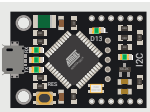


Общие функции и переменные:

#include <iarduino_Metro.h>

Подключение библиотеки iarduino_Metro

iarduino_Metro_Start();

Определение подключённых модулей Metro.

i = iarduino_Metro_Start();

Определение подключённых модулей Metro с возвращением их количества. После определения подключенных модулей станет доступен массив Metro[ ], каждый элемент которого является объектом подключенного модуля и предназначен для управления им.

Metro[N].address;

Адрес модуля на шине I2C.
Тип переменной: uint8_t
Формируется и присваивается во время определения модулей функцией **iarduino_Metro_Start()**. Не меняйте адрес, так как он используется библиотекой.

Metro[N].model;

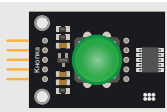
Идентификатор типа модуля.
Тип переменной: uint8_t
Тип модуля считывается во время определения модулей функцией **iarduino_Metro_Start()**. Не меняйте тип модуля, так как он используется библиотекой.

Metro[N].version;

Версия прошивки модуля.
Тип переменной: uint8_t
Версия прошивки считывается во время определения модулей функцией **iarduino_Metro_Start()**. Изменение значения переменной не изменит версию модуля и не повлияет на работу модуля и библиотеки.

Metro[N].size;

Объем памяти ОЗУ в байтах, используемый библиотекой для данного модуля.
Тип переменной: uint8_t
Библиотека создаёт массив объектов Metro[ ] с помощью функции **iarduino_Metro_Start()** в коде **setup()**, значит и память ОЗУ для работы с модулями выделяется там же, а не во время компиляции скетча. Зная объем выделенной памяти можно определить объем свободной и рассчитать, какое количество модулей Metro ещё можно подключить. Изменение значения переменной не изменит объём выделенной памяти и не повлияет на работу модуля и библиотеки.



Кнопка

Metro[N].read(ПАРАМЕТР);

Чтение событий и состояний кнопки.

Параметры:

KEY_PUSHED
Metro[N].read(KEY_PUSHED);
Вернуть **true** только если было событие «**нажимается**», после чего функция будет возвращать **false**, пока событие не повторится.

KEY_RELEASED
Metro[N].read(KEY_RELEASED);
Вернуть **true**, только если было событие «**отпускается**», после чего функция будет возвращать **false**, пока событие не повторится.

KEY_PRESSED
Metro[N].read(KEY_PRESSED);
Вернуть **true**, если кнопка нажата, иначе вернуть **false**. Параметр используется по умолчанию (если функция **read()** вызвана **без параметра**).

KEY_TRIGGER
Metro[N].read(KEY_TRIGGER);
Вернуть либо **true**, либо **false**, меняя это значение с каждым нажатием на кнопку (можно использовать для вкл/выкл устройств нажатием кнопки).

KEY_HOLD1
Metro[N].read(KEY_HOLD1);
Вернуть **true**, если кнопка удерживается дольше, чем указано в 1 аргументе функции **set()** (по умолчанию 1 секунда), иначе вернуть **false**.

KEY_HOLD2
Metro[N].read(KEY_HOLD2);
Вернуть **true**, если кнопка удерживается дольше, чем указано во 2 аргументе функции **set()** (по умолчанию 2 секунда), иначе вернуть **false**.

KEY_HOLD3
Metro[N].read(KEY_HOLD3);
Вернуть **true**, если кнопка удерживается дольше, чем указано в 3 аргументе функции **set()** (по умолчанию 3 секунда), иначе вернуть **false**.

KEY_CHANGED
Metro[N].read(KEY_CHANGED);
Вернуть **true**, если состояние кнопки изменилось (нажалась/отпустилась), после чего функция будет возвращать **false**, пока состояние не изменится.

KEY_TIME
Metro[N].read(KEY_PUSHED);
Вернуть время, прошедшее с момента установки текущего состояния кнопки в миллисекундах, от 0 до 25500 мс (кратно 100 мс). Если кнопка нажата, то возвращается время с момента её нажатия, если отпущена — время с момента её отпускания. Если прошло более 25500 мс, то время останавливается на данном значении.

KEY_SUM
Metro[N].read(KEY_PUSHED);
Вернуть количество нажатий на кнопку за то время, пока не было обращений к функции **read()**. Это позволяет зафиксировать нажатия, выполненные во время ожиданий **delay()** и т.д.

