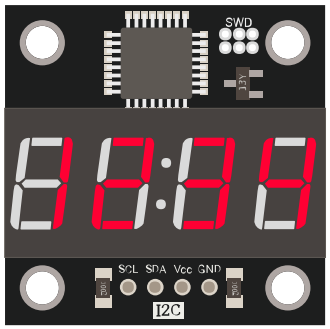
Четырехразрядный индикатор LED, FLASH-I2C



Общие сведения:

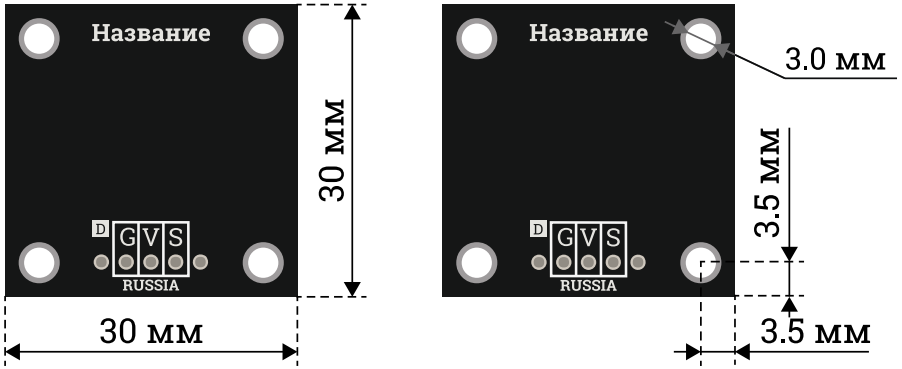
Trema модуль - Четырехразрядный индикатор LED, FLASH-I2C - является устройством вывода информации получаемой по шине I2C.

Модуль относится к серии «Flash», а значит к одной шине I2C можно подключить более 100 модулей, так как их адрес на шине I2C (по умолчанию 0x09), хранящийся в энергонезависимой памяти, можно менять программно.

Модуль является устройством вывода информации, его можно использовать для отображения чисел (в том числе и отрицательных), времени, температуры и некоторых символов.

Спецификация:

- Напряжение питания: 3,3 В или 5 В, поддерживаются оба напряжения.
- Ток потребляемый модулем: до 10 мА.
- Интерфейс: I2C.
- Скорость шины I2C: 100 кбит/с.
- Адрес на шине I2C: устанавливается программно (по умолчанию 0x09).
- Уровень логической 1 на линиях шины I2C: Vcc.
- Количество разрядов индикатора: 4.
- Частота обновления выводимых данных: от 1 до 255 Гц.
- Рабочая температура: от -20 до +70 °C.
- Габариты: 30 x 30 мм.
- Вес: 15 г.



Подключение:

Перед подключением модуля ознакомьтесь с разделом "Смена адреса модуля на шине I2C" в данной статье.

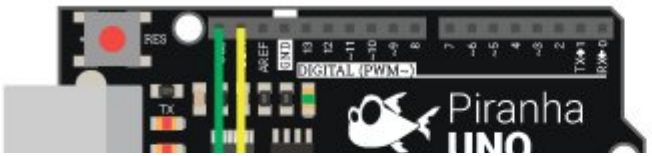
На плате модуля расположен разъем из 4 выводов для подключения к шине I2C.

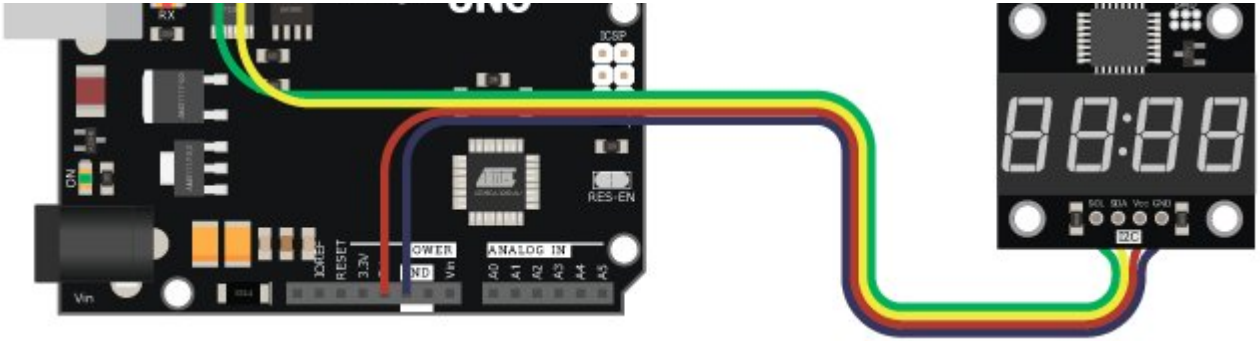
- **SCL** - вход/выход линии тактирования шины I2C.
- **SDA** - вход/выход линии данных шины I2C.
- **Vcc** - вход питания 3,3 или 5 В.
- **GND** - общий вывод питания.

