OLED экран 128×64 / 0,96” (Trema-модуль V2.0)

Общие сведения:

Trema-модуль OLED-дисплей (128x64) - это графический дисплей, каждый пиксель которого является отдельным OLED (organic light-emitting diode) светодиодом. В отличии от TFT (Thin-Film Transistor), LCD (ЖК) и других дисплеев, этот дисплей не нуждается в подсветке, каждый пиксель светится самостоятельно, а не затемняет свет подсветки. Благодаря этому черный цвет - действительно чёрный (выключенный светодиод не светится в темноте), а белый цвет - действительно белый (не проходит через слой жидких кристаллов или тонкоплёночных транзисторов).

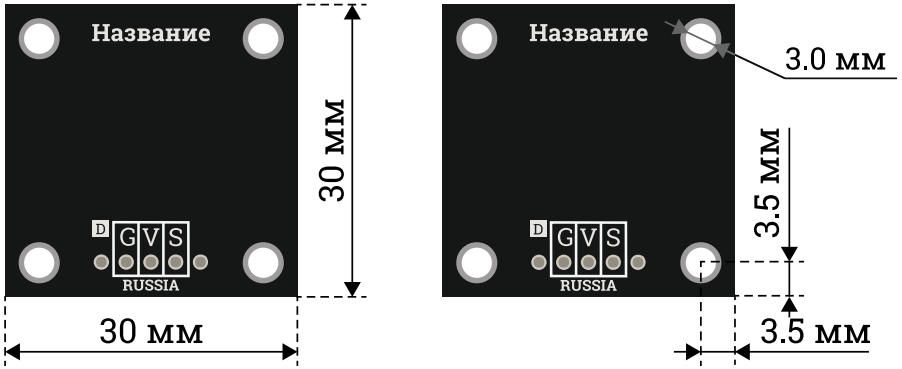
Использование органических светодиодов позволило достичь угла обзора более 160° и значительно снизить энергопотребление. Так же стоит отметить высокую контрастность (что повышает удобочитаемость текста и изображений), и небольшие размеры дисплея, всего 0.96 дюйма. Всё это, в сочетании с удобством и функциональностью, позволяет сказать что OLED-дисплей (128x64) один из лучших.

Спецификация:

- Тип дисплея: графический, OLED (organic light-emitting diode) на основе органических светодиодов.
- Тип драйвера матрицы: SSD1306.
- Разрешение: 128 x 64 точек.
- Цвет пикселей (светодиодов): белый.
- Количество цветов: белый и черный (монохромный).
- Угол обзора: > 160°.
- Коэффициент контрастности: 10000:1.
- Яркость: >120 кд/м^2.
- Напряжение питания: 3,3 ... 5 В.
- Энергопотребление: до 80 мВт (при свечении всего экрана);
- Интерфейс: I2C (поддерживается Arduino, WeMos, STM32, MSP430 и множеством других микроконтроллеров, и отладочных плат).
- Адрес на шине I2C: 0x3C или 0x3D выбирается переключателем.
- Время отклика < 10 мкс.
- Рабочая температура: -40 ... 85 °C.

- Габариты: 30x30 мм.

Все модули линейки "Trema" выполнены в одном формате



Подключение:

Дисплей подключается к аппаратной или программной шине I2C Arduino. Логические уровни шины I2C не должны превышать напряжение питания.

В комплекте имеется кабель для быстрого и удобного подключения модуля к колодке I2C на Trema Shield.

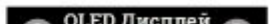
Если на шине I2C уже имеется другое устройство, то для подключения модуля, предлагаем воспользоваться I2C Hub.

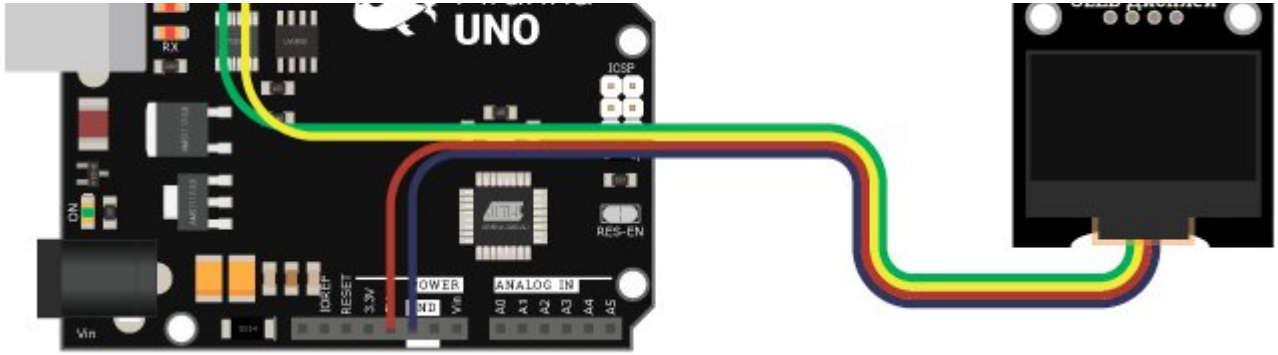
Адрес дисплея на шине I2C выбирается переключателем на обратной стороне платы.

Модуль удобно подключать 3 способами, в зависимости от ситуации:

Способ - 1 : Используя проводной шлейф и Piranha UNO

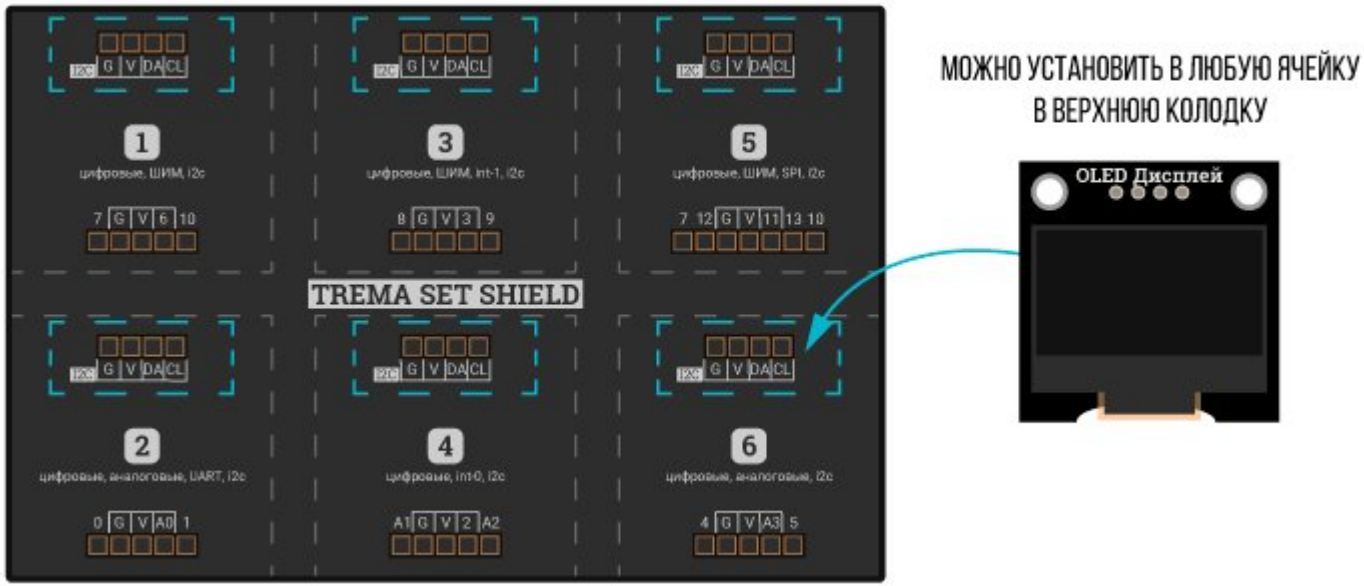
Используя провода «Папа — Мама», подключаем напрямую к контроллеру Piranha UNO.





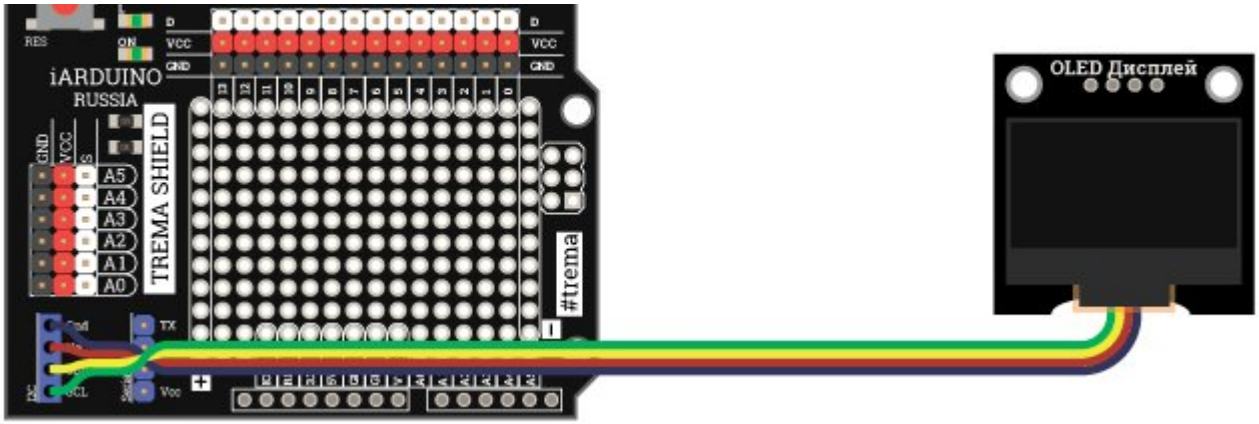
Способ - 2 : Используя Trema Set Shield

Модуль можно подключить к любому из I2C входов Trema Set Shield.



Способ - 3 : Используя проводной шлейф и Shield

Используя 4-х проводной шлейф, к Trema Shield, Trema-Power Shield, Motor Shield, Trema Shield NANO и тд.



Питание:

Входное напряжение питания от 3,3 до 5,5 В постоянного тока, подаётся на выводы Vcc и GND модуля.

Подробнее о модуле:

Матрица OLED-дисплея управляется встроенным драйвером SSD1306, отличающимся низким энергопотреблением и высокой скоростью отклика. В Trema-модуле OLED-дисплей (128x64) используется интерфейс I2C с возможностью выбора адреса на шине. Адрес выбирается переключателем установленным на обороте модуля. Доступны два адреса: 0x3C (0x78) и 0x3D (0x7A), в скобках (и на плате) указан адрес с учётом бита RW = 0. К одной шине можно подключить два OLED-дисплея (128x64), указав им разные адреса.

Специально для Trema-модуля OLED-дисплей (128x64), нами разработано сразу две библиотеки: iarduino_OLED и iarduino_OLED_txt.

Обе библиотеки позволяют указывать адрес дисплея на шине I2C. В обеих библиотеках присутствуют шрифты с поддержкой Русского языка. Как понятно из названия, библиотека iarduino_OLED_txt предназначена только для вывода текста и чисел, но благодаря этому, она использует минимум памяти ОЗУ. Библиотека iarduino_OLED может использоваться для вывода графики, изображений, текста и чисел, но как и многие другие графические библиотеки для данного типа дисплеев, она займёт не менее 1 КБ памяти ОЗУ (более половины ОЗУ Arduino UNO).