Metro[N].set(ВРЕМЯ1, ВРЕМЯ2, ВРЕМЯ3);

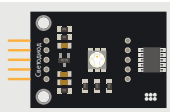
Задаёт время удержания кнопки в миллисекундах для срабатывания функции **read()** с одним из параметров: **KEY_HOLD1**, **KEY_HOLD2**, **KEY_HOLD3**. Если функцию не вызывать, то используются значения по умолчанию: **ВРЕМЯ1**=1000 мс, **ВРЕМЯ2**=2000 мс, **ВРЕМЯ3**=3000 мс.

Параметры:

ВРЕМЯ1
Metro[N].set(500, ВРЕМЯ2, ВРЕМЯ3);
Целочисленное значение, определяющее время удержания кнопки, по истечении которого функция **read()**, вызванная с параметром **KEY_HOLD1**, станет возвращать **true**.

ВРЕМЯ2
Metro[N].set(500, 2500, ВРЕМЯ3);
Целочисленное значение, определяющее время удержания кнопки, по истечении которого функция **read()**, вызванная с параметром **KEY_HOLD2**, станет возвращать **true**.

ВРЕМЯ3
Metro[N].set(500, 2500, 5000);
Целочисленное значение, определяющее время удержания кнопки, по истечении которого функция **read()**, вызванная с параметром **KEY_HOLD3**, станет возвращать **true**.



RGB светодиод

Metro[N].off();

Выключение светодиода.

Metro[N].on(ЦВЕТ,
ДЛИТЕЛЬНОСТЬ,
ПАУЗА,
КОЛИЧЕСТВО);

Metro[N].on(RGB(красный, зелёный, синий),
ДЛИТЕЛЬНОСТЬ,
ПАУЗА,
КОЛИЧЕСТВО);

Metro[N].on(RGB(ФУНКЦИЯ),
ДЛИТЕЛЬНОСТЬ,
ПАУЗА);

Включение светодиода. Все параметры функции **on()** являются **необязательными**.

Параметры:

БЕЗ ПАРАМЕТРА
`Metro[N].on();`
Включает светодиод белым цветом.

ЦВЕТ
`Metro[N].on(0xff5500);`
Можно задать одним числом в шестнадцатеричной системе счисления (как в Photoshop). Пример: **0x**ff5500, где **ff** - красный; **55** - зелёный; **00** - синий; Так же можно задать через функцию **RGB()**, см. ниже.

ДЛИТЕЛЬНОСТЬ
`Metro[N].on(0xff5500, 500);`
Целочисленное значение определяющее время свечения светодиода в миллисекундах от 0 до 25500. Если **ДЛИТЕЛЬНОСТЬ** не указана, то светодиод будет светиться постоянно.

ПАУЗА
`Metro[N].on(0xff5500, 500, 1000);`
Целочисленное значение, определяющее время паузы между свечениями в миллисекундах от 0 до 25500. Если указан **ЦВЕТ** и **ДЛИТЕЛЬНОСТЬ**, а **ПАУЗА** не указана, то светодиод включится однократно на указанное в **ДЛИТЕЛЬНОСТИ** время (без повторов).

КОЛИЧЕСТВО
`Metro[N].on(0xff5500, 500, 1000, 5);`
Целое количество миганий светодиодом от 1 до 255. Если указаны **ЦВЕТ**, **ДЛИТЕЛЬНОСТЬ** и **ПАУЗА**, а **КОЛИЧЕСТВО** миганий не указано, то светодиод будет мигать постоянно.

RGB(красный, зелёный, синий)
`Metro[N].on(RGB(255, 85, 0));`
Альтернативный способ задания цвета. Цвет указывается через дополнительную функцию **RGB()**, разделяющую цвет на яркость **красного**, **зелёного** и **синего** как целое значение от 0 до 255. Пример: **RGB(255 , 85 , 0)**, где **красный** = 255(макс), **зелёный** = 85, **синий** = 0 (мин).

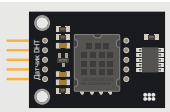
RGB(ФУНКЦИЯ)
`Metro[N].on(RGB(2));`
Включение функции автоперелива цвета, где параметр **ФУНКЦИЯ** является номером функции автоперелива, целое число от 1 до 255.

Metro[N].frequency(ЧАСТОТА);

Установка частоты ШИМ светодиода.

Параметры:

ЧАСТОТА
`Metro[N].frequency(25);`
Задаётся целым числом Гц от 10 до 255. Яркость цветов определяется коэффициентом заполнения ШИМ, частоту которого можно менять в пределах от 10 до 255 Гц. По умолчанию используются сигналы ШИМ с частотой 100 Гц. Это оптимальное значение для корректной работы модуля.