Модуль удобно подключать 3 способами, в зависимости от ситуации:

Способ - 1 : Используя проводной шлейф и Piranha UNO

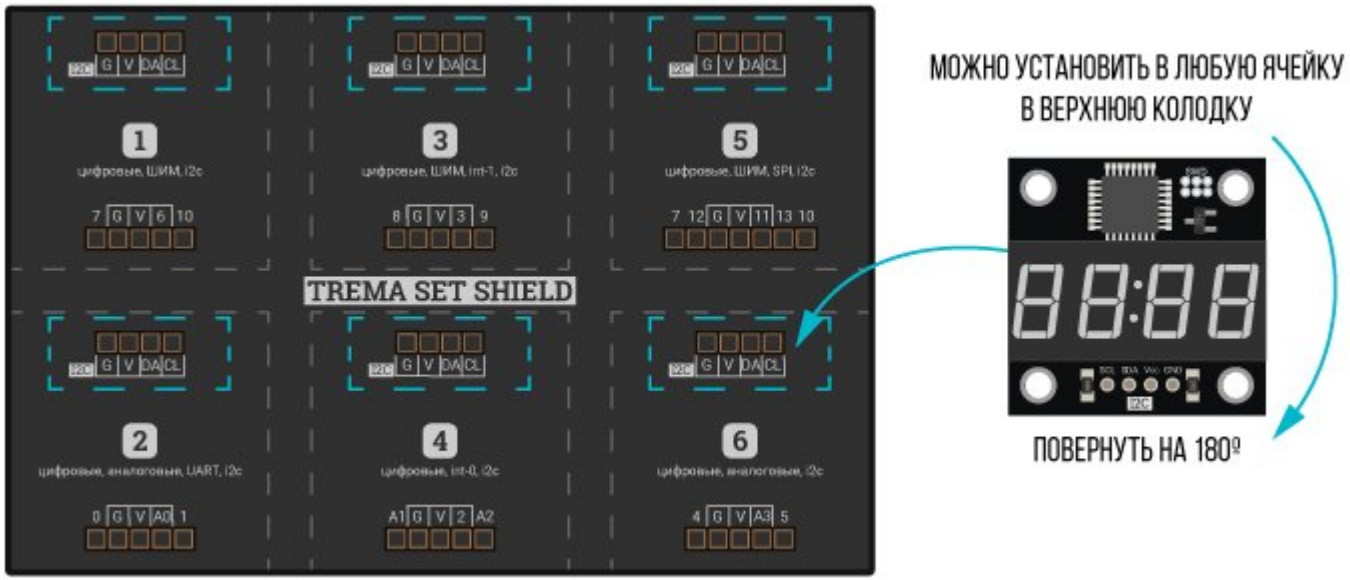
Используя провода «Папа — Мама», подключаем напрямую к контроллеру Piranha UNO.





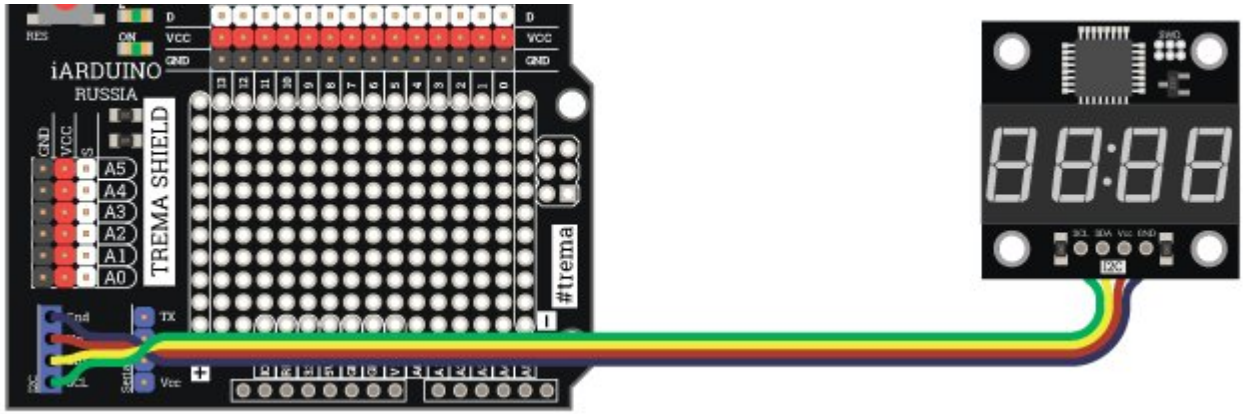
Способ - 2 : Используя Trema Set Shield

Модуль можно подключить в любую верхнюю колодку Trema Set Shield.



