов выше базового уровня (восхождение). Это значение зависит от текущей высоты ссылки (см. [setFontRefHeightAll](#)).

**Arguments:**

**Returns:** Подъем текущего шрифта.

**See also:** [setFont](#) [getDescent](#) [setFontRefHeightAll](#)

**Example:** На рисунке ниже, восхождение 18 и значени спуска -5 (минус 5!).



# getDescent

getDescent(void)

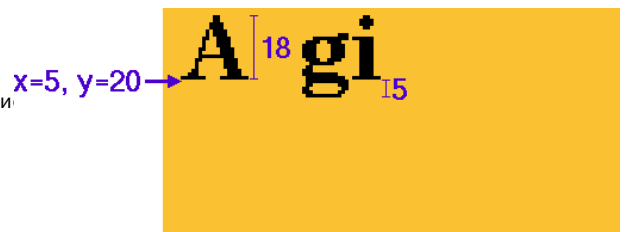
**Description:** Возвращает эталонную высоту глифов ниже базового уровня (спуска). Для большинства шрифтов это значение будет отрицательным. Это значение зависит от текущей высоты ссылки (см. [setFontRefHeightAll](#)).

**Arguments:**

**Returns:** Спуск текущего шрифта.

**See also:** [setFont](#) [getDescent](#) [setFontRefHeightAll](#)

**Example:** На рисунке ниже, восхождение 18 и значени спуска -5 (минус 5!).



## getDisplayHeight

getDisplayHeight(void)

**Description:** Возвращает высоту дисплея.

**Arguments:**

**Returns:** Высота дисплея.

**See also:** [getDisplayWidth](#)

**Example:** -

## getDisplayWidth

getDisplayWidth(void)

**Description:** Возвращает ширину дисплея.

**Arguments:**

**Returns:** Ширина дисплея.

**See also:** [getDisplayHeight](#)

**Example:** -

## getMaxCharHeight

getMaxCharHeight(void)

**Description:** Каждый глиф хранится как растровое изображение. Это возвращает высоту самого большого растрового изображения в шрифте.

**Arguments:**

**Returns:** Самая большая высота любого глифа в шрифте.

**See also:** [getMaxCharWidth](#)

**Example:** -

## getMaxCharWidth

getMaxCharWidth(void)

**Description:** Каждый глиф хранится как растровое изображение. Это возвращает ширину самого большого растрового изображения в шрифте.

**Arguments:**

**Returns:** Самая большая ширина любого глифа в шрифте.

**See also:** [getMaxCharHeight](#)

**Example:** -

## getMenuEvent

getMenuEvent(void)

**Description:** Возвращает событие нажатия клавиши. Номера контактов до шести контактов должны быть установлены с помощью функции `begin`.

getMenuEvent return values
U8X8_MSG_GPIO_MENU_SELECT
U8X8_MSG_GPIO_MENU_NEXT
U8X8_MSG_GPIO_MENU_PREV
U8X8_MSG_GPIO_MENU_HOME
U8X8_MSG_GPIO_MENU_UP
U8X8_MSG_GPIO_MENU_DOWN

**Arguments:** -

**Returns:** 0, если кнопка не была нажата или ключевое событие нажатия.

**See also:** [begin](#)

## getStrWidth

getStrWidth(const char \*s)

**Description:** Вернуть ширину пикселя строки.

**Arguments:**

- s: текст.

**Returns:** Ширина строки, если нарисована текущим шрифтом ([setFont](#)).

**See also:** [setFont](#) [drawStr](#)

## getUTF8Width

getUTF8Width(const char \*s)

**Description:** Возвращает ширину в пикселях строки в кодировке UTF-8.

**Arguments:**

- s: Закодированный текст UTF-8.

**Returns:** Ширина строки, если нарисована текущим шрифтом ([setFont](#)).

**See also:** [setFont](#) [drawStr](#)

## home

home(void)

**Description:** Помещает курсор функции `print()` в верхний левый угол. Части текста могут быть невидимы после этой команды, если ссылка на глиф находится не в верхней части символов.

**Arguments:**

**Returns:** -

**See also:** [print](#) [clear](#)

# initDisplay

initDisplay()

**Description:** Сбросьте и настройте дисплей. Эта процедура должна быть вызвана до того, как какие-либо другие процедуры начнут рисовать что-либо на дисплее. Эта процедура оставляет дисплей в режиме энергосбережения. Чтобы что-то увидеть на экране, сначала отключите режим энергосбережения (`setPowerSave`). Эта процедура вызывается процедурой начала. Первоначально необходимо вызвать либо `begin`, либо `initDisplay`.