Дело в том, что драйвер дисплея SSD1306 получает данные побайтно, каждый бит полученного байта устанавливает цвет отдельного пикселя на экране. Получается что мы не можем записать цвет менее 8 пикселей, значит при записи цвета одного пикселя, мы сотрём

значения остальных 7 (если их состояние нам неизвестно). А так как драйвер SSD1306, по шинам I2C и SPI, не позволяет считывать информацию из своего ОЗУ о состоянии пикселей, то единственным выходом является создание массива в ОЗУ Arduino, данные которого целиком дублируют ОЗУ дисплея. Сначала изображение формируется в созданном массиве, а уже потом передаётся в дисплей. Разрешение экрана = 128x64 = 8192 пикселей, значит размер создаваемого массива должен быть 8192 / 8 = 1024 Байт = 1 КБ.

Подробнее про установку библиотеки читайте в нашей инструкции..

Примеры (для обеих библиотек):

Вывод времени с момента старта скетча:

Представленный ниже скетч будет выводить на дисплей, время с момента старта скетча, в формате ЧЧ:ММ:СС.

Пример с использованием текстовой библиотеки iarduino_OLED_txt:

```
#include <iarduino_OLED_txt.h>                          // Подключаем библиотеку iarduino_OLED_txt.
iarduino_OLED_txt myOLED(0x3C);                         // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                        //
extern uint8_t MediumFont[];                            // Подключаем шрифт MediumFont.
uint32_t       i;                                       // Объявляем переменную для хранения времени прошедшего с момента с
int            h, m, s;                                 // Объявляем переменную для хранения времени в часах, минутах и сек
                                                        //
void setup(){                                           //
    myOLED.begin();                                     // Инициируем работу с дисплеем.
    myOLED.setFont(MediumFont);                         // Указываем шрифт который требуется использовать для вывода цифр и
}                                                       //
void loop(){                                            //
    i=millis();                                         // Получаем количество миллисекунд прошедшее с момента старта скетч
    if(i%1000==0){ delay(1);                            // Выполняем скетч 1 раз в секунду. Так как условие истинно в течен
        i/=1000; h=i/60/60%24; m=i/60%60; s=i%60;       // Рассчитываем часы, минуты и секунды.
                                  myOLED.setCursor(16,4); // Устанавливаем курсор в 16 столбец 4 строки.
        if(h<10){myOLED.print(0);} myOLED.print(h);     // Выводим часы прошедшие с момента старта скетча, в формате ЧЧ.
                                  myOLED.print(":");    // Выводим текст состоящий из одного символа «:»
```

```
        if(m<10){myOLED.print(0);} myOLED.print(m);        // Выводим минуты прошедшие с момента старта скетча, в формате ММ.
                                   myOLED.print(":");      // Выводим текст состоящий из одного символа «:»
        if(s<10){myOLED.print(0);} myOLED.print(s);        // Выводим секунды прошедшие с момента старта скетча, в формате СС.
    }                                                      //
}                                                          //
```

**Пример с использованием графической библиотеки [iarduino_OLED](): **

```
#include <iarduino_OLED.h>                                 // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);                                // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                           //
extern uint8_t MediumFont[];                               // Подключаем шрифт MediumFont.
uint32_t       i;                                          // Объявляем переменную для хранения времени прошедшего с момента с
int            h, m, s;                                    // Объявляем переменную для хранения времени в часах, минутах и сек
                                                           //
void setup(){                                              //
    myOLED.begin();                                        // Инициируем работу с дисплеем.
    myOLED.setFont(MediumFont);                            // Указываем шрифт который требуется использовать для вывода цифр и
}                                                          //
void loop(){                                               //
    i=millis();                                            // Получаем количество миллисекунд прошедшее с момента старта скетч
    if(i%1000==0){ delay(1);                               // Выполняем скетч 1 раз в секунду. Так как условие истинно в течен
        i/=1000; h=i/60/60%24; m=i/60%60; s=i%60;          // Рассчитываем часы, минуты и секунды.
                                   myOLED.setCursor(16,39);// Устанавливаем курсор в координату 16:31, это будет нижняя левая
        if(h<10){myOLED.print(0);} myOLED.print(h);        // Выводим часы прошедшие с момента старта скетча, в формате ЧЧ.
                                   myOLED.print(":");      // Выводим текст состоящий из одного символа «:»
        if(m<10){myOLED.print(0);} myOLED.print(m);        // Выводим минуты прошедшие с момента старта скетча, в формате ММ.
                                   myOLED.print(":");      // Выводим текст состоящий из одного символа «:»
        if(s<10){myOLED.print(0);} myOLED.print(s);        // Выводим секунды прошедшие с момента старта скетча, в формате СС.
    }                                                      //
}                                                          //
```

```
                                                                                //
```

Оба примера работают одинаково. Код скетчей отличается названием подключаемой библиотеки (1 и 2 строки) и номером вертикальной координаты (16 строка). В первом примере (для текстовой библиотеки) указывается номер строки (от 0 до 7), а во втором примере (для графической библиотеки) указывается номер пикселя (от 0 до 63).

В представленных выше примерах, координата начала вывода текста указана функцией setCursor(x,y), но координаты можно указывать и в функциях print("текст",x,y), как это будет сделано в следующих примерах.

Вывод текста на два дисплея:

Представленный ниже скетч выводит номер и адрес дисплея. На дисплеях должны быть установлены разные адреса (установка осуществляется переключателем на обратной стороне платы), иначе на обоих дисплеях будет одинаковая информация. Данным скетчем можно точно определить адрес Вашего дисплея на шине I2C.

Пример с использованием текстовой библиотеки [iarduino_OLED_txt]:

```
#include <iarduino_OLED_txt.h>                          // Подключаем библиотеку iarduino_OLED_txt.
iarduino_OLED_txt myOLED_1(0x3C);                       // Объявляем объект myOLED_1, указывая адрес первого дисплея на шин
iarduino_OLED_txt myOLED_2(0x3D);                       // Объявляем объект myOLED_2, указывая адрес второго дисплея на шин
                                                        //
extern uint8_t MediumFontRus[];                         // Подключаем шрифт MediumFontRus.
extern uint8_t SmallFontRus[];                          // Подключаем шрифт SmallFontRus.
                                                        // Если Вы не используете Кириллицу, то лучше подключить шрифты Мед
void setup(){                                           //
    myOLED_1.begin  ();                                 // Инициируем работу с первым дисплеем.
    myOLED_2.begin  ();                                 // Инициируем работу с вторым дисплеем.
    myOLED_1.setFont(MediumFontRus);                    // Указываем шрифт который требуется использовать для вывода цифр и
    myOLED_2.setFont(MediumFontRus);                    // Указываем шрифт который требуется использовать для вывода цифр и
    myOLED_1.print  ("1 дисплей", OLED_C, 3);           // Выводим текст на первый дисплей по центру 3 строки (высота шрифт
    myOLED_2.print  ("2 дисплей", OLED_C, 3);           // Выводим текст на второй дисплей по центру 3 строки (высота шрифт
```

```
    myOLED_1.setFont(SmallFontRus);                     // Указываем шрифт который требуется использовать для вывода цифр и
    myOLED_2.setFont(SmallFontRus);                     // Указываем шрифт который требуется использовать для вывода цифр и
    myOLED_1.print  ("Адрес дисплея 0x3C", OLED_C, 5);  // Выводим текст на первый дисплей по центру 5 строки (высота шрифт
    myOLED_2.print  ("Адрес дисплея 0x3D", OLED_C, 5);  // Выводим текст на второй дисплей по центру 5 строки (высота шрифт
}                                                       //
void loop(){}                                           //
```

Пример с использованием графической библиотеки iarduino_OLED:

Пример с использованием графической библиотеки iarduino_OLED, так же должен отличаться названием библиотеки (1, 2 и 3 строки) и вертикальной координатой выводимых строк (13, 14, 17 и 18 строки), но он не сможет работать на Arduino UNO, так как для каждого объекта библиотека создаст буфер размером в 1 КБ, что займет всю память ОЗУ Arduino UNO.

Демонстрация вывода текста Кириллицей с указанием разных кодировок:

Представленные ниже скетчи позволят выводить Русский текст при работе в разных версиях Arduino IDE и ОС отличных от Windows.

Пример с использованием текстовой библиотеки iarduino_OLED_txt:

```
#include <iarduino_OLED_txt.h>                  // Подключаем библиотеку iarduino_OLED_txt.
iarduino_OLED_txt myOLED(0x3C);                 // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                //
extern uint8_t SmallFontRus[];                  // Подключаем шрифт SmallFontRus.
                                                //
void setup(){                                   //
    myOLED.begin();                             // Инициируем работу с дисплеем.
    myOLED.setFont(SmallFontRus);               // Указываем шрифт который требуется использовать для вывода цифр и
}                                               //
void loop(){                                    //
//  Вывод текста в кодировке UTF-8:             //
    myOLED.clrScr();                            // Чистим экран.
    myOLED.print("UTF8", 0, 0);                 // Выводим текст начиная с 0 столбца 0 строки.
```

```
    myOLED.setCoding(TXT_UTF8);                 // Меняем кодировку на UTF-8 (по умолчанию).
    myOLED.print("Ардуино iArduino", OLED_C, 4); // Выводим текст по центру 4 строки.
    delay (5000);                               // Ждём 5 секунд.
                                                //
//  Вывод текста в кодировке CP866:             //
    myOLED.clrScr();                            // Чистим экран.
    myOLED.print("CP866", 0, 0);                // Выводим текст начиная с 0 столбца 0 строки.
    myOLED.setCoding(TXT_CP866);                // Меняем кодировку на CP866.
    myOLED.print("Ардуино iArduino", OLED_C, 4); // Выводим текст по центру 4 строки.
    delay (5000);                               // Ждём 5 секунд.
                                                //
//  Вывод текста в кодировке WINDOWS-1251:      //
    myOLED.clrScr();                            // Чистим экран.
    myOLED.print("WIN1251", 0, 0);              // Выводим текст начиная с 0 столбца 0 строки.
    myOLED.setCoding(TXT_WIN1251);              // Меняем кодировку на WINDOWS-1251.
    myOLED.print("Ардуино iArduino", OLED_C, 4); // Выводим текст по центру 4 строки.
    delay (5000);                               // Ждём 5 секунд.
}                                               //
```

Пример с использованием графической библиотеки [iarduino_OLED]:

```
#include <iarduino_OLED.h>                      // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);                     // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                //
extern uint8_t SmallFontRus[];                  // Подключаем шрифт SmallFontRus.
                                                //
void setup(){                                   //
    myOLED.begin();                             // Инициируем работу с дисплеем.
    myOLED.setFont(SmallFontRus);               // Указываем шрифт который требуется использовать для вывода цифр и
}                                               //
void loop(){                                    //
```

```
//  Вывод текста в кодировке UTF-8:                    //
    myOLED.clrScr();                                   // Чистим экран.
    myOLED.print("UTF8", 0, 7);                        // Выводим текст начиная с координаты 0x7 (нижняя левая точка текст
    myOLED.setCoding(TXT_UTF8);                        // Меняем кодировку на UTF-8 (по умолчанию).
    myOLED.print("Ардуино iArduino", OLED_C, 39);      // Выводим текст по центру экрана, координата нижней части текста п
    delay (5000);                                      // Ждём 5 секунд.
                                                       //
//  Вывод текста в кодировке CP866:                    //
    myOLED.clrScr();                                   // Чистим экран.
    myOLED.print("CP866", 0, 7);                       // Выводим текст начиная с координаты 0x7 (нижняя левая точка текст
    myOLED.setCoding(TXT_CP866);                       // Меняем кодировку на CP866.
    myOLED.print("Ардуино iArduino", OLED_C, 39);      // Выводим текст по центру экрана, координата нижней части текста п
    delay (5000);                                      // Ждём 5 секунд.
                                                       //
//  Вывод текста в кодировке WINDOWS-1251:             //
    myOLED.clrScr();                                   // Чистим экран.
    myOLED.print("WIN1251", 0, 7);                     // Выводим текст начиная с координаты 0x7 (нижняя левая точка текст
    myOLED.setCoding(TXT_WIN1251);                     // Меняем кодировку на WINDOWS-1251.
    myOLED.print("Ардуино iArduino", OLED_C, 39);      // Выводим текст по центру экрана, координата нижней части текста п
    delay (5000);                                      // Ждём 5 секунд.
```

Оба примера работают одинаково. Код скетчей отличается названием подключаемой библиотеки (1 и 2 строки) и номером вертикальной координаты (13,15, 20,22, 27,29 строки). В первом примере (для текстовой библиотеки) указывается номер строки (от 0 до 7), а во втором примере (для графической библиотеки) указывается номер пикселя (от 0 до 63).