Способ - 3 : Используя проводной шлейф и Shield

Используя 3-х проводной шлейф, к Trema Shield, Trema-Power Shield, Motor Shield, Trema Shield NANO и тд.



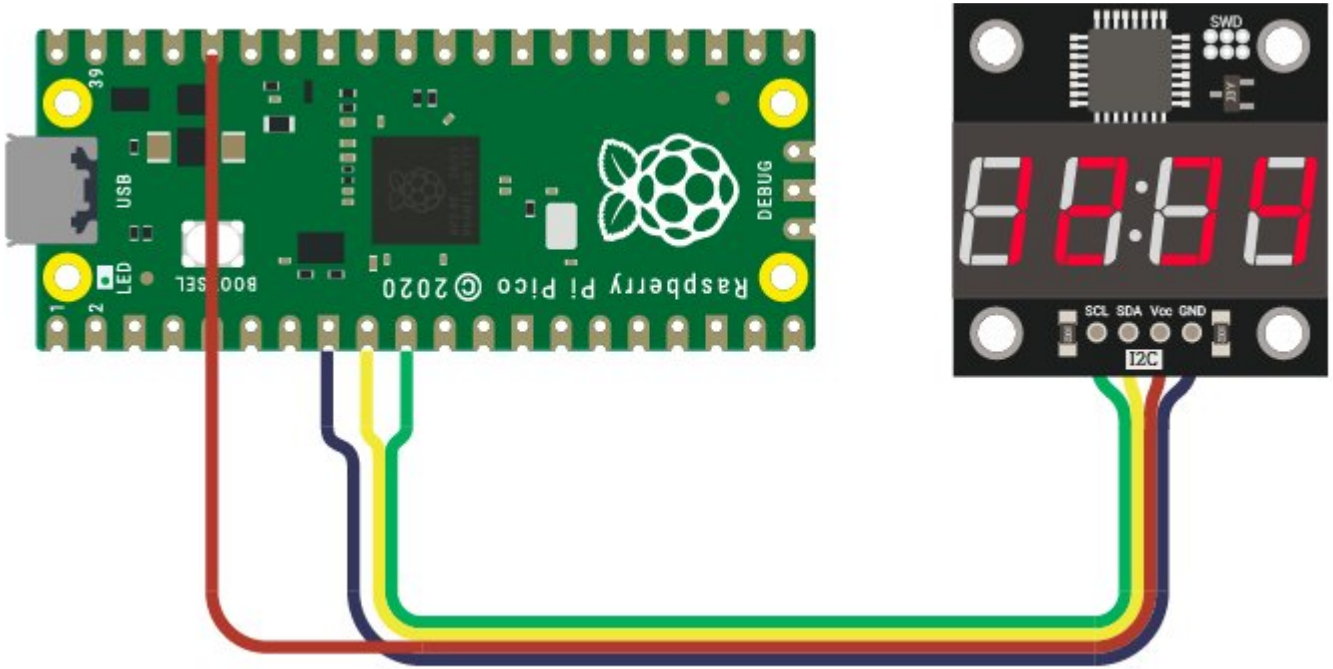
Подключение к Raspberry Pi Pico

Для работы с Pi Pico необходимо установить поддержку платы в Arduino IDE. Нажмите здесь для перехода на подробную статью о том, как это сделать.

В этом случае логическую часть модуля необходимо питать от 3-х вольт Pi Pico

Для работы с Pi Pico версия установленной библиотеки должна быть 1.0.1 и выше

Модуль	**Pi Pico**
SDA	GP6
SCL	GP7
Vcc	3V3(OUT)
GND	GND



Питание:

Входное напряжение питания модуля 3,3В или 5В постоянного тока (поддерживаются оба напряжения питания), подаётся на выводы Vcc и GND.

Подробнее о модуле:

Модуль построен на базе четырёхразрядного светодиодного индикатора, микроконтроллера STM32F030K6 и снабжён собственным стабилизатором напряжения. Модуль способен выводить числа, время, температуру и некоторые символы.

Модуль позволяет:

- Менять свой адрес на шине I2C.
- Управлять внутренней подтяжкой линий шины I2C (по умолчанию включена).
- Выводить числа, время, температуру и некоторые символы.

- Выравнивать выводимые числа по любой из 4 позиций.
- Определять направление сдвига числа от позиции по которой выполнено выравнивание.
- Задавать частоту обновления информации дисплея, от 1 до 255 Гц.
- Указывать яркость свечения всех сегментов дисплея, от 0 до 7.
- Поворачивать изображение индикатора на 180°.
- Указывать любому разряду индикатора мигать включёнными сегментами с заданной частотой.