**Arguments:**

**Returns:** -

**See also:** [setPowerSave](#) [begin](#)

**Example:**



## nextPage

nextPage()

**Description:** Эта команда является частью цикла (изображения), который отображает содержимое дисплея. Эта команда должна использоваться вместе с `firstPage`. Есть некоторые ограничения: не изменяйте содержимое при выполнении этого цикла. Всегда все перерисовывайте. Невозможно перерисовать только часть контента. Преимущество заключается в меньшем потреблении ОЗУ по сравнению с буфером полного кадра в ОЗУ, см. `SendBuffer`. Эта процедура отправит сообщение обновления (`refreshDisplay`) на устройство e-Paper/e-Ink после завершения цикла (непосредственно перед возвратом 0).

**Arguments:**

**Returns:** 0, как только цикл завершен (все данные передаются на дисплей).

**Note:** Эта процедура добавляет высоту (в строках плитки) текущего буфера к текущему положению страницы.

**See also:** `firstpage`

**Example:**

```
u8g2.firstPage();
do {

/* все графические команды должны отображаться в корпусе цикла. */

u8g2.setFont(u8g2_font_ncenB14_tr);
u8g2.drawStr(0,20,"Hello World!");
}
while ( u8g2.nextPage() );
```

## print

print(...)

**Description:** Это функция Arduino `print()`. См. Описание на веб-странице Arduino [here](#) и [here](#). Эта процедура запишет текст в текущую позицию курсора с текущим шрифтом, установленным `setFont`. Положение курсора может быть установлено с помощью `setCursor`. Поддержка UTF-8 может быть включена с помощью `enableUTF8Print`. Эта функция может печатать значения переменных и поддерживает макрос `F()`.

**Arguments:** см. ссылки.

**Returns:** -

**Note 1:** Эта функция зависит от текущего `font_mode` и `drawing_color`.

**Note 2:** Используйте `print(u8x8_u8toa(value, digits))` или `print(u8x8_u16toa(value, digits))`, чтобы печатать числа постоянной ширины (при необходимости числа имеют префикс 0).

**See also:** `print (U8x8)`, `enableUTF8Print`, `setCursor`, `setFont`

**Example:**

```
u8g2.setFont(u8g2_font_ncenB14_tr);
u8g2.setCursor(0, 15);
u8g2.print("Hello World!");
```



# sendBuffer

sendBuffer(void)

**Description:** Отправить на дисплей содержимое буфера кадров памяти. Используйте `clearBuffer`, чтобы очистить буфер, и функции рисования, чтобы отрисовать что-то в буфере кадра. Эта процедура полезна только с полным буфером кадра в ОЗУ микроконтроллера (конструктор с параметром буфера «f», см. Здесь). Эта процедура также отправит сообщение об обновлении (`refreshDisplay`) на устройство e-Paper/e-Ink.

**Arguments:**

**Returns:** –

**Note:** Фактически эта процедура отправит текущую страницу на дисплей. Это означает, что содержимое внутреннего пиксельного буфера будет помещено в строку плитки, заданную текущей позицией страницы. Это означает, что эту процедуру можно использовать для частичных обновлений на выгружаемых устройствах (конструктор с опцией буфера «1» или «2»). Однако это будет работать только для ЖК-дисплеев. Он не будет работать с большинством устройств электронной бумаги / электронных чернил из-за переключателя буфера в контроллере дисплея. Вывод: используйте эту команду только вместе с конструкторами полных буферов. Затем он будет работать со всеми ЖК-дисплеями и устройствами e-Paper/e-Ink.

**See also:** [clearBuffer](#), [updateDisplay](#)

**Example:**

```
void loop(void) {
  u8g2.clearBuffer();

  // ... написать что-то в буфер

  u8g2.sendBuffer();
  delay(1000);
}
```

# sendF

sendF(const char \*fmt, ...)

**Description:** Отправьте специальные команды контроллеру дисплея. Эти команды указаны в таблице контроллера дисплея. U8g2 просто предоставляет интерфейс (функциональность этих команд не поддерживается). Информация передается как последовательность байтов. Каждый байт имеет особое значение:

- Байт команды (c): Команды для контроллера. Обычно этот байт активирует или деактивирует функцию в контроллере дисплея.
- Аргумент (a): некоторые команды требуют дополнительной информации. Затем командный байт требует определенного числа или аргументов.
- Данные пикселей (d): инструктирует контроллер дисплея интерпретировать байт как данные пикселей, которые должны быть записаны в память дисплея. В некоторых случаях для данных пикселей также требуется специальная команда.