Если у Вас ОС Windows, Вы работаете в среде Arduino IDE 1.8.X и скетч был сохранён перед загрузкой, то указывать кодировку нет необходимости (символы будут переданы компилятору в кодировке UTF-8, которая используется библиотекой по умолчанию). Но в некоторых случаях может понадобиться явное указание кодировки. Например если у Вас ОС Windows, Вы работаете в среде Arduino IDE 1.8.X и скетч не был сохранён перед загрузкой, то все символы будут переданы компилятору в кодировке WINDOWS-1251.

Вывод символов по их коду:

Представленные ниже скетчи могут понадобиться для указания Русских или специальных символов в строках. Этот пример можно использовать не только для создания строк выводимых на дисплей. Описанный метод действует для создания любых строк в Arduino IDE (так, например, можно создавать строки для вывода в монитор последовательного порта).

Пример с использованием текстовой библиотеки iarduino_OLED_txt:

```
#include <iarduino_OLED_txt.h>                                          // Подключаем библиотеку iarduino_OLED_txt.
iarduino_OLED_txt myOLED(0x3C);                                         // Объявляем объект myOLED, указывая адрес дисплея на ш
                                                                        //
extern uint8_t SmallFontRus[];                                          // Подключаем шрифт SmallFontRus.
                                                                        //
void setup(){                                                           //
    myOLED.begin();                                                     // Инициируем работу с дисплеем.
    myOLED.setFont(SmallFontRus);                                       // Указываем шрифт который требуется использовать для в
    myOLED.setCoding(false);                                            // Отменяем текущую кодировку, так как Русские буквы бу
    myOLED.print("\200\340\244\343\250\255\256 iArduino", OLED_C, 4);   // Выводим текст предыдущего примера "Ардуино iArduino"
                                                                        // Вместо Русских букв используем их код.
void loop(){}                                                           // Данный пример будет работать вне зависимости от Ваше
```

Пример с использованием графической библиотеки iarduino_OLED:

```
#include <iarduino_OLED.h>                                              // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);                                             // Объявляем объект myOLED, указывая адрес дисплея на ш
                                                                        //
extern uint8_t SmallFontRus[];                                          // Подключаем шрифт SmallFontRus.
                                                                        //
void setup(){                                                           //
    myOLED.begin();                                                     // Инициируем работу с дисплеем.
    myOLED.setFont(SmallFontRus);                                       // Указываем шрифт который требуется использовать для в
    myOLED.setCoding(false);                                            // Отменяем текущую кодировку, так как Русские буквы бу
```

```
    myOLED.print("\200\340\244\343\250\255\256 iArduino", OLED_C, 39); // Выводим текст "Ардуино iArduino" по центру экрана, к
                                                                       // Вместо Русских букв используем их код.
void loop(){}                                                          // Данный пример будет работать вне зависимости от Ваше
```

Оба примера работают одинаково. Код скетчей отличается названием подключаемой библиотеки (1 и 2 строки) и номером вертикальной координаты (10 строка). В первом примере (для текстовой библиотеки) указывается номер строки (от 0 до 7), а во втором примере (для графической библиотеки) указывается номер пикселя (от 0 до 63).

В приведённых выше примерах все Русские символы вводятся кодом по кодировке CP866 (именно в этой кодировке символы располагаются в библиотечных шрифтах), а латинские буквы (A-Z, a-z), цифры (0-9) и символы !"#$%&'()*+,-./:;<=>?@[]^_`{|}~ вводятся как есть, вне зависимости от используемой кодировки.

Таблица кодов Русских букв в кодировке CP866:

А	128 \200	**И**	136 \210	**Р**	144 \220	**Ш**	152 \230	**а**	160 \240	**и**	168 \250	**р**	224 \340	**ш**	232 \350	**Ё**	240 \360
Б	129 \201	**Й**	137 \211	**С**	145 \221	**Щ**	153 \231	**б**	161 \241	**й**	169 \251	**с**	225 \341	**щ**	233 \351	**ё**	241 \361
В	130 \202	**К**	138 \212	**Т**	146 \222	**Ъ**	154 \232	**в**	162 \242	**к**	170 \252	**т**	226 \342	**ъ**	234 \352		242 \362
Г	131 \203	**Л**	139 \213	**У**	147 \223	**Ы**	155 \233	**г**	163 \243	**л**	171 \253	**у**	227 \343	**ы**	235 \353		243 \363
Д	132 \204	**П**	140 \214	**Ф**	148 \224	**Ь**	156 \234	**д**	164 \244	**м**	172 \254	**ф**	228 \344	**ь**	236 \354		244 \364
Е	133 \205	**Н**	141 \215	**Х**	149 \225	**Э**	157 \235	**е**	165 \245	**н**	173 \255	**х**	229 \345	**э**	237 \355		245 \365

Ж	134 \206	**О**	142 \216	**Ц**	150 \226	**Ю**	158 \236	**ж**	166 \246	**о**	174 \256	**ц**	230 \346	**ю**	238 \356		246 \366
З	135 \207	**П**	143 \217	**Ч**	151 \227	**Я**	159 \237	**з**	167 \247	**п**	175 \257	**ч**	231 \347	**я**	239 \357		247 \367

Для вывода любого символа **нужно указать его код в 8-ричной системе счисления**, которому должен предшествовать обратный слеш «****». Данное правило действует для любых строк в Arduino IDE. Первая буква в строке примера - «А», имеет код 128. Если перевести 128 в 8-ричную систему счисления, получится $(200)_8$. Значит букву «А» можно записать как «\200», букву «р» как «\340», букву «д» как «\244» и т.д.

Для перевода чисел из 10-тичной в 8-ричную систему предлагаем воспользоваться стандартным калькулятором Windows. Откройте калькулятор, выберите вид калькулятора - «Программист» и введите число, Вы увидите его представление в разных системах счисления: HEX(16), DEC(10), OCT(8) и BIN(2).

Но если Вы желаете научиться быстро переводить числа между системами счисления 2, 4, 8, 10, 16, без калькулятора, то посмотрите Урок 32 - перевод чисел между системами счисления.

Запись текста кодами символов кажется более громоздкой, но на самом деле, такая запись занимает в 2 раза меньше памяти, чем строки записанные Русскими символами в кодировке UTF-8!

Если Вы пишите скетч с большим количеством выводимых строк и у Вас не хватает памяти ОЗУ, то предлагаем Вам использовать такой вид записи:

```
myOLED.print( F("Строка для дисплея") );  // Вывод строки на дисплей.
Serial.print( F("Строка для монитора") ); // Вывод строки в монитор последовательного порта.
```

Теперь выводимые Вами строки будут храниться в области памяти программ, не используя память ОЗУ.

Вывод текста черными буквами на белом фоне:

Иногда может понадобиться выделить заголовок, это можно сделать используя другой шрифт или просто инвертировав цвет текста. По

умолчанию текст выводится белыми буквами на чёрном фоне, но после вызова функции invText(true) текст будет черным на белом фоне (параметр true функции invText можно не указывать). Если вызвать функцию invText(false), то текст выводимый после обращения к данной функции, опять будет выводиться белыми буквами на чёрном фоне.

Пример с использованием текстовой библиотеки iarduino_OLED_txt:

```
#include <iarduino_OLED_txt.h>                          // Подключаем библиотеку iarduino_OLED_txt.
iarduino_OLED_txt myOLED(0x3C);                         // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                        //
extern uint8_t SmallFontRus[];                          // Подключаем шрифт SmallFontRus.
                                                        // Если Вы не используете Кириллицу, то лучше подключить шрифт Smal
void setup(){                                           //
    myOLED.begin();                                     // Инициируем работу с дисплеем.
    myOLED.setFont(SmallFontRus);                       // Указываем шрифт который требуется использовать для вывода цифр и
    myOLED.invText();                                   // Указываем что цвет текста выводимого после данной функции должен
    myOLED.print("*** iarduino.ru ***", OLED_C, 3);     // Выводим текст по центру 3 строки. Текст будет написан чёрными бу
    myOLED.invText(false);                              // Указываем что цвет текста выводимого после данной функции не тре
    myOLED.print("*** iarduino.ru ***", OLED_C, 5);     // Выводим текст по центру 5 строки. Текст будет написан белыми бук
}                                                       //
void loop(){}                                           //
```

Пример с использованием графической библиотеки iarduino_OLED:

```
#include <iarduino_OLED.h>                              // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);                             // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                        //
extern uint8_t SmallFontRus[];                          // Подключаем шрифт SmallFontRus.
                                                        // Если Вы не используете Кириллицу, то лучше подключить шрифт Smal
void setup(){                                           //
    myOLED.begin();                                     // Инициируем работу с дисплеем.
```

```
    myOLED.setFont(SmallFontRus);                       // Указываем шрифт который требуется использовать для вывода цифр и
    myOLED.invText();                                   // Указываем что цвет текста выводимого после данной функции должен
    myOLED.print("*** iarduino.ru ***", OLED_C, 31);    // Выводим текст по центру экрана, координата нижней части текста п
    myOLED.invText(false);                              // Указываем что цвет текста выводимого после данной функции не тре
    myOLED.print("*** iarduino.ru ***", OLED_C, 47);    // Выводим текст по центру экрана, координата нижней части текста п
}                                                       //
void loop(){}                                           //
```

Оба примера работают одинаково. Код скетчей отличается названием подключаемой библиотеки (1 и 2 строки) и номером вертикальной координаты (10 и 12 строки). В первом примере (для текстовой библиотеки) указывается номер строки (от 0 до 7), а во втором примере (для графической библиотеки) указывается номер пикселя (от 0 до 63).

Вывод чисел:

Этот пример демонстрирует вывод положительных, отрицательных, целых и дробных чисел. Для целых чисел можно указывать требуемую систему счисления.