Специально для работы с Trema модулем - Четырехразрядный индикатор LED, FLASH-I2C , нами разработана библиотека iarduino_I2C_4LED которая позволяет реализовать все функции модуля.

Подробнее про установку библиотеки читайте в нашей инструкции.

Смена адреса модуля на шине I2C:

По умолчанию все модули FLASH-I2C имеют установленный адрес 0x09. Если вы планируете подключать более 1 модуля на шину I2C, необходимо изменить адреса модулей таким образом, чтобы каждый из них был уникальным. Более подробно о том, как изменить адрес, а также о многом другом, что касается работы FLASH-I2C модулей, вы можете прочесть в **этой статье**.

В первой строке скетча необходимо записать в переменную **newAddress** адрес, который будет присвоен модулю. После этого подключите модуль к контроллеру и загрузите скетч. **Адрес может быть от 0x07 до 0x7F**.

```
uint8_t newAddress = 0x09;                                  // Назначаемый модулю адрес (0x07 < адрес < 0x7F).
                                                            //
#include <Wire.h>                                           // Подключаем библиотеку для работы с аппаратной шиной I2C.
#include <iarduino_I2C_4LED.h>                              // Подключаем библиотеку для работы с индикатором I2C-flash.
iarduino_I2C_4LED dispLED;                                  // Объявляем объект dispLED для работы с функциями и методами библис
                                                            // Если при объявлении объекта указать адрес, например, dispLED(0xBE
void setup(){                                               //
     Serial.begin(9600);                                    //
     if( dispLED.begin() ){                                 // Инициируем работу с мотором.
```

```
        Serial.print("Найден LED индикатор 0x");          //
        Serial.println( dispLED.getAddress(), HEX );      // Выводим текущий адрес модуля.
        if( dispLED.changeAddress(newAddress) ){          // Меняем адрес модуля на newAddress.
            Serial.print("Адрес изменён на 0x");          //
            Serial.println(dispLED.getAddress(),HEX );    // Выводим текущий адрес модуля.
        }else{                                            //
            Serial.println("Адрес не изменён!");          //
        }                                                 //
    }else{                                                //
        Serial.println("LED индикатор не найден!");       //
    }                                                     //
}                                                         //
                                                          //
void loop(){                                              //
}                                                         //
```

Примеры:

В данном разделе раскрыты примеры работы с модулем по шине I2C с использованием библиотеки iarduino_I2C_4LED. Сама библиотека содержит больше примеров, доступных из меню Arduino IDE: Файл / Примеры / iarduino I2C 4LED (индикатор).

Вывод времени прошедшего с момента старта скетча:

Пример выводит время в минутах и секундах.

```
#include <Wire.h>                                         // Подключаем библиотеку для работы с аппаратной шиной I2C.
#include <iarduino_I2C_4LED.h>                            // Подключаем библиотеку для работы с индикатором I2C-flash.
iarduino_I2C_4LED dispLED(0x09);                          // Объявляем объект dispLED для работы с функциями и методами библис
                                                          // Если объявить объект без указания адреса (iarduino_I2C_4LED displ
int i, j;                                                 // Объявляем переменные для хранения числа минут и секунд прошедших
                                                          //
void setup(){                                             //
```

```
     dispLED.begin();                                    // Инициируем работу с индикатором.
     dispLED.blink(0,true);                              // Указываем двоеточию мигать (если оно выводится на индикатор).
}                                                        //
                                                         //
void loop(){                                             //
     i = millis()/1000 / 60 % 60;                        // Получаем минуты  прошедшие с момента старта скетча.
     j = millis()/1000      % 60;                        // Получаем секунды прошедшие с момента старта скетча.
     dispLED.print(i, j, TIME);                          // Выводим значения i и j как время (через двоеточие).
}                                                        //
```

После загрузки данного примера, на дисплее начнут высвечиваться минуты и секунды прошедшие с момента старта скетча.

Описание функций библиотеки:

В данном разделе описаны функции библиотеки iarduino_I2C_4LED для работы с Trema модулем - Четырехразрядный индикатор LED, FLASH-I2C.

Функции данной библиотеки полностью совместимы с библиотекой iarduino_4LED используемой для управления Trema четырёхразрядным LED индикатором который подключается к Arduino без использования шины I2C.

