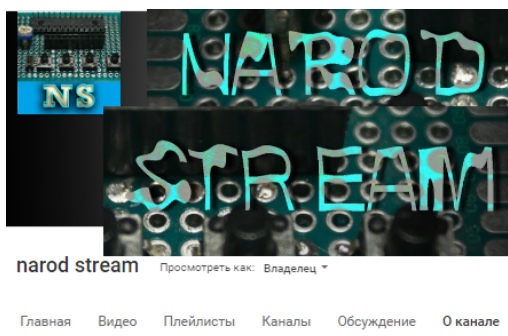


Сайт narodstream.ru

создан в поддержку
канала YouTube

NAROD STREAM








Рубрики

- [Uncategorized](#)
- [Программирование AVR](#)
- [Программирование STM32](#)

Свежие записи

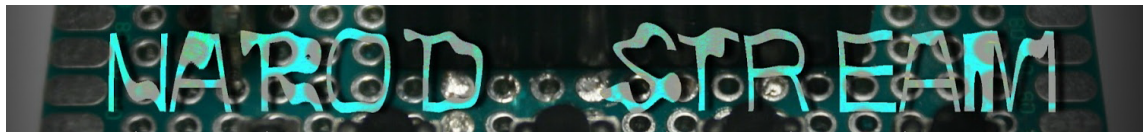
- [STM32. Урок 94. DS18B20. Несколько датчиков на одной шине. Часть 1](#)
- [Ограничение доступа к сайту за чрезмерную активность](#)
- [STM32. Урок 93. LAN. W5500. HTTP Server. Сокеты. Часть 2](#)
- [STM32. Урок 93. LAN. W5500. HTTP Server. Сокеты. Часть 1](#)
- [STM32. Урок 92. Датчик температуры DS18B20. Часть 3](#)

Последние ответы на форуме

-  [Narod Stream](#) в [Программирование МК STM32](#)
2 дн., 3 час. назад
-  [Mihail](#) в [Программирование МК STM32](#)
2 дн., 4 час. назад
-  [Dmitriy](#) в [Программирование МК AVR](#)
2 нед., 5 дн. назад
-  [nsk](#) в [Программирование МК STM32](#)
2 нед., 6 дн. назад
-  [Narod Stream](#) в [Программирование МК STM32](#)
3 нед. назад

Свежие комментарии

- [Narod Stream](#) к записи [AVR Урок 4](#).
Смотрим результаты работы
- [Narod Stream](#) к записи [STM Урок 44](#).
[SDIO. FATFS](#)
- Виктор к записи [AVR Урок 4](#). Смотрим
результаты работы
- [Narod Stream](#) к записи [STM Урок 56](#).
[System Workbench](#). Подключаем
библиотеку BSP. Часть 1
- 3k к записи [STM Урок 56](#). [System Workbench](#). Подключаем библиотеку
BSP. Часть 1

[Главная](#)[Новости](#)[Уроки по программированию МК](#)[Ссылки](#)[Форум](#)[Помощь](#)

Просмотров:

[Главная](#) > [AVR Урок 34. Дисплей TFT 240×320 8bit. Часть 6](#)

Урок 34 Часть 6

Дисплей TFT 240×320 8bit

В [предыдущей части](#) нашего занятия мы написали ещё несколько функций для работы с дисплеем и вывели некоторые примитивы на его экран.

Теперь давайте зайдём в файл `ili9341.c` и напишем вывод на экран дисплея линии с координатами начала и окончания. Так как эта функция оказалась непростой, то сначала мы её просто создадим, а писать будем постепенно

```
//  
void TFT9341_DrawLine(unsigned int color, unsigned int  
x1, unsigned int y1, unsigned int x2, unsigned int y2)  
{  
}  
//
```

Как обычно сначала во входных параметрах у нас цвет, а затем координаты начала и окончания линии.

Применим вот такую конструкцию

Мета

- [Регистрация](#)
- [Войти](#)
- [RSS записей](#)
- [RSS комментариев](#)
- [WordPress.org](#)

Уроки по программированию МК

[Программирование МК AVR](#)[Программирование МК STM32](#)[Программирование МК PIC](#)[Тесты устройств и аксессуаров](#)

```
void TFT9341_DrawLine(unsigned int color, unsigned int
x1, unsigned int y1, unsigned int x2, unsigned int y2)
{
    int steep = abs(y2 - y1) > abs(x2 - x1);
```

Здесь переменная будет равна единице в случае выполнения условия, а в случае невыполнения будет нулём. В условии у нас неравенство. Мы сравниваем — что у нас больше — разность между вертикальными координатами или между горизонтальными.