Пример с использованием текстовой библиотеки iarduino_OLED_txt:

```
#include <iarduino_OLED_txt.h>                          // Подключаем библиотеку iarduino_OLED_txt.
iarduino_OLED_txt myOLED(0x3C);                         // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                        //
extern uint8_t SmallFontRus[];                          // Подключаем шрифт SmallFontRus.
                                                        // Если Вы не используете Кириллицу, то лучше подключить шрифт Smal
void setup(){                                           //
    myOLED.begin();                                     // Инициируем работу с дисплеем.
    myOLED.setFont(SmallFontRus);                       // Указываем шрифт который требуется использовать для вывода цифр и
                                                        //
    myOLED.print( 123456789 , 0, 0);                    // Выводим целое положительное число начиная с 0 столбца 0 строки.
    myOLED.print(-123456789 , 0, 1);                    // Выводим целое отрицательное число начиная с 0 столбца 1 строки.
    myOLED.print( 123456789 , 0, 2, HEX);               // Выводим целое положительное число начиная с 0 столбца 2 строки,
```

```
    myOLED.print( 123456789 , 0, 3, OCT);          // Выводим целое положительное число начиная с 0 столбца 3 строки,
    myOLED.print(-123.456789, 0, 4);               // Выводим число с плавающей точкой  начиная с 0 столбца 4 строки,
    myOLED.print( 123.456789, 0, 5, 3);            // Выводим число с плавающей точкой  начиная с 0 столбца 5 строки,
    myOLED.print( 123       , 0, 6, BIN);          // Выводим целое положительное число начиная с 0 столбца 6 строки,
    myOLED.print( 123       , 0, 7, 12);           // Выводим целое положительное число начиная с 0 столбца 7 строки,
}                                                  //
void loop(){}                                      //
```

Пример с использованием графической библиотеки [iarduino_OLED]:

```
#include <iarduino_OLED.h>                         // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);                        // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                   //
extern uint8_t SmallFontRus[];                     // Подключаем шрифт SmallFontRus.
                                                   // Если Вы не используете Кириллицу, то лучше подключить шрифт Smal
void setup(){                                      //
    myOLED.begin();                                // Инициируем работу с дисплеем.
    myOLED.setFont(SmallFontRus);                  // Указываем шрифт который требуется использовать для вывода цифр и
                                                   //
    myOLED.print( 123456789 , 0, 7 );              // Выводим целое положительное число начиная с координаты 0x0.
    myOLED.print(-123456789 , 0, 15);              // Выводим целое отрицательное число начиная с координаты 0x15.
    myOLED.print( 123456789 , 0, 23, HEX);         // Выводим целое положительное число начиная с координаты 0x23, в 1
    myOLED.print( 123456789 , 0, 31, OCT);         // Выводим целое положительное число начиная с координаты 0x31, в 8
    myOLED.print(-123.456789, 0, 39);              // Выводим число с плавающей точкой  начиная с координаты 0x39, по
    myOLED.print( 123.456789, 0, 47, 3);           // Выводим число с плавающей точкой  начиная с координаты 0x47, ука
    myOLED.print( 123       , 0, 55, BIN);         // Выводим целое положительное число начиная с координаты 0x55, в 2
    myOLED.print( 123       , 0, 63, 12);          // Выводим целое положительное число начиная с координаты 0x63, в 1
}                                                  //
                                                   //
void loop(){}                                      //
```

Оба примера работают одинаково. Код скетчей отличается названием подключаемой библиотеки (1 и 2 строки) и номером вертикальной координаты (10 - 17 строки). В первом примере (для текстовой библиотеки) указывается номер строки (от 0 до 7), а во втором примере (для графической библиотеки) указывается номер пикселя (от 0 до 63).

Инверсия цвета дисплея:

Следующий пример демонстрирует инверсию цвета дисплея. Все черные пиксели станут белыми и наоборот. Действие функции invScr() распространяется как на уже выведенное изображение дисплея, так и на выводимое после.

```
...                                                 // Подключаем библиотеку и объявляем объект.
void setup(){                                       //
    ...                                             // Выводим данные на дисплей.
}                                                   //
loop(){                                             //
    invScr(true);  delay(2000);                     // Инвертируем цвет дисплея.
    invScr(false); delay(2000);                     // Возвращаем нормальное отображение данных.
}                                                   //
```

Данный пример будет менять цвета дисплея через каждые две секунды, он будет работать как с текстовой iarduino_OLED_txt, так и с графической iarduino_OLED библиотеками.

Примеры (только для графической библиотеки):

Установка прозрачности фона текста:

В данном примере два текста накладываются друг на друга, но текст выведенный после функции bgText(false) не имеет фонового цвета и не закрашивает данные находящиеся под ним. Если функция bgText(false) не вызывалась, или вызывалась с параметром true, то текст будет иметь фоновый цвет, который закрасит находящиеся под ним данные.

```
#include <iarduino_OLED.h>                          // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);                         // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                    //
```

```
extern uint8_t MediumFontRus[];                    // Подключаем шрифт MediumFontRus.
                                                   // Если Вы не используете Кириллицу, то лучше подключить шрифт Medi
void setup(){                                      //
    myOLED.begin();                                // Инициируем работу с дисплеем.
    myOLED.setFont(MediumFontRus);                 // Указываем шрифт который требуется использовать для вывода цифр и
}                                                  //
void loop(){                                       //
    myOLED.bgText(false);                          // Указываем что у текста нет фона.
    myOLED.clrScr();                               // Чистим экран.
    myOLED.print("arduino", 0, 32);                // Выводим текст начиная с координаты 0x32.
    myOLED.print("arduino", 5, 39);                // Выводим текст начиная с координаты 0x39. Этот текст частично нал
    delay(5000);                                   //
    myOLED.bgText(true);                           // Указываем что у текста есть фон (по умолчанию).
    myOLED.clrScr();                               // Чистим экран.
    myOLED.print("arduino", 0, 32);                // Выводим текст начиная с координаты 0x32.
    myOLED.print("arduino", 5, 39);                // Выводим текст начиная с координаты 0x39. Этот текст частично нал
    delay(5000);                                   //
}                                                  //
```

Аналогичным образом работает и функция bgImage, но не для текста, а для выводимых изображений.

Вывод предустановленных изображений:

Данные примеры демонстрируют вывод небольших изображений входящих в состав библиотеки. Эти изображения хранятся как массивы в файле «DefaultImage.c» в папке «src» библиотеки iarduino_OLED. Вы можете открыть этот файл в любом текстовом редакторе и увидеть названия всех имеющихся изображений, и способ объявления массивов. Так как предустановленные изображения находятся в отдельном файле, то перед выводом на экран, нужно подключить те массивы изображения которых Вы хотите использовать в скетче.

Список предустановленных изображений и их названий:

- Img_Battery_charging	- Img_Arrow_down	- Img_Level_1	- Img_Alarm

- Img_Battery_low	- Img_Arrow_left	- Img_Level_2	- Img_Antenna
- Img_Battery_0	- Img_Arrow_right	- Img_Level_3	- Img_Bluetooth
- Img_Battery_1	- Img_Arrow_up	- Img_Level_4	- Img_Message
- Img_Battery_2	- Img_Dynamic	- Img_Netlevel_1	- Img_Light
- Img_Battery_3	- Img_Dynamic_off	- Img_Netlevel_2	- Img_Melody
- Img_Call	- Img_Dynamic_on	- Img_Netlevel_3	- Img_Check
- Img_Call_in	- Img_Checkbox_off	- Img_Radio_off	- Img_Settings
- Img_Call_out	- Img_Checkbox_on	- Img_Radio_on	
- Img_BigBattery_low	- Img_Logo		

Узнать размеры картинок в пикселях можно функциями getImageWidth() и getImageHeight():

```
uint8_t X = myOLED.getImageWidth(Img_Logo);              // Ширина картинки из массива Img_Logo в пикселях.
uint8_t Y = myOLED.getImageHeight(Img_Logo);             // Высота картинки из массива Img_Logo в пикселях.
```

Вывод нескольких изображений на экран дисплея:

```
#include <iarduino_OLED.h>                               // Подключаем библиотеку iarduino_OLED.
iarduino_OLED  myOLED(0x3C);                             // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                         //
extern uint8_t Img_Level_4[];                            // Подключаем изображение Img_Level_4   из встроенных картинок.
extern uint8_t Img_Antenna[];                            // Подключаем изображение Img_Antenna   из встроенных картинок.
extern uint8_t Img_Battery_0[];                          // Подключаем изображение Img_Battery_0 из встроенных картинок.
extern uint8_t Img_Battery_1[];                          // Подключаем изображение Img_Battery_1 из встроенных картинок.
extern uint8_t Img_Battery_2[];                          // Подключаем изображение Img_Battery_2 из встроенных картинок.
extern uint8_t Img_Battery_3[];                          // Подключаем изображение Img_Battery_3 из встроенных картинок.
extern uint8_t Img_BigBattery_low[];                     // Подключаем изображение Img_BigBattery_low из встроенных картинок
```

```
      uint8_t i=0;                                          // Определяем переменную которая будет хранить номер от 0 до 3.
                                                            //
void setup(){                                               //
    myOLED.begin();                                         // Инициируем работу с дисплеем.
    myOLED.drawImage(Img_Level_4       , 0     , 7     );  // Выводим картинку Img_Level_4, указав координату её левого нижнег
    myOLED.drawImage(Img_Antenna       , 12    , 7     );  // Выводим картинку Img_Antenna, указав координату её левого нижнег
    myOLED.drawImage(Img_BigBattery_low, OLED_C, OLED_C);  // Выводим картинку Img_BigBattery_low по центру экрана.
}                                                           //
void loop(){                                                //
    while(millis()%300==0){                                 // Выполняем код в теле оператора while через каждые 300 мс.
      i++; if(i>3){i=0;}                                    // Увеличиваем значение переменной i и обнуляем её если значение ст
      switch(i){                                            // Создаём анимацию из четырех картинок, меняя их по значению перем
        case 0: myOLED.drawImage(Img_Battery_0, OLED_R, OLED_T); break; // Выводим изображение Img_Battery_0 в правом верхнем
        case 1: myOLED.drawImage(Img_Battery_1, OLED_R, OLED_T); break; // Выводим изображение Img_Battery_0 в правом верхнем
        case 2: myOLED.drawImage(Img_Battery_2, OLED_R, OLED_T); break; // Выводим изображение Img_Battery_0 в правом верхнем
        case 3: myOLED.drawImage(Img_Battery_3, OLED_R, OLED_T); break; // Выводим изображение Img_Battery_0 в правом верхнем
      }                                                     //
    }                                                       //
}                                                           //
```

Вывод логотипа и текста iArduino:

```
#include <iarduino_OLED.h>                                  // Подключаем библиотеку iarduino_OLED.
iarduino_OLED  myOLED(0x3C);                                // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                            //
extern uint8_t SmallFontRus[];                              // Подключаем шрифт SmallFontRus.
extern uint8_t MediumFont[];                                // Подключаем шрифт MediumFont. Если Вы желаете использовать Кирилл
extern uint8_t Img_Logo[];                                  // Подключаем изображение Img_Logo из встроенных картинок.
                                                            //
void setup(){                                               //
    myOLED.begin();                                         // Инициируем работу с дисплеем.
```

```
                                                    //
    myOLED.drawImage(Img_Logo, 0, 30);              // Выводим картинку Img_Logo в позицию 0x30.
                                                    //
    myOLED.setFont(MediumFont);                     // Указываем шрифт который требуется использовать для вывода цифр и
    myOLED.print("iarduino", OLED_R);               // Выводим текст "iarduino", выравнивая его по правому краю. Нижний
    myOLED.setFont(SmallFontRus);                   // Указываем шрифт который требуется использовать для вывода цифр и
    myOLED.print("всё для радиолюбителя", 0, 41);   // Выводим текст "всё для радиолюбителя".
}                                                   //
void loop(){}                                       //
```

В следующем примере описан алгоритм создания массива изображения. Вы можете создать свой файл со своими массивами (на подобии «DefaultImage.c») сохранив его в одной директории со скетчем, тогда изображения из Вашего файла, так же можно будет выводить на дисплей.