**Arguments:**

- `fmt`: последовательность (строка) `c`, `a` или `d`.
- `...`: последовательность байтов, разделенных запятыми, по одному байту на символ в строке `fmt`. Байт будет интерпретироваться соответственно символу в той же позиции строки `fmt`.

**Returns:** –

**Note:** Функция `C` будет доступна с v2.27

**Example 1:** Отправить одну команду `byte`: Включить инверсию цвета дисплея на многих дисплеях:

```
u8g2.sendF("c", 0x0a7);
```

**Example 2:** Отправить несколько команд с аргументами: Активировать аппаратный прокрутки влево на дисплее SSD1306

```
u8g2.sendF("caaaaaac", 0x027, 0, 3, 0, 7, 0, 255, 0x2f);
```

## setAutoPageClear

setAutoPageClear(uint8\_t mode)

**Description:** Включает (**mode = 1**) или отключает (**mode = 0**) автоматическую очистку буфера пикселей с помощью процедур firstPage и nextPage. По умолчанию он включен, и в большинстве случаев отключать его не требуется. Если этот параметр отключен, пользователь должен установить ВСЕ пиксели текущего буфера пикселей в какое-либо подходящее состояние. Буфер можно стереть вручную с помощью процедуры clearBuffer. Одним из приложений для использования этой функции является ситуация, когда фон визуализируется вручную посредством прямого управления пиксельным буфером (см. Пример DirectAccess.ino).

**Arguments:**

- mode: 0, чтобы отключить автоматическую очистку внутреннего буфера пикселей. Значение по умолчанию - 1.

**Returns:** Ширина буфера в плитках.

**See also:** [getBufferPtr](#)

## setBitmapMode

setBitmapMode(uint8\_t is\_transparent)

**Description:** Определяет, будут ли растровые функции записывать цвет фона (режим 0 / сплошной, is\_transparent = 0) или нет (режим 1 / прозрачный, is\_transparent = 1). Режим по умолчанию - 0 (сплошной режим).

**Arguments:** включить (1) или отключить (0) прозрачный режим.

**Returns:** -

**See also:** [drawBitmap](#) [drawXBM](#)

**Note:** Эта функция будет доступна с v2.15.x

**Example:**

```
u8g2.setDrawColor(1);
u8g2.setBitmapMode(0);
u8g2.drawXBM(4,3, u8g2_logo_97x51_width, u8g2_logo_97x51_height, u8g2_logo_97x51_bits);
u8g2.drawXBM(12,11, u8g2_logo_97x51_width, u8g2_logo_97x51_height, u8g2_logo_97x51_bits);
```



```
u8g2.setDrawColor(1);
u8g2.setBitmapMode(1);
u8g2.drawXBM(4,3, u8g2_logo_97x51_width, u8g2_logo_97x51_height, u8g2_logo_97x51_bits);
u8g2.drawXBM(12,11, u8g2_logo_97x51_width, u8g2_logo_97x51_height, u8g2_logo_97x51_bits);
```



## setBusClock

```
setBusClock(uint32_t clock_speed);
```

**Description:** Только среда Arduino: назначьте тактовую частоту шины (частоту) для I2C и SPI. Если эта функция не вызывается, будут использоваться значения по умолчанию. Эта команда должна быть размещена перед первым вызовом `u8g2.begin()` или `u8g2.initDisplay()`.

**Arguments:**

- `clock_speed`: Тактовая частота шины I2C или SPI (в Гц)

**Returns:** -

**See also:** [begin](#)

**Note 1:** Значения скорости шины по умолчанию позволяют надежно использовать самые медленные дисплеи. С другой стороны, конкретный дисплей может поддерживать более высокую тактовую частоту шины. Например, SSD1327 по умолчанию использует 100 кГц для I2C, но, похоже, во многих случаях поддерживает 400 кГц. Рекомендуется протестировать более высокие значения частоты шины в текущем приложении. Для I2C используйте «`u8g2.setBusClock (200000);`» или «`u8g2.setBusClock (400000);`». Для SPI попробуйте значения между «`u8g2.setBusClock (1000000);`» и «`u8g2.setBusClock (8000000);`».