Библиотека iarduino_I2C_4LED может 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 библиотеки:

- Если адрес модуля известен (в примере используется адрес 0x09):

```
#include <iarduino_I2C_4LED.h>   // Подключаем библиотеку для работы с модулем.
iarduino_I2C_4LED dispLED(0x09); // Создаём объект dispLED для работы с функциями и методами библиотеки iarduino_I2C_4LED, ука
```

- Если адрес модуля неизвестен (адрес будет найден автоматически):

```
#include <iarduino_I2C_4LED.h>   // Подключаем библиотеку для работы с модулем.
iarduino_I2C_4LED dispLED;       // Создаём объект dispLED для работы с функциями и методами библиотеки iarduino_I2C_4LED, без
```

- При создании объекта без указания адреса, на шине должен находиться только один модуль.

Функция begin();

- Назначение: Инициализация работы с модулем.
- Синтаксис: begin();
- Параметры: Нет.
- Возвращаемое значение: bool - результат инициализации (true или false).
- Примечание: По результату инициализации можно определить наличие модуля на шине.
- Пример:

```
if( dispLED.begin() ){ Serial.print( "Модуль найден и инициирован!" ); }
else                 { Serial.print( "Модуль не найден на шине I2C" ); }
```

Функция reset();

- Назначение: Перезагрузка модуля.
- Синтаксис: reset();
- Параметры: Нет.
- Возвращаемое значение: bool - результат перезагрузки (true или false).
- Пример:

```
if( dispLED.reset() ){ Serial.print( "Модуль    перезагружен" ); }
else                 { Serial.print( "Модуль не перезагружен" ); }
```

Функция changeAddress();

- Назначение: Смена адреса модуля на шине I2C.
- Синтаксис: changeAddress(АДРЕС);
- Параметр:
  - uint8_t АДРЕС - новый адрес модуля на шине I2C (целое число от 0x08 до 0x7E)
- Возвращаемое значение: bool - результат смены адреса (true или false).
- Примечание:
  - Адрес модуля сохраняется в энергонезависимую память, а значит будет действовать и после отключения питания.
  - Текущий адрес модуля можно узнать функцией getAddress().
- Пример:

```
if( dispLED.changeAddress(0x12) ){ Serial.print( "Адрес модуля изменён на 0x12" ); }
else                              { Serial.print( "Не удалось изменить адрес"    ); }
```

Функция getAddress();

- Назначение: Запрос текущего адреса модуля на шине I2C.
- Синтаксис: getAddress();
- Параметры: Нет.
- Возвращаемое значение: uint8_t АДРЕС - текущий адрес модуля на шине I2C (от 0x08 до 0x7E)
- Примечание: Функция может понадобиться если адрес модуля не указан при создании объекта, а обнаружен библиотекой.
- Пример:

```
Serial.print( "Адрес модуля на шине I2C = 0x" );
Serial.println( dispLED.getAddress(), HEX );
```

Функция getVersion();

- Назначение: Запрос версии прошивки модуля.
- Синтаксис: getVersion();
- Параметры: Нет

- Возвращаемое значение: uint8_t ВЕРСИЯ - номер версии прошивки от 0 до 255.
- Пример:

```
Serial.print( "Версия прошивки модуля " );
Serial.println( dispLED.getVersion() );
```

Функция setPullI2C();

- Назначение: Управление внутрисхемной подтяжкой линий шины I2C.
- Синтаксис: setPullI2C([ФЛАГ]);
- Параметр:
  - bool ФЛАГ требующий установить внутрисхемную подтяжку линий шины I2C (true или false).
- Возвращаемое значение:
  - bool - результат включения / отключения внутрисхемной подтяжки (true или false).
- Примечание:
  - Вызов функции без параметра равносилен вызову функции с параметром true - установить.
  - Флаг установки внутрисхемной подтяжки сохраняется в энергонезависимую память модуля, а значит будет действовать и после отключения питания.
  - Внутрисхемная подтяжка линий шины I2C осуществляется до уровня 3,3 В, но допускает устанавливать внешние подтягивающие резисторы и иные модули с подтяжкой до уровня 3,3 В или 5 В, вне зависимости от состояния внутрисхемной подтяжки модуля.
- Пример:

```
if( dispLED.setPullI2C(true ) ){ Serial.print( "Внутрисхемная подтяжка установлена." ); }
if( dispLED.setPullI2C(false) ){ Serial.print( "Внутрисхемная подтяжка отключена."   ); }
```

Функция getPullI2C();

- Назначение: Запрос состояния внутрисхемной подтяжки линий шины I2C.
- Синтаксис: getPullI2C();
- Параметры: Нет.