И затем уже в зависимости от результата войдём в условие

```
int steep = abs(y2 - y1) > abs(x2 - x1);
if (steep) {
    swap(x1, y1);
    swap(x2, y2);
}
```

Если результат истинный, то мы поменяемся значениями вертикальных и горизонтальных координат.

Теперь, если у нас первая горизонтальная координата больше второй, то поменяемся их значениями, причём поменяемся и вертикальными

```
    swap(x2, y2);
}
if (x1 > x2) {
    swap(x1, x2);
    swap(y1, y2);
}
```

Добавим 2 переменных

```
    swap(y1, y2);
}
int dx, dy;
```

Занесём в них разность координат

```
int dx, dy;
dx = x2 - x1;
dy = abs(y2 - y1);
```

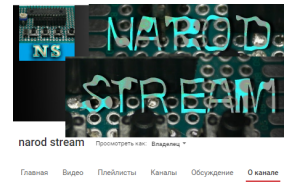
В горизонтальных координатах мы не применяем функцию абсолютной величины, так как выше мы уже добились того, что теперь у нас x2 всегда больше или равно x1.

сохраним в переменную половину разницы между вертикальными координатами и создадим ещё одну переменную

```
dy = abs(y2 - y1);
int err = dx / 2;
int ystep;
```

Затем, в зависимости от того, какая вертикальная координата меньше, занесём в эту переменную значение

Заходите на канал Narod Stream



Архивы

- Октябрь 2017
- Сентябрь 2017
- Август 2017
- Июль 2017
- Июнь 2017
- Май 2017
- Апрель 2017
- Март 2017

```
int ystep;
if (y1 < y2) {
    ystep = 1;
} else {
    ystep = -1;
}
```

И в конце функции цикл, в котором мы и будем рисовать нашу прямую линию, постепенно инкрементируя горизонтальную координату

```
    ystep = -1;
}
for (; x1<=x2; x1++) {
}
}
```

Теперь займёмся телом данного цикла.

А в цикле будет условие, в котором мы нарисуем точку линии в зависимости от значения переменной **steep**

```
for (; x1<=x2; x1++) {
if (steep) {
    TFT9341_DrawPixel(y1, x1, color);
} else {
    TFT9341_DrawPixel(x1, y1, color);
}
}
```

Далее занесём в переменную разницу вертикальных координат с противоположным знаком

```
TFT9341_DrawPixel(x1, y1, color);
}
err -= dy;
```


Ну и в конце цикла условие, в котором мы наращиваем значение y1 или убавляем его в зависимости от значения переменной ystep

```
err -= dy;
if (err < 0) {
    y1 += ystep;
    err += dx;
}
}
}
```

Вот такая вот интересная функция. Ну, а теперь, конечно, тест в main(), но перед этим мы не забываем про прототип в заголовочном файле. В тесте мы выведем вертикальные линии случайного цвета

```
TFT9341_FillScreen(BLACK);
for(i=0;i<240;i++)
{
```

- [Февраль 2017](#)
- [Январь 2017](#)
- [Декабрь 2016](#)
- [Ноябрь 2016](#)



31 ДЕНЬ	88 083
	9 064
07 ДНЕЙ	22 948
	2 994
24 ЧАСА	3 036
	713
СЕГОДНЯ	1 414
	385
НАПЛИНУ	82
	23

```

TFT9341_DrawLine(TFT9341_RandColor(),i,0,i,319);
}
_delay_ms(500);
TFT9341_FillScreen(BLACK);

```

Скомпилируем наш код, прошьём контроллер и посмотрим на наш дисплей



Далее напомним тест вывода прямых линий со случайными координатами и случайным цветом

```

TFT9341_FillScreen(BLACK);
for(i=0;i<1000;i++)
{
    TFT9341_DrawLine(TFT9341_RandColor(),rand()%240,rand()
    %320,rand()%240,rand()%320);
}
_delay_ms(500);
TFT9341_FillScreen(BLACK);