**Note 2:** U8g2 всегда будет назначать лучшую частоту шины для текущего дисплея. Однако, если на шине I2C имеется несколько клиентов, может случиться так, что выбранная скорость I2C будет слишком высокой для других устройств. В этом случае заставьте U8g2 использовать скорость, приемлемую для всех клиентов: например, попробуйте «`u8g2.setBusClock (100000);`» который должен работать на всех устройствах.

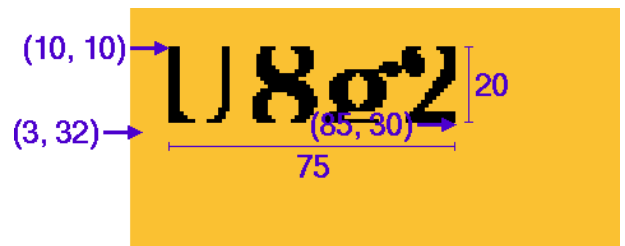
## setClipWindow

```
setClipWindow(x0, y0, x1, y1);
```

**Description:** Ограничивает весь графический вывод указанным диапазоном. Диапазон определяется от **x0** (включен) до **x1** (исключен) и от **y0** (включен) до **y1** (исключен). Используйте `setMaxClipWindow`, чтобы восстановить запись всего окна.

**Arguments:**

- `u8g2`: Указатель на структуру `u8g2`.
- `x0`: Левый край видимой области.
- `y0`: Верхний край видимой области.
- `x1`: Правый край No1 видимой области.
- `y1`: Нижний край No1 видимой области.



**Returns:** -

**See also:** [setMaxClipWindow](#)

**Example:**

```
u8g2.setClipWindow(10, 10, 85, 30);
u8g2.setDrawColor(1);
u8g2.drawStr(3, 32, "U8g2");
```

## setContrast

```
setContrast(uint8_t value);
```

**Description:** Установите контрастность или яркость дисплея (если поддерживается). Диапазон значений: от 0 (без контраста) до 255 (максимальная контрастность или яркость).

**Arguments:**

- `value`: Контрастность или яркость от 0 до 255.

**Returns:** -

**See also:** -

## setCursor

setCursor(x, y)

**Description:** Определите курсор для функции print. Любой вывод функции печати начнется с этой позиции.

**Arguments:**

- x, y: Положение в пикселях курсора функции print

**Returns:** -

**See also:** -

**See also:** [print home](#)

**Example:**

```
u8g2.setFont(u8g2_font_ncenB14_tr);  
u8g2.setCursor(0, 15);  
u8g2.print("Hello World!");
```

x=0, y=15→



## setDisplayRotation

setDisplayRotation(const u8g2\_cb\_t \*u8g2\_cb)

**Description:** Изменяет поворот дисплея. Обычно поворот определяется как часть конструктора U8g2. Аргумент u8g2\_cb может иметь одно из следующих значений:

u8g2_cb	Description
U8G2_R0	Нет вращения, пейзаж
U8G2_R1	90 степень вращения по часовой стрелке
U8G2_R2	180 степень вращения по часовой стрелке
U8G2_R3	270 степень вращения по часовой стрелке
U8G2_MIRROR	Нет вращения, ландшафта, содержимого дисплея отражается (v2.6.x)

**Arguments:**

- u8g2\_cb: Показать аргумент поворота.

**Returns:** -

**See also:** -

# setDrawColor

setDrawColor(uint8\_t color)

**Description:** Определяет битовое значение (индекс цвета) для всех функций рисования. Все функции рисования изменяют память дисплея на это битовое значение. Значение по умолчанию – 1. Например, процедура drawBox установит все пиксели для определенной области в битовое значение, указанное здесь. В v2.11 новое значение цвета 2 активирует режим XOR. Исключения:

- **clear, clearBuffer:** Обе функции всегда устанавливают для буфера значение пикселя 0. Цветовой аргумент setDrawColor игнорируется.
- **drawGlyph:** все процедуры рисования шрифтов будут использовать этот аргумент цвета в качестве цвета переднего плана. В непрозрачном (сплошном) режиме (setFontMode) дополнением значения цвета будет цвет фона и установлено значение 0 для значения цвета 2 (однако рекомендуется не использовать сплошной цвет и режим XOR вместе):

Font Mode	Draw Color	Glyph Foreground Color	Glyph Background Color
0: solid	0	0	1
0: solid	1	1	0
0: solid	2	XOR	0
1: transparent	0	0	–
1: transparent	1	1	–
1: transparent	2	XOR	–

**Arguments:** 0 (очистить значение пикселя в ОЗУ дисплея), 1 (установить значение пикселя) или 2 (режим XOR)

**Returns:** –

**Note:** Не все графические процедуры поддерживают режим XOR. В особенности режим XOR не поддерживается drawCircle, drawDisc, drawEllipse и drawFilledEllipse.