Датчик влажности и температуры

Metro[N].read(ПАРАМЕТР);

Чтение температуры и влажности. Значения влажности и температуры измеряются модулем каждые 3сек.

Параметры:

DHT_TEMPERATURE
`Metro[N].read(DHT_TEMPERATURE);`
Вернуть текущую температуру в °C, от -40 до +80.

DHT_HUMIDITY
`Metro[N].read(DHT_HUMIDITY);`
Вернуть относительную влажность воздуха в %, от 0 до 100.

DHT_CHANGED_TEM
`Metro[N].read(DHT_CHANGED_TEM);`
Вернуть **true**, если температура изменилась, иначе вернуть **false**. Считается, что температура изменилась, если с момента последнего изменения она увеличилась или уменьшилась на значение, указанное в 1 аргументе функции **set()** (по умолчанию 1°C).

DHT_CHANGED_HUM
`Metro[N].read(DHT_CHANGED_HUM);`
Вернуть **true**, если влажность изменилась, иначе вернуть **false**. Считается, что влажность изменилась, если с момента последнего изменения она увеличилась или уменьшилась на значение, указанное во 2 аргументе функции **set()** (по умолчанию 1%).

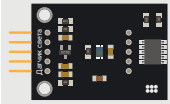
Metro[N].set(ТЕМПЕРАТУРА, ВЛАЖНОСТЬ);

Функция задаёт значения для определения факта изменения температуры или влажности. Если функцию **set()** не вызывать, то будут использоваться значения, установленные по умолчанию: 1°C и 1%.

Параметры:

ТЕМПЕРАТУРА
`Metro[N].set(5, ВЛАЖНОСТЬ);`
Целое количество °C (от 1 до 25), на которое должна подняться или упасть температура, чтобы функция **read()** с параметром **DHT_CHANGED_TEM** вернула **true**.

ВЛАЖНОСТЬ
`Metro[N].set(5, 10);`
Целое количество % (от 1 до 25), на которое должна подняться или упасть влажность, чтобы функция **read()** с параметром **DHT_CHANGED_HUM** вернула **true**.



Датчик освещённости

Metro[N].read(ПАРАМЕТР);

Чтение освещённости.

Параметры:

DSL_LUX
Metro[N].read(DSL_LUX);
Вернуть текущую освещённость в лк, от 0 до 65535.

DSL_CHANGED
Metro[N].read(DSL_CHANGED);
Вернуть **true**, если освещённость изменилась, иначе вернуть **false**. Считается, что освещённость изменилась, если с момента последнего изменения она увеличилась или уменьшилась на значение, указанное функцией **set()** (по умолчанию 1 лк).

Metro[N].set(ОСВЕЩЁННОСТЬ);

Задаёт значение, определяющее факт изменения освещённости. Если функцию **set()** не вызывать, то будет использовано значение по умолчанию: 100 лк.

Параметры:

ОСВЕЩЁННОСТЬ
Metro[N].set(50);
Целое количество лк (от 1 до 255), на которое должна увеличиться или уменьшиться текущая освещённость, чтобы функция **read()** с параметром **DSL_CHANGED** вернула **true**.

Зуммер

Metro[N].off();

Выключение звука.

Metro[N].on(ЧАСТОТА,
ДЛИТЕЛЬНОСТЬ,
ПАУЗА,
КОЛИЧЕСТВО);

Metro[N].on(MELODY(НОМЕР));

Воспроизведение звука. Все параметры являются **необязательными**.

Параметры:

БЕЗ ПАРАМЕТРА
Metro[N].on();
Воспроизвести тональный сигнал с частотой 2,7 кГц.

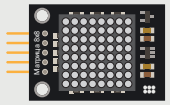
ЧАСТОТА
Metro[N].on(1000);
Целочисленное значение, определяющее высоту звука в Гц, от 10 до 10000. Если **ЧАСТОТА** не указана, то будет воспроизведён тональный сигнал с частотой 2,7 кГц.

ДЛИТЕЛЬНОСТЬ
Metro[N].on(1000, 2000);
Целочисленное значение, определяющее время воспроизведения тонального сигнала в мс, от 0 до 25500. Если **ДЛИТЕЛЬНОСТЬ** не указана, то сигнал будет воспроизводиться постоянно.