Вывод собственных изображений:

Данный пример демонстрирует создание массивов изображений в области ОЗУ (RAM) или ПЗУ (ROM), и вывод этих изображений на дисплей. В скетче создаются два массива myImageInRAM и myImageInROM содержащие одни и те же данные (одно и то же изображение), но массив myImageInRAM будет храниться в области ОЗУ, а массив myImageInROM в области ПЗУ, так как он был объявлен с модификатором PROGMEM. Далее, в коде setup, оба изображения выводятся на дисплей, с указанием координат левой нижней точки изображения и типа памяти в которой хранится массив.

```
#include <iarduino_OLED.h>                          // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);                         // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                    //
const uint8_t myImageInRAM[] =                      // Создаём массив myImageInRAM содержащий изображение смайлика. Мас
{   9, 8,             //              ЛЕВО           // Первые два числа, это размеры изображения: ширина 9px, высота 8p
    B00111100,        //              ####           // = 0x3C
    B01000010,        //             #    #          // = 0x42
    B10010001,        //            #  #   #         // = 0x91      Если байты массива указывать не в двоичной,
    B10100101,        //            # #  # #         // = 0xA5      а в шестнадцатеричной системе счисления,
```

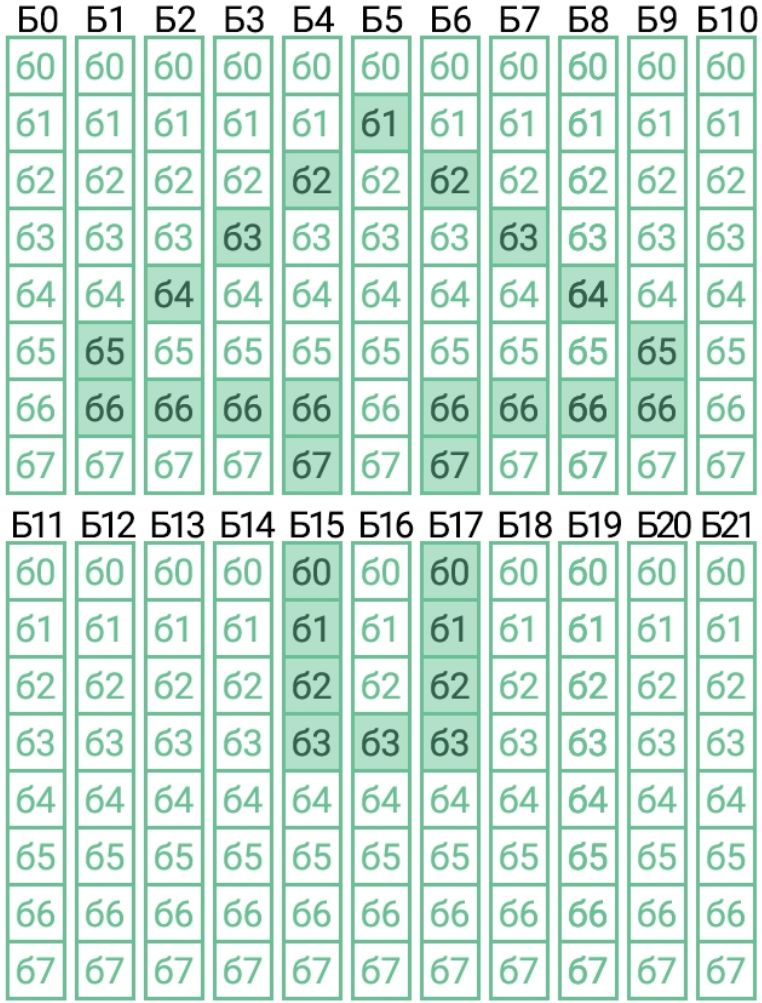
```
    B10100001,        //       НИЗ  # #    #  ВЕРХ        // = 0xA1      то запись массива будет более короткой:
    B10100101,        //            # #  # #               // = 0xA5
    B10010001,        //            #  #   #               // = 0x91      const uint8_t myImageInRAM[]={9,8,0x3C,0x42,0x91,0xA
    B01000010,        //             #    #                // = 0x42
    B00111100         //              ####                 // = 0x3C
};                    //             ПРАВО                 //
                                                           //
const uint8_t myImageInROM[] PROGMEM =                     // Создаём массив myImageInROM содержащий изображение смайлика. Мас
{9,8, 0x3C,0x42,0x91,0xA5,0xA1,0xA5,0x91,0x42,0x3C};       // Данные этого массива аналогичны предыдущему, но записаны в 16-ри
                                                           //
void setup(){                                              //
    myOLED.begin();                                        // Инициируем работу с дисплеем.
                                                           //
    myOLED.drawImage(myImageInRAM, 36, 27, IMG_RAM);       // Выводим картинку myImageInRAM в позицию 36x27, указав, что карти
    myOLED.drawImage(myImageInROM, 81, 27, IMG_ROM);       // Выводим картинку myImageInRAM в позицию 81x27, указав, что карти
                                                           //
}                                                          //
void loop(){}                                              //
```

Данные первого массива можно было записать в шестнадцатеричной системе счисления, как у второго массива, но для примера они представлены в двоичном коде, так легче объяснить принцип создания изображений:

Посмотрите на то, как создан первый массив - myImageInRAM. Он начинается с двух чисел определяющих размер изображения 9,8 (ширина 9 пикселей, высота 8 пикселей). Далее следуют байты, каждый бит которых отвечает за цвет пикселя (0-черный, 1-белый). Если заменить 0 на пробел, а 1 на символ "#", как это сделано в комментарии (правее данных), то можно увидеть изображение смайлика (повёрнутое на 90° по ч.с.) - это и есть то изображение, которое хранится в массиве.

Создание больших изображений:

В предыдущем примере было создано изображение с высотой в 8 пикселей. Ниже представлен пример создания массива содержащего изображение стрелки с высотой в 16 пикселей:



Б0...Б21 - байты массива (элементы массива).
б0...б7 - биты байтов (биты элементов).

```
const uint8_t myImageInRAM[] =          // Создаём массив myImageInRAM.
{    11, 16,                            // Размер изображения: ш=11px, в=16px.
     B00000000,                         // B0.
     B01100000,                         // B1.
     B01010000,                         // B2.
     B01001000,                         // B3.
     B11000100,                         // B4.
     B00000010,                         // B5.
     B11000100,                         // B6.
     B01001000,                         // B7.
     B01010000,                         // B8.
     B01100000,                         // B9.
     B00000000,                         // B10.

     B00000000,                         // B11.
     B00000000,                         // B12.
     B00000000,                         // B13.
     B00000000,                         // B14.
     B00001111,                         // B15.
     B00001000,                         // B16.
     B00001111,                         // B17.
     B00000000,                         // B18.
     B00000000,                         // B19.
     B00000000,                         // B20.
     B00000000,                         // B21.
};
```

Массив начинается с двух чисел определяющих размер изображения 11,16 (ширина 11 пикселей, высота 16 пикселей). Далее следуют байты, каждый бит которых отвечает за цвет пикселя (0-черный, 1-белый).

Изображение прорисовывается байтами, слева направо, сверху вниз. А каждый байт прорисовывается сверху вниз, от младшего к старшему биту. Сначала побайтно заполняется верх изображения, слева направо (на рисунке байты Б0-Б10), а по достижении правого края

(ширины изображения), начинает прорисовываться следующая часть изображения, расположенная ниже, опять слева направо (на рисунке байты Б11-Б21), и так далее, пока не будет достигнут самый нижний правый пиксель изображения.

Обратите внимание на то, что нижние пикселы картинки со стрелкой (старшие биты б7..б5 старший байтов Б11...Б21) являются пустыми (черными). Это значит что мы можем изменить размер картинки с 11x16, на 11x13, указав новый размер (вместо старого) в начале массива. Теперь картинка будет занимать меньше места на экране.

Вывод фигур:

На дисплей можно выводить такие фигуры как: точка (drawPixel), линия (drawLine), прямоугольник (drawRect) и круг (drawCircle). Функции прорисовки фигур принимают в качестве обязательных параметров, только координаты фигур. Дополнительными параметрами являются: флаг заливки (указывает на необходимость закрасить фигуру) и цвет (фигуры и её заливки). Если дополнительные параметры не указывать, то фигура будет выводиться белым цветом, без заливки.

```
#include <iarduino_OLED.h>                          // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);                         // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                    //
void setup(){                                       //
    myOLED.begin();                                 // Инициируем работу с дисплеем.
}                                                   //
void loop(){                                        //
    myOLED.drawRect   (10, 10,  50, 50, true , 1);  // Прорисовываем квадрат по двум точкам: 10x10, 50x50       , залит
    myOLED.drawRect   (15, 15,  45, 45, true , 0);  // Прорисовываем квадрат по двум точкам: 15x15, 45x45       , залит
    myOLED.drawLine   (10, 10,  50, 50,        0);  // Прорисовываем линию через две точки:  10x10, 50x50       ,
    myOLED.drawLine   (10, 50,  50, 10,        0);  // Прорисовываем линию через две точки:  10x50, 50x10       ,
    myOLED.drawCircle (30, 30,      10, false, 1);  // Прорисовываем круг с центром с точке  30x30 и радиусом 10, залит
    myOLED.drawCircle (30, 30,       5, true , 1);  // Прорисовываем круг с центром с точке  30x30 и радиусом 5 , залит
    myOLED.drawPixel  (30, 30,                 0);  // Прорисовываем точку в координате      30x30,
    myOLED.drawRect   (60, 10, 100, 50, false, 1);  // Прорисовываем квадрат по двум точкам: 60x10, 100x50      , залит
    myOLED.drawRect   (65, 15,  95, 45, true , 1);  // Прорисовываем квадрат по двум точкам: 65x15,  95x45      , залит
    myOLED.drawLine   (60, 10, 100, 50,        1);  // Прорисовываем линию через две точки:  60x10, 100x50      ,
    myOLED.drawLine   (60, 50, 100, 10,        1);  // Прорисовываем линию через две точки:  60x50, 100x10      ,
```

```
    myOLED.drawCircle (80, 30,      10, false, 0);          // Прорисовываем круг с центром с точке  80x30 и радиусом 10, залит
    myOLED.drawCircle (80, 30,       5, true,  0);          // Прорисовываем круг с центром с точке  80x30 и радиусом 10, залит
    myOLED.drawPixel  (80, 30,                 1);          // Прорисовываем точку в координате      80x30,
    delay(2000);                                            // Ждём 2 секунды.
    myOLED.clrScr();                                        // Чистим экран.
    delay(1000);                                            // Ждём 1 секунду.
}                                                           //
```

Загрузив данный пример в Arduino, можно заметить, что для прорисовки всего экрана требуется время. Заметно как фигуры прорисовываются по очереди. Дело в том, что все выше описанные примеры выводили данные на дисплей автоматически (сразу после обращения к функциям библиотеки). Это значит, что общение с дисплеем по шине I2C происходит при каждом обращении к функциям вывода чисел, текста, изображений, фигур, очистки и заливки экрана. Следующий пример выводит те же фигуры, что и данный, но не по очереди, а все сразу, что значительно сокращает время.