**See also:** [drawBox](#) [drawGlyph](#) [setFontMode](#)

**Example 1:** Строка на фоновом шаблоне со значениями цвета 0 и 1 в прозрачном режиме:



**Example 2:** Режим прозрачного шрифта с разными значениями цвета:

```
u8g2.setFontMode(1); /* активировать режим прозрачного шрифта */
u8g2.setDrawColor(1); /* цвет 1 для коробки */
u8g2.drawBox(22, 2, 35, 50);
u8g2.setFont(u8g2_font_ncenB14_tf);
u8g2.setDrawColor(0);
u8g2.drawStr(5, 18, "abcd");
u8g2.setDrawColor(1);
u8g2.drawStr(5, 33, "abcd");
u8g2.setDrawColor(2);
u8g2.drawStr(5, 48, "abcd");
```



# setFlipMode

setFlipMode(uint8\_t is\_enable)

**Description:** Некоторые дисплеи поддерживают поворот внутреннего буфера кадра на 180 градусов. Этой аппаратной функцией можно управлять с помощью этой процедуры. Важно: перерисуйте весь экран после изменения режима переворота. Лучше всего сначала очистить дисплей, затем изменить режим отображения и, наконец, перерисовать содержимое. Результаты будут неопределенными для любого существующего контента на экране.

**Arguments:** is\_enable: Включение (1) или отключение (0) поворота содержимого дисплея на 180 градусов

**Returns:** -

**See also:** -

# setFont

setFont(const uint8\_t \*font)

**Description:** Определите шрифт u8g2 для функций рисования глифов и строк. Примечание. НЕЛЬЗЯ использовать шрифт u8x8. Доступные шрифты перечислены здесь ([here](#)). Последние два символа имени шрифта определяют тип и набор символов для шрифта:

Font Name	Font Type
u8g2_xxx_tx	Прозрачные гильфы переменной ширины
u8g2_xxx_mx	Глифы моноширинной / фиксированной ширины
u8g2_xxx_hx	Глифы с переменной шириной и общей высотой
u8g2_xxx_8x	Глифы моноширинной / фиксированной ширины в блоке 8x8
Font Name	Character Set
u8g2_xxx_xe	Extended: в шрифт включены символы с юникодами от 32 до 701 (v2.16.x также будет включать большие символы ß).
u8g2_xxx_xf	Full: в шрифт включены глифы с юникодом от 32 до 255.
u8g2_xxx_xr	Restricted: включены только символы от 32 до 127.
u8g2_xxx_xu	Uppercase: Цифры и прописные буквы
u8g2_xxx_xn	Включены числа и некоторые дополнительные символы для печати даты и времени.
u8g2_xxx_x_something	Специальная подборка глифов. Смотрите изображение шрифта для деталей.

**Arguments:**

- font: Указать на шрифт u8g2. Список доступных шрифтов [здесь](#).

**Returns:** -

**See also:** [drawUTF8](#) [drawStr](#) [drawGlyph](#) [List of u8g2 fonts](#)

**Example:**

Шрифт u8g2\_font\_5x7\_tr и u8g2\_font\_pressstart2p\_8u

```
u8g2_font_5x7_tr
BBX Width 5, Height 7, Capital A 6
Font Data Size: 841 Bytes
 32/0020 !"#$%&'()*+,-./
 48/0030 0123456789:;<=>?
 64/0040 @ABCDEFGHIJKLMNO
 80/0050 PQRSTUVWXYZ[\]^_
 96/0060 `abcdefghijklmnopqrstuvwxyz
112/0070 pqrstuvwxyz{|}~
The quick brown fox
jumps over the lazy dog.
```

```
u8g2_font_pressstart2p_8u
BBX Width 8, Height 7, Capital A 8
Font Data Size: 831 Bytes
 32/0020 !"#$%&'()*+,-./
 48/0030 0123456789:;<=>?
 64/0040 @ABCDEFGHIJKLMNO
 80/0050 PQRSTUVWXYZ[\]^_
```

# setFontDirection

setFontDirection(uint8\_t dir)

**Description:** Аргументы определяют направление рисования всех строк или глифов.

Argument	String Rotation	Description
0	0 degree	Слева направо
1	90 degree	Сверху вниз
2	180 degree	Справа налево
3	270 degree	Вверх