- Возвращаемое значение: bool - ФЛАГ включения внутрисхемной подтяжки (true или false).
- Пример:

```
if( dispLED.getPullI2C() ){ Serial.print( "Внутрисхемная подтяжка включена."  ); }
else                      { Serial.print( "Внутрисхемная подтяжка отключена." ); }
```

Функция clear();

- Назначение: очистка индикатора
- Синтаксис: clear();
- Параметры: Нет.
- Возвращаемые значения: Нет.
- Пример:

```
dispLED.clear(); // Чистим LED индикатор (все диоды выключатся).
```

Функция light();

- Назначение: Установка яркости свечения индикатора.
- Синтаксис: light(ЧИСЛО);
- Параметры:
  - uint8_t ЧИСЛО - целое число, определяющее яркость, от 0 до 7 (максимальная яркость).
- Возвращаемые значения: Нет.
- Примечание: Яркость по умолчанию = 6.
- Пример:

```
dispLED.light(7); // Устанавливаем максимальную яркость свечения LED индикатора.
```

Функция point();

- Назначение: Установка точек.

- Синтаксис: point(ПОЗИЦИЯ , СОСТОЯНИЕ);
- Параметры:
  - uint8_t ПОЗИЦИЯ - целое число, указывающее позицию точки: 1, 2, 3, 4 или 0 для двоеточия.
  - bool СОСТОЯНИЕ - флаг может принимать значение 0 (выключить) или 1 (включить).
- Возвращаемые значения: Нет.
- Примечание:
  - Если первый параметр ПОЗИЦИЯ больше 4, то СОСТОЯНИЕ применяется ко всем точкам и двоеточию индикатора.
  - Функция управляет только точками и не влияет на ранее установленные цифры.
- Пример:

```
displED.point(0, true);  // Включаем двоеточие     (светится только двоеточие).
displED.point(1, true);  // Включаем первую точку  (светится и двоеточие и самая левая точка).
displED.point(0, false); // Выключаем двоеточие    (светится только самая левая точка).
```

Функция blink();

- Назначение: Управление миганием цифр и символов в указанных позициях.
- Синтаксис: blink(ПОЗИЦИЯ , СОСТОЯНИЕ);
- Параметры:
  - uint8_t ПОЗИЦИЯ - целое число, указывающее позицию цифры: 1, 2, 3, 4 или 0 для двоеточия.
  - bool СОСТОЯНИЕ - флаг может принимать значение 0 (не мигать) или 1 (мигать).
- Возвращаемые значения: Нет.
- Примечание:
  - Если первый параметр ПОЗИЦИЯ больше 4, то СОСТОЯНИЕ применяется ко всему индикатору.
  - Функция управляет только миганием и не влияет на ранее установленные цифры, символы и точки.
  - Частота миганий задаётся функцией `frequ()` .
- Пример:

```
displED.blink(0, true);  // Заставляем мигать двоеточие (если двоеточие включено).
displED.blink(1, true);  // Заставляем мигать первую цифру или символ, включая её точку (теперь будет мигать и двоеточие, и си
```

```
dispLED.blink(0, false); // Отключаем мигание двоеточия (теперь будет мигать только цифра или символ находящаяся в первой пози
```

Функция frequ();

- Назначение: Установка частоты миганий цифр и символов указанных функцией `blink();` .
- Синтаксис: frequ(ЧАСТОТА);
- Параметр:
  - uint8_t ЧАСТОТА - целое число, указывающее частоту в Гц, от 1 до 4.
- Возвращаемые значения: Нет.
- Примечание:
  - Частота миганий цифр и символов позиция которых указана функцией `blink();` не зависит от частоты обновления дисплея указанной функцией `fps();` .
  - Частота по умолчанию = 1 Гц.
- Пример:

```
dispLED.frequ(1);  // Цифры и символы индикатора определённые функцией blink() будут мигать с частотой 1 Гц.
```

Функция turn();

- Назначение: Разворот изображения индикатора.
- Синтаксис: turn(ФЛАГ);
- Параметр:
  - bool ФЛАГ - может принимать значение 0 (без разворота) или 1 (повернуть на 180°).
- Возвращаемые значения: Нет.
- Примечание:
  - Если на индикаторе есть точки и они физически расположены только снизу цифр, то при развороте изображения, точки окажутся вверху и перед цифрами.
- Пример:

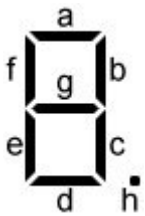
```

```

```
dispLED.turn(1);  // Развернуть изображение индикатора на 180°.
```

Функция setLED();

- Назначение: Установка светодиодов (сегментов) индикатора по битам.
- Синтаксис: setLED ([[[[[БАЙТ_№1] , БАЙТ_№2] , БАЙТ_№3] , БАЙТ_№4] , ФЛАГ]);
- Параметры:
  - БАЙТ_№1 - каждый бит этого байта включает свой светодиод (сегмент) в 1 позиции.
  - БАЙТ_№2 - каждый бит этого байта включает свой светодиод (сегмент) в 2 позиции.
  - БАЙТ_№3 - каждый бит этого байта включает свой светодиод (сегмент) в 3 позиции.
  - БАЙТ_№4 - каждый бит этого байта включает свой светодиод (сегмент) в 4 позиции.
  - bool ФЛАГ - включает двоеточие.