Обновление дисплея:

В данном примере показано, как можно управлять передачей данных на дисплей (управлять обновлением дисплея). Если вызвать функцию autoUpdate с параметром false, то последующие обращения к функциям вывода не приведут к изменениям на экране дисплея. Числа, текст, изображения, фигуры, фон и т.д. будут формироваться и храниться в буфере (в ОЗУ Arduino). А передача данных на дисплей (обновление экрана) будет формироваться только после вызова функции update(). Таким образом можно сформировать изображение из множества элементов, а передать его на дисплей за один раз.

```
#include <iarduino_OLED.h>                                  // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);                                 // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3
                                                            //
void setup(){                                               //
    myOLED.begin();                                         // Инициируем работу с дисплеем.
    myOLED.autoUpdate(false);                               // Запрещаем автоматический вывод данных. Информация на дисплее буд
}                                                           //
void loop(){                                                //
    myOLED.drawRect   (10, 10,  50, 50, true , 1);          // Прорисовываем квадрат по двум точкам: 10x10, 50x50       , залит
```

```
    myOLED.drawRect   (15, 15,  45, 45, true , 0);        // Прорисовываем квадрат по двум точкам: 15x15, 45x45       , залит
    myOLED.drawLine   (10, 10,  50, 50,        0);        // Прорисовываем линию через две точки:  10x10, 50x50       ,
    myOLED.drawLine   (10, 50,  50, 10,        0);        // Прорисовываем линию через две точки:  10x50, 50x10       ,
    myOLED.drawCircle (30, 30,      10, false, 1);        // Прорисовываем круг с центром с точке  30x30 и радиусом 10, залит
    myOLED.drawCircle (30, 30,       5, true , 1);        // Прорисовываем круг с центром с точке  30x30 и радиусом 5 , залит
    myOLED.drawPixel  (30, 30,                  0);        // Прорисовываем точку в координате      30x30,
    myOLED.drawRect   (60, 10, 100, 50, false, 1);        // Прорисовываем квадрат по двум точкам: 60x10, 100x50      , залит
    myOLED.drawRect   (65, 15,  95, 45, true , 1);        // Прорисовываем квадрат по двум точкам: 65x15,  95x45      , залит
    myOLED.drawLine   (60, 10, 100, 50,        1);        // Прорисовываем линию через две точки:  60x10, 100x50      ,
    myOLED.drawLine   (60, 50, 100, 10,        1);        // Прорисовываем линию через две точки:  60x50, 100x10      ,
    myOLED.drawCircle (80, 30,      10, false, 0);        // Прорисовываем круг с центром с точке  80x30 и радиусом 10, залит
    myOLED.drawCircle (80, 30,       5, true,  0);        // Прорисовываем круг с центром с точке  80x30 и радиусом 10, залит
    myOLED.drawPixel  (80, 30,                  1);        // Прорисовываем точку в координате      80x30,
    myOLED.update();                                      // Обновляем информацию на дисплее.
    delay(2000);                                          // Ждём 2 секунды.
    myOLED.clrScr();                                      // Чистим экран.
    myOLED.update();                                      // Обновляем информацию на дисплее.
    delay(1000);                                          // Ждём 1 секунду.
}                                                         //
```

Данный пример выводит те же фигуры, что и предыдущий, но значительно быстрее. Скетчи отличаются тем, что в данном примере добавлены функции: autoUpdate с параметром false (6 строка) и update (23 и 26 строки). В любой момент можно вызвать функцию autoUpdate с параметром true и тогда дисплей вновь будет обновляться при каждом обращении к функциям вывода данных.

Описание основных функций библиотек:

Обе библиотеки iarduino_OLED и iarduino_OLED_txt могут использовать как аппаратную, так и программную реализацию шины I2C.

О том как выбрать тип шины I2C рассказано в статье Wiki - расширенные возможности библиотек iarduino для шины I2C.

Подключение библиотек:

Подключение текстовой библиотеки iarduino_OLED_txt:

```
#include <iarduino_OLED_txt.h>  // Подключаем библиотеку iarduino_OLED_txt.
iarduino_OLED_txt myOLED(0x3C); // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3C или 0x3D.
```

Подключение графической библиотеки iarduino_OLED:

```
#include <iarduino_OLED.h>      // Подключаем библиотеку iarduino_OLED.
iarduino_OLED myOLED(0x3C);     // Объявляем объект myOLED, указывая адрес дисплея на шине I2C: 0x3C или 0x3D.
```

Если Вы планируете выводить только текст, то используйте библиотеку iarduino_OLED_txt (она использует минимум ОЗУ).
Если Вы планируете выводить текст, фигуры или изображения, то используйте библиотеку iarduino_OLED.
При объявлении объекта допускается указывать адрес с учетом бита RW=0 (написанный на плате): 0x78 или 0x7A.
Для работы с двумя дисплеями нужно объявить два объекта (с разными именами и адресами).

Функция begin();

- Назначение: Инициализация работы с дисплеем.
- Синтаксис: begin();
- Параметры: нет.
- Возвращаемые значения: нет.
- Примечание:
  - Функцию необходимо вызывать до обращения к остальным функциям и методам объекта.
  - Функцию достаточно вызвать 1 раз в коде setup.
- Пример:

```
myOLED.begin();                 // Инициализация дисплея.
```

Функция clrScr();

- Назначение: Очистка экрана дисплея.
- Синтаксис: clrScr([ЗАЛИТЬ]);
- Параметры:
  - ЗАЛИТЬ - не обязательный параметр (true или false) указывающий что экран нужно залить белым цветом, после его очистки. Значение по умолчанию - false.
- Возвращаемые значения: нет.
- Примечание:
  - Вызов функции без параметра аналогичен вызову с параметром false.
- Пример:

```
myOLED.clrScr();               // Очистить экран дисплея (без заливки).
```

Функция fillScr();

- Назначение: Заливка дисплея.
- Синтаксис: fillScr([ЦВЕТ]);
- Параметры:
  - ЦВЕТ - не обязательный параметр (0 или 1), 1 - залить белым цветом, 0 - залить чёрным цветом. Значение по умолчанию - 1.
- Возвращаемые значения: нет.
- Примечание:
  - Вызов функции без параметра аналогичен вызову с параметром 1 (залить белым цветом).
- Пример:

```
myOLED.fillScr();              // Залить экран дисплея белым цветом.
```

Функция invScr();

- Назначение: Инверсия цветов экрана.
- Синтаксис: invScr([ФЛАГ]);

- Параметры:
  - ФЛАГ - не обязательный параметр (true или false) указывающий что цвета экрана нужно инвертировать.
    Значение по умолчанию - true.
- Возвращаемые значения: нет.
- Примечание:
  - Если флаг установлен, то на экране поменяются цвета как уже выведенных, так и выводимых в дальнейшем данных.
  - Если флаг сброшен, то цвета на экране вернутся к обычному состоянию.
  - Вызов функции без параметра аналогичен вызову с параметром true.
- Пример:

```
myOLED.invScr();                // Инвертировать цвета экрана.
```

Функция invText();

- Назначение: Инвертировать цвет выводимого текста.
- Синтаксис: invText([ФЛАГ]);
- Параметры:
  - ФЛАГ - не обязательный параметр (true или false) указывающий что цвет выводимого текста нужно инвертировать.
    Значение по умолчанию - true.
- Возвращаемые значения: нет.
- Примечание:
  - Если флаг установлен, то весь текст выводимый после обращения к данной функции будет отображаться с инверсией цвета.
  - Если флаг сброшен, то весь текст выводимый после обращения к данной функции будет отображаться без инверсии цвета.
  - Вызов функции без параметра аналогичен вызову с параметром true.
- Пример:

```
myOLED.invText();               // В дальнейшем выводить текст инвертировав его цвет.
```

Функция bgText();

- Назначение: Управление наличием фона у выводимого текста.
- Синтаксис: bgText([ФЛАГ]);
- Параметры:
  - ФЛАГ - не обязательный параметр (true или false) указывающий что у текста есть фон.
    Значение по умолчанию - true.
- Возвращаемые значения: нет.
- Примечание:
  - Если флаг установлен, то весь текст выводимый после обращения к данной функции будет отображаться с фоновым цветом.
  - Если флаг сброшен, то весь текст выводимый после обращения к данной функции будет отображаться без фонового цвета.
  - Вызов функции без параметра аналогичен вызову с параметром true.
  - Отсутствие фона означает, что данные находящиеся под выводимым текстом останутся видны.
  - Функция bgText() доступна только в графической библиотеке iarduino_OLED.
- Пример:

```
myOLED.bgText(false);          // В дальнейшем выводить текст без фона.
```

Функция bgImage();

- Назначение: Управление наличием фона у выводимых изображений.
- Синтаксис: bgImage([ФЛАГ]);
- Параметры:
  - ФЛАГ - не обязательный параметр (true или false) указывающий что у изображений есть фон.
    Значение по умолчанию - true.
- Возвращаемые значения: нет.
- Примечание:
  - Если флаг установлен, то все изображения выводимые после обращения к данной функции будут отображаться с фоновым цветом.
  - Если флаг сброшен, то все изображения выводимые после обращения к данной функции будут отображаться без фонового цвета.
  - Вызов функции без параметра аналогичен вызову с параметром true.
  - Отсутствие фона означает, что данные находящиеся под выводимым изображением останутся видны.

    - Функция bgText() доступна только в графической библиотеке iarduino_OLED.
- Пример:

```
myOLED.bgImage(false);          // В дальнейшем выводить изображения без фона.
```

Функция setFont();

- Назначение: Выбор шрифта для выводимого текста.
- Синтаксис: setFont(ШРИФТ);
- Параметры:
    - ШРИФТ - название массива шрифта, который будет использован для вновь выводимого текста.
- Возвращаемые значения: нет.
- Примечание:
    - Перед выбором шрифта, его нужно подключить (см. пример).
- Список предустановленных шрифтов:
    - **SmallFont** - содержит цифры, символы и Латинские буквы размером 6x8 px.
    - **SmallFontRus** - содержит цифры, символы, Латинские и Кириллические буквы размером 6x8 px.
    - **MediumFont** - содержит цифры, символы и Латинские буквы размером 12x16 px.
    - **MediumFontRus** - содержит цифры, символы, Латинские и Кириллические буквы размером 12x16 px.
    - **MediumNumbers** - содержит цифры размером 12x16 px.
    - **BigNumbers** - содержит цифры размером 14x24 px.
    - **MegaNumbers** - содержит цифры размером 24x40 px.
- Пример:

```
...                             //
extern uint8_t MediumFontRus[]; // Подключаем шрифт MediumFontRus.
...                             //
void setup(){                   //
  myOLED.setFont(MediumFontRus);// Выбираем шрифт для вывода текста.
  myOLED.print("Текст");        // Выводим текст. Текст будет отображаться на дисплее выбранным шрифтом.
```

```
}                              //
```

Функция getFontWidth();

- Назначение: Получение ширины символов шрифта.
- Синтаксис: getFontWidth();
- Параметры: нет.
- Возвращаемые значения: uint8_t - ширина символов выбранного шрифта в пикселях.
- Пример:

```
uint8_t i;                    // Объявляем переменную i.
myOLED.setFont(SmallFontRus); // Выбираем шрифт для вывода текста.
i = myOLED.getFontWidth();    // Получаем ширину символов выбранного шрифта.
```

Функция getFontHeight();

- Назначение: Получение высоты символов шрифта.
- Синтаксис: getFontHeight();
- Параметры: нет.
- Возвращаемые значения: uint8_t - высота символов выбранного шрифта.
- Примечание: высота символов для графической библиотеки iarduino_OLED возвращается в пикселях, а для текстовой библиотеки iarduino_OLED_txt в строках.
- Пример:

```
uint8_t i;                    // Объявляем переменную i.
myOLED.setFont(SmallFontRus); // Выбираем шрифт для вывода текста.
i = myOLED.getFontHeight();   // Получаем высоту символов выбранного шрифта.
```

Функция setCoding();

- Назначение: Указание кодировки текста в скетче.