**Arguments:**

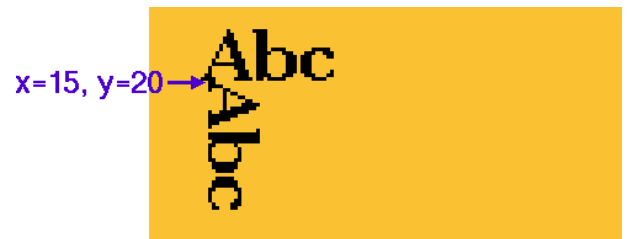
- dir: Направление письма / вращение строки

**Returns:**

**See also:** [drawStr](#)

**Example:**

```
u8g2.setFont(u8g2_font_ncenB14_tf);
u8g2.setFontDirection(0);
u8g2.drawStr(15, 20, "Abc");
u8g2.setFontDirection(1);
u8g2.drawStr(15, 20, "Abc");
```



# setFontMode

setFontMode(uint8\_t is\_transparent)

**Description:** Определяет, будут ли функции рисования глифов и строк записывать цвет фона (режим 0 / сплошной, is\_transparent = 0) или нет (режим 1 / прозрачный, is\_transparent = 1). Режим по умолчанию - 0 (цвет фона символов перезаписывается).

**Arguments:**

- is\_transparent: Включить (1) или отключить (0) прозрачный режим.

**Returns:** -

**Note:** Всегда выбирайте подходящий шрифт, в зависимости от режима шрифта:

Font Name	Font Type	Suitable for...
u8g2_xxx_tx	Transparent glyphs with variable width	is_transparent = 1, XOR Mode
u8g2_xxx_mx	Monospace/fixed width glyphs	is_transparent = 0
u8g2_xxx_hx	Glyphs with variable width and common height	is_transparent = 0
u8g2_xxx_8x	Monospace/fixed width glyphs in a 8x8 box	is_transparent = 0

**See also:** [setDrawColor](#) [setFont](#)

**Example:** В этом примере показаны четыре комбинации с цветовой ценностью 0 и 1 ([setDrawColor](#)).

/\* написать фоновый шаблон, то: \*/

```
u8g2.setFontMode(0);
u8g2.setDrawColor(1);
u8g2.drawStr(3, 15, "Color=1, Mode 0");
u8g2.setDrawColor(0);
u8g2.drawStr(3, 30, "Color=0, Mode 0");
u8g2.setFontMode(1);
u8g2.setDrawColor(1);
u8g2.drawStr(3, 45, "Color=1, Mode 1");
u8g2.setDrawColor(0);
u8g2.drawStr(3, 60, "Color=0, Mode 1");
```





## setFontPosBaseline

## setFontPosBottom

## setFontPosTop

## setFontPosCenter

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

**Description:** Измените ссылочную позицию для функции рисования глифа и строки. По умолчанию ссылочной является «Базовая линия».

**Arguments:**

**Returns:** -

**See also:** [drawUTF8](#) [drawStr](#) [drawGlyph](#)

**Example:**

```
u8g2.setFont(u8g2_font_ncenB18_tf); // изменить позицию ссылки на вертикальный центр шрифта
u8g2.setFontPosCenter();
u8g2.drawStr(5, 20, "Agi");
```



## setFontRefHeightAll

## setFontRefHeightExtendedText

## setFontRefHeightText

```
setFontRefHeightAll(void)
setFontRefExtendedHeightText(void)
setFontRefHeightText(void)
```

**Description:** Вызов одной из этих процедур определит метод расчета подъема и спуска текущего шрифта. Этот метод будет использоваться для текущего и всех остальных шрифтов, которые будут установлены с помощью `setFont()`. Изменение этого метода расчета влияет на `getAscent()` и `getDescent()`. По умолчанию установлен `FontRefHeightText()`.

- `setFontRefHeightAll`: Восхождение будет наивысшим подъемом из всех глифов текущего шрифта. Спуск будет наивысшим спуском из всех глифов текущего шрифта.
- `setFontRefHeightExtendedText`: Подъем будет самым большим подъемом «А», «1» или «(» текущего шрифта. Спуск будет спуском «g» или «(» текущего шрифта.
- `setFontRefHeightText`: Подъем будет подъемом «А» или «1» текущего шрифта. Descent будет спуском «g» текущего шрифта (это значение по умолчанию после запуска).