- Возвращаемые значения: Нет.
- Примечание:
  - Каждый бит байта соответствует одному светодиоду (сегменту) в следующем порядке: **hgfedcba** (старший бит управляет сегментом «h», а младший бит управляет сегментом «a»).
  - Все параметры являются необязательными, отсутствие параметра означает что все светодиоды (сегменты) в данной позиции будут выключены.
- Пример:

```
dispLED.setLED(B00000110, B01100011);  // В 1 позиции включены сегменты «c» и «b» (как у цифры 1), а во 2 разряде включены сег
```

Функция print();

- Назначение: Вывод числа, массива чисел, текста, времени или температуры.
- Синтаксис: print (ЗНАЧЕНИЕ , ПАРАМЕТРЫ_ВЫВОДА_ЧИСЛА);
- Параметры:
  - ЗНАЧЕНИЕ - то, что требуется вывести на индикатор, это может быть:
    - Целое число (как положительное, так и отрицательное).

        - Дробное число (как положительное, так и отрицательное).
        - Массив (из 4 положительных целых чисел, от 0 до 9)
        - Текст (из следующих символов "0123456789 abcdefghijlnopstu .,:;*-_")
    - ПАРАМЕТРЫ_ВЫВОДА_ЧИСЛА - допускается указывать от 0 до 5 параметров:
        - LEN1, LEN2, LEN3 или LEN4 - количество символов в выводимом числе (включая знак минус).
        - POS1, POS2, POS3 или POS4 - позиция от левого края, к которой привязывается выводимое число.
        - LEFT или RIGHT - направление вывода относительно указанной позиции.
        - DEC или HEX - система счисления для выводимого числа.
        - TIME или TEMP - отображение чисел в виде времени или числа в виде температуры.
        - символ типа char - для заполнения им недостающих разрядов
- Возвращаемые значения: Нет.
- Примечание:
    - Параметры указываются только если значением является число (целое или с плавающей точкой).
    - Если первым параметром (после значения) указать число, то оно будет означать количество выводимых разрядов после запятой.
- Примеры:

```
dispLED.print( 12345.6789 , 2 , POS1 , LEN3 , RIGHT ); // Число 12345.6789 будет выведено как 5.67
```

Вывод без форматирования:

dispLED.print(1 **);** // Вывод целого числа 1.	
dispLED.print(3.45 **);** // Вывод дробного числа 3.45 (по умолчанию выводится 1 знак после запятой)	
dispLED.print(-12 **);** // Вывод отрицательного целого числа -12.	
dispLED.print(-3.45 **);** // Вывод отрицательного дробного числа -3.45	

dispLED.print(250 , HEX **);** // Вывод числа 250 в шестнадцатиричной системе счисления	
int a[4]={0,3,6,9}; **dispLED.print(** a **);** // Вывод массива из 4 положительных цифр от 0 до 9	
dispLED.print(23 , TEMP **);** // Вывод температуры	
dispLED.print(23.6 , TEMP **);** // Вывод температуры	
dispLED.print(4 , 5 , TIME **);** // Вывод времени	
dispLED.print("67" **);** // Вывод текста	
dispLED.print("OFF" **);** // Вывод текста из букв латинского алфавита	
dispLED.print("8:9" **);** // Вывод текста с двоеточием	

Вывод чисел с указанием количества знаков после запятой:

dispLED.print(1 , 2 **);** // Вывод целого числа 1 с 2 знаками после запятой	
dispLED.print(-2 , 1 **);** // Вывод отрицательного целого числа -2 с 1 знаком после запятой	
dispLED.print(3.45 , 2 **);** // Вывод дробного числа 3.45 с 2 знаками после запятой	
dispLED.print(-3.45 , 0 **);** // Вывод отрицательного дробного числа -3.45 без знаков после запятой	

Вывод чисел с указанием их размерности:

dispLED.print(1 , LEN2 **);** // Вывод целого числа 1 в 2 разрядах	
dispLED.print(-1 , LEN3 **);** // Вывод отрицательного целого числа -1 в 3 разрядах	
dispLED.print(2.3 , LEN3 **);** // Вывод дробного числа 2.3 в 3 разрядах	
dispLED.print(-4.567 , 1 , LEN4 **);** // Вывод дробного числа -4.567 в 4 разрядах , с 1 знаком после запятой	
dispLED.print(8901 , LEN1 **);** // Вывод целого числа 8901 в 1 разряде	
dispLED.print(7 , LEN3 , '-' **);** // Вывод числа 7 в 3 разрядах, с заполнением пустых разрядов символом '-'	