- Синтаксис: setCoding([КОДИРОВКА]);
- Параметры:
  - КОДИРОВКА - тип кодировки:
    - TXT_CP866 - текст в скетче представлен в кодировке CP-866.
    - TXT_WIN1251 - текст в скетче представлен в кодировке WINDOWS-1251.
    - TXT_UTF8 - текст в скетче представлен в кодировке UTF-8.
    - false - кодировка текста в скетче совпадает с положением символов в массивах шрифтов (CP-866).
- Возвращаемые значения: нет.
- Примечание:
  - Функция полезна, если Вы выводите Русский текст и он некорректно отображается на экране дисплея.
  - В разных версиях Arduino IDE использовались разные кодировки для хранения скетчей, явное указание кодировки позволяет работать в более ранних версиях Arduino IDE.
  - Если функция setCoding не вызывалась, то считается что текст скетча представлен в кодировке UTF-8.
  - Вызов функции с параметром false позволяет выводить символы по их коду (см. примеры).
  - Вызов функции без параметра аналогичен вызову с параметром false.
- Пример:

```
myOLED.setCoding(TXT_UTF8);   // Текст скетча представлен в кодировке UTF-8.
```

Функция setCursor();

- Назначение: Установка курсора в указанную позицию на экране.
- Синтаксис: setCursor(X , Y);
- Параметры:
  - X - координата по оси X в пикселях.
  - Y - координата по оси Y в пикселях или строках (см. примечание).
- Возвращаемые значения: нет.
- Примечание:
  - Координата по оси Y для графической библиотеки iarduino_OLED указывается в пикселях (от 0 до 63), а для текстовой библиотеки

iarduino_OLED_txt в строках (от 0 до 7).
    - После данной функции можно вывести текст или изображение без указания их координат в функциях print() или drawImage().
    - Если требуется задать координату только по одной оси, а другую ось оставить без изменений, то для оси которую требуется оставить без изменений, нужно установить значение OLED_N.
- Пример:

```
myOLED.setCursor(100,7);      // Установить курсор в позицию: 100px:7px для графической библиотеки, или 100px:7стр для текстов
myOLED.setCursor(50,OLED_N);  // Установить курсор по горизонтали в позицию 100px, а по вертикали оставить без изменений.
myOLED.print("Текст");        // Вывести текст на дисплей. Нижняя левая точка текста будет находиться в позиции где находится
```

Функция setCursorShift();

- Назначение: Сдвиг курсора на указанное значение.
- Синтаксис: setCursorShift(X, Y);
- Параметры:
    - X - смещение курсора от его текущей позиции по оси X на указанное количество пикселей.
    - Y - смещение курсора от его текущей позиции по оси Y на указанное количество пикселей или строк (см. примечание).
- Возвращаемые значения: нет.
- Примечание:
    - Смещение по оси Y для графической библиотеки iarduino_OLED указывается в пикселях (от -63 до 63), а для текстовой библиотеки iarduino_OLED_txt в строках (от -7 до 7).
    - Данная функция похожа на функцию setCursor(), но она не указывает координаты курсора, а сдвигает его.
    - Функция может быть полезна для задания точных интервалов между выводимыми текстами и (или) картинками.
- Пример:

```
myOLED.print("Привет");       // Выводим текст "Привет". После вывода текста курсор будет находиться в той же строке, но на сл
myOLED.setCursorShift(5,0);   // Сдвигаем курсор на 5 пикселей вправо по оси X (для сдвига влево, нужно указать отрицательное
myOLED.print("мир");          // Выводим текст "мир". В результате на дисплее будет текст "Привет мир" с шириной отступа между
```

Функция print();

- Назначение: Вывод текста на экран дисплея.
- Синтаксис: print(ТЕКСТ [, X] [, Y]);
- Параметры:
  - ТЕКСТ - текст который требуется вывести на дисплей.
  - X - координата дисплея в которой будет находиться левый край выводимого текста (не обязательный параметр).
    - Значение в пикселях от 0 до 127 (точное указание координаты по оси X).
    - OLED_L - выровнять текст по левому краю экрана.
    - OLED_C - выровнять текст по центру экрана.
    - OLED_R - выровнять текст по правому краю экрана.
    - OLED_N - оставить координату без изменений (см. примечание).
  - Y - координата дисплея в которой будет находиться нижний край выводимого текста (не обязательный параметр).
    - Значение в пикселях от 0 до 63 или строках от 0 до 7 (см. примечание).
    - OLED_T - выровнять текст по верхнему краю экрана.
    - OLED_C - выровнять текст по центру экрана.
    - OLED_B - выровнять текст по нижнему краю экрана.
    - OLED_N - оставить координату без изменений (см. примечание).
- Возвращаемые значения: нет.
- Примечание:
  - Координата по оси Y для графической библиотеки iarduino_OLED указывается в пикселях (от 0 до 63), а для текстовой библиотеки iarduino_OLED_txt в строках (от 0 до 7).
  - Если координаты X и(или) Y не указывать, или указать OLED_N, то координатой будет служить текущая позиция курсора.
  - После вывода текста, позиция курсора будет находиться на той же строке в точке следующей за последним символом текста.
  - Перед использованием функции print() необходимо выбрать шрифт функцией setFont()
  - Функция print() может использоваться для вывода чисел (см. ниже).
- Пример:

```
myOLED.print("Привет", 0, 7); // Выводим текст "Привет", начиная с координаты X=0, Y=7. После вывода текста координата Y остан
```

```
myOLED.print(" мир");          // Выводим текст " мир" (первый символ - пробел), без указания координат. В результате на диспле
```

Функция print();

- Назначение: Вывод чисел на экран дисплея.
- Синтаксис: print(ЧИСЛО [, X] [, Y] [, ПАРАМЕТР]);
- Параметры:
  - ЧИСЛО - положительное, отрицательное, целоё, число или число с плавающей точкой, которое требуется вывести на дисплей.
  - X - координата дисплея в которой будет находиться левый край выводимого числа (не обязательный параметр).
    - Значение в пикселях от 0 до 127 (точное указание координаты по оси X).
    - OLED_L - выровнять число по левому краю экрана.
    - OLED_C - выровнять число по центру экрана.
    - OLED_R - выровнять число по правому краю экрана.
    - OLED_N - оставить координату без изменений (см. примечание).
  - Y - координата дисплея в которой будет находиться нижний край выводимого числа (не обязательный параметр).
    - Значение в пикселях от 0 до 63 или строках от 0 до 7 (см. примечание).
    - OLED_T - выровнять текст по верхнему краю экрана.
    - OLED_C - выровнять текст по центру экрана.
    - OLED_B - выровнять текст по нижнему краю экрана.
    - OLED_N - оставить координату без изменений (см. примечание).
  - ПАРАМЕТР - значение зависит от типа числа.
    - Если выводится число с плавающей точкой, то параметр определяет количество выводимых знаков после запятой (по умолчанию 2).
    - Если выводится целое число, то параметр определяет основание системы счисления выводимого числа (по умолчанию 10).
- Возвращаемые значения: нет.
- Примечание:
  - Координата по оси Y для графической библиотеки iarduino_OLED указывается в пикселях (от 0 до 63), а для текстовой библиотеки iarduino_OLED_txt в строках (от 0 до 7).
  - Если координаты X и(или) Y не указывать, или указать OLED_N, то координатой будет служить текущая позиция курсора.

    - После вывода числа, позиция курсора будет находиться на той же строке в точке следующей за последнеё цифрой числа.
    - Перед использованием функции print() необходимо выбрать шрифт функцией setFont().
    - Функция print() может использоваться для вывода текста (см. выше).
- Пример:

```
myOLED.print(123, 4, 5, 6);   // Выводим число 123 с началом в координате X=4, Y=5, число выводится в шестеричной системе счис
myOLED.print(" мир");         // Выводим текст " мир" (первый символ - пробел), без указания координат. В результате на диспле
```

Функция drawImage();

- Назначение: Вывод изображений на экран дисплея.
- Синтаксис: drawImage(КАРТИНКА [, X] [, Y] [, ТИП_ПАМЯТИ]);
- Параметры:
    - КАРТИНКА - название массива содержащего выводимое изображение.
    - X - координата дисплея в которой будет находиться левый край выводимого изображения (не обязательный параметр).
        - Значение в пикселях от 0 до 127 (точное указание координаты по оси X).
        - OLED_L - выровнять изображение по левому краю экрана.
        - OLED_C - выровнять изображение по центру экрана.
        - OLED_R - выровнять изображение по правому краю экрана.
        - OLED_N - оставить координату без изменений (см. примечание).
    - Y - координата дисплея в которой будет находиться нижний край выводимого изображения (не обязательный параметр).
        - Значение в пикселях (от 0 до 63) (точное указание координаты по оси Y).
        - OLED_T - выровнять изображение по верхнему краю экрана.
        - OLED_C - выровнять изображение по центру экрана.
        - OLED_B - выровнять изображение по нижнему краю экрана.
        - OLED_N - оставить координату без изменений (см. примечание).
    - ТИП_ПАМЯТИ - не обязательный параметр определяющий тип памяти в которой находится массив с изображением.
        - IMG_RAM - массив с изображением находится в области ОЗУ (оперативная память).
        - IMG_ROM - массив с изображением находится в области ПЗУ (память программ).