**Arguments:**

**Returns:**

**See also:** [getAscent](#) [getDescent](#)

## setI2CAddress

setI2CAddress(uint8\_t adr)

**Description:** По умолчанию U8g2 принимает наименьший возможный I2C-адрес дисплея. Эта процедура назначит адрес I2C для u8g2, если дисплей настроен на другой адрес. Вызовите эту процедуру перед begin().

**Arguments:**

adr: Адрес I2C, умноженный на 2 (младший бит должен быть равен нулю)

**Returns:** -

**See also:** [begin](#)

**Note:** Эта процедура доступна с U8g2 v2.6.x

## setMaxClipWindow

setMaxClipWindow(void)

**Description:** Удаляет эффект setClipWindow. Графика записывается на весь дисплей.

**Arguments:**

**Returns:** -

**See also:** [setClipWindow](#)

## setPowerSave

setPowerSave(uint8\_t is\_enable)

**Description:** Активирует (is\_enable = 1) или отключает (is\_enable = 0) режим энергосбережения дисплея. Если активирован режим энергосбережения, на дисплее ничего не будет видно. Содержимое ОЗУ дисплея не изменяется. Эта процедура также вызывается с самого начала.

**Arguments:**

- is\_enable: Включите (1) или отключите (0) режим экономии энергии для дисплея.

**Returns:** -

**See also:** [begin](#)

# updateDisplay

## updateDisplayArea

updateDisplay(void)

updateDisplayArea(uint8\_t tx, uint8\_t ty, uint8\_t tw, uint8\_t th)

**Description:** Обновляет всю или указанную прямоугольную область дисплея. Функция updateDisplay() почти идентична sendBuffer(). Функция updateDisplayArea() обновит указанную область прямоугольника: только указанная область копируется из внутреннего буфера на дисплей. Площадь указывается в плитках. Один тайл — это область размером 8x8 пикселей. Чтобы получить значение пикселя, умножьте значение тайла на 8 (для U8G2\_R0). Координаты плитки не зависят от примененного поворота в конструкторе U8g2, но имеют ту же ориентацию, что и U8G2\_R0. Для других поворотов вычисление между положением плитки значения пикселя более сложное. Три функции-члена sendBuffer, updateDisplay и updateDisplayArea предназначены для режима полного буфера (конструктор с \_F\_ в имени). Однако sendBuffer и updateDisplay также могут использоваться в страничном режиме. Если updateDisplay используется вместе с дисплеями ePaper, убедитесь, что на дисплей отправляется правильная последовательность обновления. Различия между sendBuffer, updateDisplay и updateDisplayArea:

Behavior/Feature	sendBuffer	updateDisplay	updateDisplayArea
Отправляет сообщение refreshDisplay	yes	no	no
Работает в режиме полного буфера	yes	yes	yes
Работает в режиме буфера страницы	yes	yes	no

### Arguments:

- tx, ty: Верхний левый угол области, данный как положение плитки.
- tw, th: Ширина и высота области в плитках.

### Returns: -

**Note 1:** updateDisplay() будет работать для всех контроллеров дисплея. updateDisplayArea() не будет полностью работать для следующих контроллеров: SH1122, LD7032, ST7920, ST7986, LC7981, T6963, SED1330, RA8835, MAX7219, LS0xx

**Note 2:** Диапазон для tx: 0..getBufferTileWidth()-1 and for ty: 0..getBufferTileHeight()-1. Нет проверки переполнения. Область должна полностью вписаться в область отображения. Особенно следующие условия должны быть правдой: tx+tw <= getBufferTileWidth() and ty+th <= getBufferTileHeight().

**Note 3:** setClipWindow vs updateDisplayArea: Оба могут генерировать аналогичные визуальные эффекты, однако ...

### setClipWindow

- Координаты пикселей
- Используется в цикле firstPage / nextPage
- Действительно для режима полного и страничного буфера
- Ограничит количество пикселей, помещаемых в буфер
- Повышение производительности за счет меньшего количества пикселей
- Будет работать с любой командой поворота в конструкторе
- Будет работать с любыми настройками для u8g2.setFlipMode()

### updateDisplayArea

- Координаты плитки
- Должен использоваться вне цикла firstPage / nextPage
- Действительно только для режима полного буфера
- Ограничит передачу данных на дисплей
- Повышение производительности за счет меньшей передачи данных на дисплей
- Будет работать с любой командой поворота в конструкторе, но требует более сложного расчета координат плитки, если поворот не равен U8G2\_R0.
- Будет работать с любыми настройками для u8g2.setFlipMode()