ПАУЗА
Metro[N].on(1000, 2000, 200);
Целочисленное значение, определяющее время паузы между тональными сигналами в мс, от 0 до 25500. Если указана **ЧАСТОТА** и **ДЛИТЕЛЬНОСТЬ**, а **ПАУЗА** не указана, то тональный сигнал будет воспроизведён однократно.

КОЛИЧЕСТВО
Metro[N].on(1000, 2000, 200, 3);
Целое количество повторений тонального сигнала, от 1 до 255. Если указаны **ЧАСТОТА**, **ДЛИТЕЛЬНОСТЬ** и **ПАУЗА**, а **КОЛИЧЕСТВО** повторов не указано, то тональные сигналы будут повторяться постоянно.

MELODY(НОМЕР)
Metro[N].on(MELODY(5));
Воспроизведение сигнала или мелодии в соответствии с указанным **НОМЕРОМ**.



Матрица 8x8

Metro[N].set(СИМВОЛ);

Metro[N].set(«ТЕКСТ», СКОРОСТЬ, ПАУЗА);

Выводит символ или текст бегущей строкой. Бегущая строка приостанавливается на первом и последнем символе на 1 секунду. Если указан параметр ПАУЗА (не равный 0), то по окончании указанного времени бегущая строка запустится заново, иначе после прохождения бегущей строки дисплей погаснет.

Параметры:

СИМВОЛ
Metro[N].set('H');
Символ или код символа от 1 до 255, который требуется вывести на дисплей.

«ТЕКСТ»
Metro[N].set(«Hello world!»);
Содержимое бегущей строки, до 512 символов. Текст записывается в модуль и выводится с указанной скоростью, после чего дисплей гаснет.

СКОРОСТЬ
Metro[N].set(«Hello world!», 250);
Целочисленное значение от 0 до 255, определяющее скорость движения бегущей строки: чем выше значение, тем быстрее прокручивается текст. Если указан 0, то бегущая строка сдвигаться не будет. Если указано значение от 1 до 255, то задержка между сдвигом бегущей строки на 1 пиксель будет равна (256 - **СКОРОСТЬ**) / 100 секунд.

ПАУЗА
Metro[N].set(«Hello world!», 250, 5000);
Целочисленное значение, определяющее время до повторного запуска бегущей строки. Указывается в миллисекундах от 0 до 25500 и округляется до сотых. Если параметр отсутствует или равен 0, то повторный запуск осуществляться не будет.

Metro[N].frequency(ЧАСТОТА);

Задаёт частоту обновления экрана.

Параметры:

ЧАСТОТА
Metro[N].frequency(25);
Целочисленное значение, определяющее частоту обновления дисплея как количество кадров в секунду, от 10 до 255. По умолчанию задано обновление 100 кадров/сек.

Metro[N].off(АНИМАЦИЯ);

Выключение дисплея (с анимацией исчезания изображения или без). По окончании анимации исчезания в пустой фон все светодиоды будут выключены. По окончании анимации исчезания в закрашенный фон все светодиоды будут светиться. При выключении дисплея без анимации все светодиоды одновременно погаснут.

Параметры:

БЕЗ ПАРАМЕТРА
Metro[N].off();
Выключить дисплей без анимации.

АНИМАЦИЯ
`Metro[N].off();` — Выключить дисплей с анимацией.

Значения АНИМАЦИИ выключения:

X8_EMPTY_RIPPLES
`Metro[N].off(X8_EMPTY_RIPPLES);` — Анимация исчезания рябью в пустой фон.

X8_FILLED_RIPPLES
`Metro[N].off(X8_FILLED_RIPPLES);` — Анимация исчезания рябью в закрашенный фон.

X8_EMPTY_DOWN
`Metro[N].off(X8_EMPTY_DOWN);` — Анимация исчезания сверху-вниз в пустой фон.

X8_FILLED_DOWN
`Metro[N].off(X8_FILLED_DOWN);` — Анимация исчезания сверху-вниз в закрашенный фон.

X8_EMPTY_TOP
`Metro[N].off(X8_EMPTY_TOP);` — Анимация исчезания снизу-вверх в пустой фон.

X8_FILLED_TOP
`Metro[N].off(X8_FILLED_TOP);` — Анимация исчезания снизу-вверх в закрашенный фон.

Metro[N].on(МАССИВ, АНИМАЦИЯ);

Metro[N].on(ЯРКОСТЬ);

Metro[N].on(ПОВОРОТ);

Включение дисплея (с анимацией появления изображения или без). В начале выполнения анимации дисплей либо очищается, либо закрашивается, создавая фон для появления изображения МАССИВА. По окончании анимации на дисплее будет изображение МАССИВА. Функция с параметрами **ЯРКОСТЬ** или **ПОВОРОТ** не включает дисплей, а устанавливает его параметры. Допускается указывать **АНИМАЦИЮ** в качестве первого параметра(без **МАССИВА**).