Указание направления сдвига чисел:

dispLED.print(1 , RIGHT **);** // Вывод целого числа 1 со сдвигом вправо от старшего разряда	
dispLED.print(12 , RIGHT **);** // Вывод целого числа 12 со сдвигом вправо от старшего разряда	
dispLED.print(12.3 , RIGHT **);** // Вывод дробного числа 12.3 со сдвигом вправо от старшего разряда	
dispLED.print(1 , LEFT **);** // Вывод целого числа 1 со сдвигом влево (по умолчанию) от младшего разряда	
dispLED.print(12 , LEFT **);** // Вывод целого числа 12 со сдвигом влево (по умолчанию) от младшего разряда	

dispLED.print(12.3 , LEFT **);** // Вывод дробного числа 12.3 со сдвигом влево (по умолчанию) от младшего разряда	

Вывод чисел с привязкой к определённой позиции от левого края:

dispLED.print(74 , POS4 **);** // Вывод целого числа 74 со сдвигом влево (по умолчанию) от 4 позиции	
dispLED.print(74 , POS3 **);** // Вывод целого числа 74 со сдвигом влево (по умолчанию) от 3 позиции	
dispLED.print(74 , POS2 **);** // Вывод целого числа 74 со сдвигом влево (по умолчанию) от 2 позиции	
dispLED.print(74 , POS1 **);** // Вывод целого числа 74 со сдвигом влево (по умолчанию) от 1 позиции	
dispLED.print(74 , POS4 , RIGHT **);** // Вывод целого числа 74 со сдвигом вправо от 4 позиции	
dispLED.print(74 , POS3 , RIGHT **);** // Вывод целого числа 74 со сдвигом вправо от 3 позиции	
dispLED.print(74 , POS2 , RIGHT **);** // Вывод целого числа 74 со сдвигом вправо от 2 позиции	
dispLED.print(74 , POS1 , RIGHT **);** // Вывод целого числа 74 со сдвигом вправо от 1 позиции	

Параметры вывода числа можно комбинировать в любой последовательности:

за исключением числа, указывающего количество знаков после запятой, оно должно быть первым после выводимого значения

dispLED.print(12345.6789 , 2 , POS1 , LEN3 , RIGHT **);** // Вывод дробного числа 12345.6789	

```
// выводить 2 знака после запятой
// POS1 - число вывести на дисплей начиная с 1 позиции от левого края
// LEN3 - размерность выводимого числа 3 разряда
// RIGHT - число сдвигать вправо от указанной позиции.
```

Дополнительные функции библиотеки:

В библиотеке реализованы две дополнительные функции, применение которых может быть интересно в познавательных целях.

Функция fps();

- Назначение: Установка частоты обновления всего изображения индикатора.
- Синтаксис: fps(ЧАСТОТА);
- Параметр:
  - bool ЧАСТОТА - значение от 1 до 255 Гц.
- Возвращаемые значения: Нет.
- Примечание:
  - Светодиоды индикатора светятся не одновременно, а поочерёдно включаются и выключаются так, что в любой момент времени может светиться только один светодиод.
  - Частота заданная данной функцией аналогична частоте кадровой развёртки дисплея, она определяет сколько раз в секунду обновится всё изображение индикатора.
  - На частотах ниже 30 Гц заметно мерцание, а на частотах ниже 10 Гц можно отследить в каком порядке включаются светодиоды.
- Пример:

```
dispLED.fps(2);  // Обновлять изображение индикатора всего 2 раза в секунду.
```

Функция scheme();

- Назначение: Установка схемы включения светодиодов.
- Синтаксис: scheme(СХЕМА);
- Параметр:

    - uint8_t СХЕМА - может принимать одно из двух значений:
        - LED_CA - светодиоды индикатора включены по схеме с общим анодом.
        - LED_CC - светодиоды индикатора включены по схеме с общим катодом.
- Возвращаемые значения: Нет.
- Примечание:
    - Изменение схемы включения светодиодов может быть полезно только при замене индикатора на плате модуля.
- Пример:

```
dispLED.scheme(LED_CA);  // Указываем модулю управлять индикатором собранном по схеме с общими анодами.
dispLED.scheme(LED_CC);  // Указываем модулю управлять индикатором собранном по схеме с общими катодом.
```

Ссылки:

- Библиотека iarduino_I2C_4LED.
- Расширенные возможности библиотек iarduino для шины I2C.
- Wiki - Установка библиотек в Arduino IDE.