**Note 4:** Подробнеее обсуждение произошло здесь: <https://github.com/olikraus/u8g2/issues/736>

**Example:** [https://github.com/olikraus/u8g2/blob/master/sys/arduino/u8g2\\_full\\_buffer/UpdateArea/UpdateArea.ino](https://github.com/olikraus/u8g2/blob/master/sys/arduino/u8g2_full_buffer/UpdateArea/UpdateArea.ino)

**See also:** [sendBuffer](#), [getBufferTileHeight](#), [getBufferTileWidth](#), [setClipWindow](#)

## userInterfaceInputValue

```
userInterfaceInputValue(const char *title, const char *pre, uint8_t *value,
uint8_t lo, uint8_t hi, uint8_t digits, const char *post)
```

**Description:** Запрашивает ввод 8-битного значения. Весь вывод на дисплей и обработка клавиш выполняются внутри этой функции.

**Arguments:**

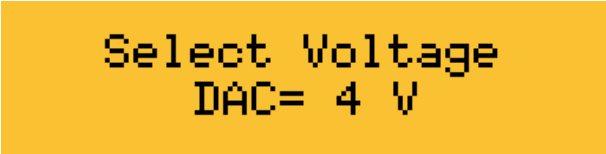
title: Многострочное описание значения (строки должны быть разделены символом \n).  
pre: Текст перед value.  
value: Указатель на переменную, которая будет заполнена вводом от пользователя.  
lo: Наименьшее значение, которое может выбрать пользователь.  
hi: Наибольшее значение, которое может выбрать пользователь.  
digits: Количество цифр (от 1 до 3).  
post: Текст после value.

**Returns:** 1, если пользователь нажал кнопку выбора. 0, если пользователь нажал кнопку home/cancel. Выбранное значение будет сохранено в value только в том случае, если пользователь нажал кнопку выбора.

**See also:** [begin](#)

**Example:**

```
u8g2.userInterfaceInputValue("Select Voltage", "DAC= ", &v, 0, 5, 1, " V");
```



Select Voltage  
DAC= 4 V

## userInterfaceMessage

```
userInterfaceMessage(const char *title1, const char *title2, const char
*title3, const char *buttons)
```

**Description:** Отображает текст сообщения и ожидает ввода пользователя. Пользователь может нажать одну кнопку или выбрать одну из двух или более кнопок.

**Arguments:**

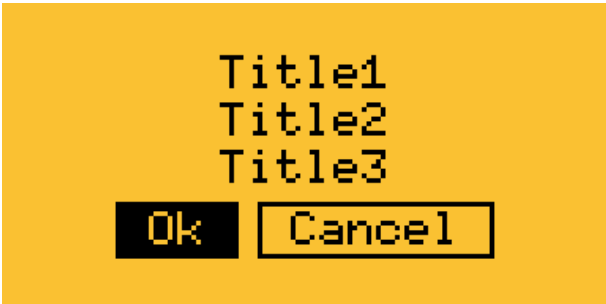
title1: Первое многострочное описание (строки должны быть разделены \n)  
title2: Второе однострочное описание (одна линия рисуется до первого \n или \0).  
title3: Третье многострочное описание (строки должны быть разделены символом \n).  
button: Одна или несколько кнопок, разделенных \n.

**Returns:** От 1 до n, если была выбрана одна из кнопок. 0, если пользователь нажал кнопку home/cancel.

**See also:** [begin](#)

**Example:**

```
u8g2.setFont(u8g2_font_6x10_tf);
u8g2.setFontRefHeightAll(); /* это добавит дополнительное пространство для текста внутри кнопок */
u8g2.userInterfaceMessage("Title1", "Title2", "Title3", " Ok \n Cancel ");
```



Title1  
Title2  
Title3

Ok

Cancel

## userInterfaceSelectionList

`userInterfaceSelectionList(const char *title, uint8_t start_pos, const char *sl)`

**Description:** Отображение списка прокручиваемых и выбираемых опций. Пользователь может выбрать один из вариантов.

**Arguments:**

- `start_pos`: Элемент, который выделяется первым (начинается с 1).
- `sl`: Список параметров, по одному на строку (строки должны быть разделены символом `\n`).

**Returns:** От 1 до n, если была выбрана одна из кнопок. 0, если пользователь нажал кнопку `home/cancel`.

**See also:** [begin](#)

**Example:**

```
u8g2.userInterfaceSelectionList("Title", 2, "abcdef\nghijkl\nmnopqr");
```