Параметры:

БЕЗ ПАРАМЕТРА
`Metro[N].on();` — Включить все светодиоды.

МАССИВ
`Metro[N].on(image);` — 8 байтовый массив, биты которого сформируют изображение на дисплее.

АНИМАЦИЯ
`Metro[N].on(X8_EMPTY_DOWN;` — Вывести на дисплей массив изображения с анимацией. Можно использовать без указания массива. Значения см.ниже.

ЯРКОСТЬ
`Metro[N].on(5);` — Целочисленное значение от 0 до 10, определяющее яркость дисплея. Установка яркости не изменит текущее изображение дисплея.

ПОВОРОТ
`Metro[N].on(X8_ANGLE_180);` — Устанавливает угол поворота дисплея. Задание угла повернёт изобрежение, но не изменит его. Значения см.ниже.

Значения АНИМАЦИИ появления:

X8_EMPTY_RIPPLES
`Metro[N].on(X8_EMPTY_RIPPLES);` — Анимация появления рябью из пустого фона.

X8_FILLED_RIPPLES
`Metro[N].on(X8_FILLED_RIPPLES);` — Анимация появления рябью из закрашенного фона.

X8_EMPTY_DOWN
`Metro[N].on(X8_EMPTY_DOWN);` — Анимация появления сверху-вниз из пустого фона.

X8_FILLED_DOWN
`Metro[N].on(X8_FILLED_DOWN);` — Анимация появления сверху-вниз из закрашенного фона.

X8_EMPTY_TOP
`Metro[N].on(X8_EMPTY_TOP);` — Анимация появления снизу-вверх из пустого фона.

X8_FILLED_TOP
`Metro[N].on(X8_FILLED_TOP);` — Анимация появления снизу-вверх из закрашенного фона.

Значения ПОВОРОТОВ изображения:

X8_ANGLE_0
`Metro[N].on(X8_ANGLE_0);` — Выводить данные на дисплей с поворотом в 0° по часовой стрелке.

X8_ANGLE_90
`Metro[N].on(X8_ANGLE_90);` — Выводить данные на дисплей с поворотом в 90° по часовой стрелке.

X8_ANGLE_180
`Metro[N].on(X8_ANGLE_180);` — Выводить данные на дисплей с поворотом в 180° по часовой стрелке.

X8_ANGLE_270
`Metro[N].on(X8_ANGLE_270);` — Выводить данные на дисплей с поворотом в 270° по часовой стрелке.

Типы данных

Однобайтовые типы данных:

bool
boolean — Целочисленные значения. Принимают любые, возвращают **0** или **1**.

char
int8_t — Целочисленные значения или символы от **-128** до **127**

byte
unsigned char
uint8_t — Беззнаковые целочисленные значения от **0** до **255**

Двухбайтовые типы данных:

int
short
int16_t — Целочисленные значения от **-32'768** до **32'767**

word
unsigned int
unsigned short
uint16_t — Беззнаковые целочисленные значения от **0** до **65'535**

Четырёхбайтовые типы данных:

long
int32_t — Целочисленные значения от **-2'147'483'648** до **2'147'483'647**

unsigned long
uint32_t — Беззнаковые целочисленные значения от **0** до **4'294'967'295**

float — Числа с плавающей точкой от **-2'147'483'648,0** до **2'147'483'647,0**

Восьмибайтовые типы данных:

double — Числа с плавающей точкой удвоенной точности от **-9'223'372'036'854'775'808,0** до **9'223'372'036'854'775'807,0**

long long
int64_t — Целочисленные значения от **-9'223'372'036'854'775'808** до **9'223'372'036'854'775'807**

unsigned long long
uint64_t — Беззнаковые целочисленные значения от **0** до **18'446'744'073'709'551'615**

Типы данных с определяемым размером:

void — Отуствие данных

String — Строка

тип A[размер] — Массив A указанного размера, с элементами указанного типа

тип A[ ] — Массив A указанного типа, без прямого указания размера

тип A[размер][размер] — Двумерный массив A указанного типа и размера

тип A[ ][ ] — Двумерный массив A указанного типа, без прямого указания размера

char A[размер] — Массив или строка, состоящая из указанного количества символов

char A[ ] — Массив или строка без прямого указания количества символов