- Возвращаемые значения: нет.
- Примечание:
  - Если координаты X и Y не указывать, или указать OLED_N, то координатой будет служить текущая позиция курсора.
  - Если тип памяти не указывать, то функция будет выводить массив изображения из памяти программ (IMG_ROM).
  - После вывода изображения, позиция курсора будет находиться в точке справа от нижнего правого пикселя изображения.
  - Перед выводом предустановленных изображений, нужно подключить требуемый массив (см. пример).
  - Функция drawImage() доступна только в графической библиотеке iarduino_OLED.
- Список предустановленных изображений и их названий:

- Img_Battery_charging	- Img_Arrow_down	- Img_Level_1	- Img_Alarm
- Img_Battery_low	- Img_Arrow_left	- Img_Level_2	- Img_Antenna
- Img_Battery_0	- Img_Arrow_right	- Img_Level_3	- Img_Bluetooth
- Img_Battery_1	- Img_Arrow_up	- Img_Level_4	- Img_Message
- Img_Battery_2	- Img_Dynamic	- Img_Netlevel_1	- Img_Light
- Img_Battery_3	- Img_Dynamic_off	- Img_Netlevel_2	- Img_Melody
- Img_Call	- Img_Dynamic_on	- Img_Netlevel_3	- Img_Check
- Img_Call_in	- Img_Checkbox_off	- Img_Radio_off	- Img_Settings
- Img_Call_out	- Img_Checkbox_on	- Img_Radio_on	
- Img_BigBattery_low	- Img_Logo		

- Пример:

```
extern uint8_t Img_Logo[];                                          // Подключаем массив предустановленного изображения Img_Log
uint8_t i[] = {9,8, 0x3C,0x42,0x91,0xA5,0xA1,0xA5,0x91,0x42,0x3C}; // Определяем массив с изображением 9x8 в области ОЗУ.
myOLED.drawImage(i, 59, 36, IMG_RAM);                               // Выводим изображение из массива i. Координата нижней лево
```

```
myOLED.drawImage(Img_Logo, OLED_N, OLED_N, IMG_ROM);                // Выводим изображение из массива Img_Logo справа от предыд
```

Функция getImageWidth();

- Назначение: Получение ширины изображения.
- Синтаксис: getImageWidth(КАРТИНКА [, ТИП_ПАМЯТИ]);
- Параметры:
  - КАРТИНКА - название массива с изображением, ширину которого требуется узнать.
  - ТИП_ПАМЯТИ - не обязательный параметр определяющий тип памяти в которой находится массив с изображением.
    - IMG_RAM - массив с изображением находится в области ОЗУ (оперативная память).
    - IMG_ROM - массив с изображением находится в области ПЗУ (память программ).
- Возвращаемое значение: (uint8_t) ширина изображения в пикселях.
- Примечание:
  - Если тип памяти не указывать, то функция будет обращаться к массиву из памяти программ (IMG_ROM).
  - Функция getImageWidth() доступна только в графической библиотеке iarduino_OLED.
- Пример:

```
uint8_t w = myOLED.getImageWidth(Img_Logo); // получить ширину изображения хранящегося в массиве Img_Logo.
```

Функция getImageHeight();

- Назначение: Получение высоты изображения.
- Синтаксис: getImageHeight(КАРТИНКА [, ТИП_ПАМЯТИ]);
- Параметры:
  - КАРТИНКА - название массива с изображением, высоту которого требуется узнать.
  - ТИП_ПАМЯТИ - не обязательный параметр определяющий тип памяти в которой находится массив с изображением.
    - IMG_RAM - массив с изображением находится в области ОЗУ (оперативная память).
    - IMG_ROM - массив с изображением находится в области ПЗУ (память программ).
- Возвращаемое значение: (uint8_t) высота изображения в пикселях.

- Примечание:
  - Если тип памяти не указывать, то функция будет обращаться к массиву из памяти программ (IMG_ROM).
  - Функция getImageHeight() доступна только в графической библиотеке iarduino_OLED.
- Пример:

```
uint8_t h = myOLED.getImageHeight(Img_Logo); // получить высоту изображения хранящегося в массиве Img_Logo.
```

Функция drawPixel();

- Назначение: Вывод точки на экран дисплея.
- Синтаксис: drawPixel(X , Y [, ЦВЕТ]);
- Параметры:
  - X - координата дисплея в пикселях по оси X (от 0 до 127) для вывода точки.
  - Y - координата дисплея в пикселях по оси Y (от 0 до 63) для вывода точки.
  - ЦВЕТ - не обязательный параметр (0 или 1), 1 - белый, 0 - чёрный.
- Возвращаемые значения: нет.
- Примечание:
  - Если цвет не указан, то точка будет белой.
  - После вывода точки позиция курсора будет находиться в указанной координате X,Y.
  - Функция drawPixel() доступна только в графической библиотеке iarduino_OLED.
- Пример:

```
myOLED.drawPixel(10, 20);     // Вывести белую точку в координате X=10, Y=20.
```

Функция getPixel();

- Назначение: Получение цвета точки на экране дисплея.
- Синтаксис: getPixel(X , Y);
- Параметры:
  - X - координата дисплея в пикселях по оси X (от 0 до 127).

    - Y - координата дисплея в пикселях по оси Y (от 0 до 63).
- Возвращаемые значения: (bool) цвет точки по указанным координатам (0-чёрный, 1-белый).
- Примечание: Функция getPixel() доступна только в графической библиотеке iarduino_OLED.
- Пример:

```
myOLED.getPixel(10, 20);      // Получить цвет точки в координате X=10, Y=20.
```

Функция drawLine();

- Назначение: Вывод линии на экран дисплея.
- Синтаксис: drawLine(X1,Y1 , X2,Y2 [, ЦВЕТ]);
- Параметры:
    - X1, Y1 - координата первой точки для вывода линии.
    - X2, Y2 - координата второй точки для вывода линии.
    - ЦВЕТ - не обязательный параметр (0 или 1), 1 - белый, 0 - чёрный.
- Возвращаемые значения: нет.
- Примечание:
    - Если цвет не указан, то линия будет белой.
    - Допускается не указывать координаты первой точки X1,Y1, тогда координатой будет служить текущая позиция курсора.
    - После вывода линии позиция курсора будет находиться в координате X2,Y2.
    - Функция drawLine() доступна только в графической библиотеке iarduino_OLED
- Пример:

```
myOLED.drawLine(10,20,30,40); // Вывести линию между точками X=10,Y=20 и X=30,Y=40.
```

Функция drawRect();

- Назначение: Вывод прямоугольника на экран дисплея.
- Синтаксис: drawRect(X1,Y1 , X2,Y2 [, ЗАЛИТЬ] [, ЦВЕТ]);
- Параметры:

    - X1, Y1 - координата первой точки для вывода прямоугольника.
    - X2, Y2 - координата второй точки для вывода прямоугольника.
    - ЗАЛИТЬ - не обязательный параметр (true или false) указывающий что прямоугольник нужно залить указанным цветом.
    - ЦВЕТ - не обязательный параметр (0 или 1), 1 - белый, 0 - чёрный.
- Возвращаемые значения: нет.
- Примечание:
    - Если цвет не указан, то прямоугольник будет белый.
    - Если не указывать флаг заливки, то прямоугольник останется не залитым.
    - После вывода прямоугольника позиция курсора будет находиться в координате X2,Y2.
    - Функция drawRect() доступна только в графической библиотеке iarduino_OLED
- Пример:

```
myOLED.drawRect(10,20,30,40); // Вывести прямоугольник между точками X=10,Y=20 и X=30,Y=40. Прямоугольник будет белым и без за
```

Функция drawCircle();

- Назначение: Вывод круга на экран дисплея.
- Синтаксис: drawCircle(X , Y , РАДИУС [, ЗАЛИТЬ] [, ЦВЕТ]);
- Параметры:
    - X, Y - координата центра круга в пикселях.
    - РАДИУС - длинна радиуса круга в пикселях.
    - ЗАЛИТЬ - не обязательный параметр (true или false) указывающий что круг нужно залить указанным цветом.
    - ЦВЕТ - не обязательный параметр (0 или 1), 1 - белый, 0 - чёрный.
- Возвращаемые значения: нет.
- Примечание:
    - Если цвет не указан, то круг будет белый.
    - Если не указывать флаг заливки, то круг останется не залитым.
    - После вывода круга позиция курсора будет находиться в указанной координате X,Y.
    - Функция drawCircle() доступна только в графической библиотеке iarduino_OLED

- Пример:

```
myOLED.drawCircle(30,20,10);  // Вывести круг с центром в точке X=30,Y=20 и радиусом 10 пикселей. Круг будет белым и без залив
```

Функция autoUpdate();

- Назначение: Управление обновлением данных на дисплее.
- Синтаксис: autoUpdate(ФЛАГ);
- Параметры:
  - ФЛАГ - (true или false) разрешает обновление данных на экране дисплея после каждого обращения к функциям вывода.
- Возвращаемые значения: нет.
- Примечание:
  - Если флаг установлен, то данные на дисплее будут обновляться после каждого вызова функций вывода данных (по умолчанию).
  - Если флаг сброшен, то обращение к функциям вывода данных не приведёт к изменению на экране дисплея. Все изменения будут храниться в буфере (в ОЗУ Arduino). Данные из буфера будут отправляться на дисплей вызовом функции update().
  - Функция autoUpdate() доступна только в графической библиотеке iarduino_OLED
- Пример:

```
myOLED.autoUpdate(false);     // Запрещаем автоматическое обновление данных на дисплее.
myOLED.drawCircle(30,20,10);  // Выводим круг. Круг сохранится в буфере, но не будет прорисован на дисплее.
myOLED.drawRect(20,10,40,30); // Выводим прямоугольник. Прямоугольник сохранится в буфере, но не будет прорисован на дисплее.
myOLED.print("iArduino.ru");  // Выводим текст. Текст сохранится в буфере, но не будет прорисован на дисплее.
myOLED.update();              // Отправляем данные из буфера на дисплей. На дисплее одновременно прорисуются и круг, и прямоуг
```

Функция update();

- Назначение: Передача данных из буфера на дисплей.
- Синтаксис: update();
- Параметры: нет.

- Возвращаемые значения: нет.
- Примечание:
  - Функция актуальна только после обращения к функции autoUpdate с параметром false (см. выше).
  - Функция update() доступна только в графической библиотеке iarduino_OLED
- Пример:

```
myOLED.autoUpdate(false);      // Запрещаем автоматическое обновление данных на дисплее.
myOLED.drawCircle(30,20,10);   // Выводим круг. Круг сохранится в буфере, но не будет прорисован на дисплее.
myOLED.drawRect(20,10,40,30); // Выводим прямоугольник. Прямоугольник сохранится в буфере, но не будет прорисован на дисплее.
myOLED.print("iArduino.ru");   // Выводим текст. Текст сохранится в буфере, но не будет прорисован на дисплее.
myOLED.update();               // Отправляем данные из буфера на дисплей. На дисплее одновременно прорисуются и круг, и прямоуг
```

Переменная numX

- Значение: Содержит координату курсора по оси X в пикселях (от 0 до 127).
- Назначение: Можно читать и указывать положительное целое число не прибегая к функциям библиотек.

Переменная numY

- Значение: Содержит координату курсора по оси Y. для графической библиотеки iarduino_OLED - в пикселях (от 0 до 63), а для текстовой библиотеки iarduino_OLED_txt - в строках (от 0 до 7).
- Назначение: Можно читать и указывать положительное целое число не прибегая к функциям библиотек.

Отличия библиотек:

Библиотеки iarduino_OLED и iarduino_OLED_txt схожи по назначению и синтаксису, но имеют следующие отличия:

- Библиотека iarduino_OLED_txt использует минимум памяти ОЗУ и предназначена только для вывода текста и чисел, а значит не поддерживает функции: drawImage, drawPixel, getPixel, drawLine, drawRect, drawCircle, bgText, bgImage, autoUpdate, update.
- Библиотека iarduino_OLED поддерживает все описанные выше функции, но занимает не менее 1 КБ памяти ОЗУ.
- Координаты по оси X (горизонтально) для обеих библиотек указываются в пикселях (от 0 до 127), а координаты оси Y (вертикально)

указываются:

- для текстовой библиотеки iarduino_OLED_txt в строках (от 0 до 7);
- для графической библиотеки iarduino_OLED в пикселях (от 0 до 63).

Применение:

- Проекты с выводом информации на дисплей;