```

Давайте проверим данный тест



Вернёмся в файл с функциями и напомним функцию рисования незакрашенного прямоугольника. Так как прямоугольники у нас будут горизонтальными или вертикальными (не под наклоном), то реализация этой задачи значительно упрощается до вывода четырёх прямых линий

```
//-----
void TFT9341_DrawRect(unsigned int color,unsigned int
x1, unsigned int y1, unsigned int x2, unsigned int y2)
{
    TFT9341_DrawLine(color,x1,y1,x2,y1);
    TFT9341_DrawLine(color,x2,y1,x2,y2);
    TFT9341_DrawLine(color,x1,y1,x1,y2);
    TFT9341_DrawLine(color,x1,y2,x2,y2);
}
//-----
```

Входные параметры стандартные, как и у закрашенного прямоугольника, а в теле вывод прямых линий: верхней, левой, правой и нижней.

Создадим для функции прототип и напишем интересный тест в main(), в котором выведем наши прямоугольники красивым образом

```
TFT9341_FillScreen(BLACK);
for(i=0;i<120;i++)
{
    TFT9341_DrawRect(TFT9341_RandColor(),i,i,239-i,319-
i);
}
_delay_ms(500);
TFT9341_FillScreen(BLACK);
```

Проверим тест на практике



Теперь напишем функцию отрисовки окружности с определённым радиусом

```
//-----
void TFT9341_DrawCircle(int x0, int y0, int r,
unsigned int color)
{
    int f = 1 - r;
    int ddF_x = 1;
    int ddF_y = -2 * r;
    int x = 0;
    int y = r;
```

```

TFT9341_DrawPixel(x0 , y0+r, color);
TFT9341_DrawPixel(x0 , y0-r, color);
TFT9341_DrawPixel(x0+r, y0 , color);
TFT9341_DrawPixel(x0-r, y0 , color);
while (x<y) {
    if (f >= 0) {
        y--;
        ddF_y += 2;
        f += ddF_y;
    }
    x++;
    ddF_x += 2;
    f += ddF_x;
    TFT9341_DrawPixel(x0 + x, y0 + y, color);
    TFT9341_DrawPixel(x0 - x, y0 + y, color);
    TFT9341_DrawPixel(x0 + x, y0 - y, color);
    TFT9341_DrawPixel(x0 - x, y0 - y, color);
    TFT9341_DrawPixel(x0 + y, y0 + x, color);
    TFT9341_DrawPixel(x0 - y, y0 + x, color);
    TFT9341_DrawPixel(x0 + y, y0 - x, color);
    TFT9341_DrawPixel(x0 - y, y0 - x, color);
}
}
//_____

```

Данная функция была взята из какого-то примера и переработана под наш дисплей.

Во входных параметрах координаты центра, радиус в пикселях и цвет.

Напишем для неё прототип и напишем тест в main(), в котором мы будем выводить в случайное место окружности радиусом в 20 пикселей

```

TFT9341_FillScreen(BLACK);
for(i=0;i<2000;i++)
{

    TFT9341_DrawCircle(rand()%200+20, rand()%280+20, 20, TFT9
341_RandColor());
}
_delay_ms(500);
TFT9341_FillScreen(BLACK);

```

Запас 20 пикселей в координатах сделан для того, чтобы окружности уместились полностью в видимой области и не попали в область вне экрана.

Соберём код, прошьём контроллер и посмотрим результат работы нашего теста на практике



В **следующей части** нашего урока мы напишем функции для работы с текстом и попробуем вывести отдельные символы, а также некоторые строки на дисплей.

*Предыдущая
часть*

*Программирование
МК AVR*

*Следующая
часть*

Техническая документация на контроллер дисплея ILI9341

Программатор и символьный дисплей LCD 20×4 можно приобрести [здесь](#):

Программатор (продавец надёжный) [USBASP USBISP 2.0](#)
[Дисплей LCD 20×4](#)

Смотреть ВИДЕОУРОК (нажмите на картинку)



Добавить комментарий

Ваш e-mail не будет опубликован. Обязательные поля помечены *

Имя *

E-mail *

Сайт

+ девять = четырнадцать ↺

Отправить комментарий

[Главная](#) | [Новости](#) | [Уроки по программированию МК](#)
| [Программирование микроконтроллеров AVR](#)
| [Программирование микроконтроллеров STM32](#)
| [Программирование микроконтроллеров PIC](#) | [Тесты устройств и аксессуаров](#)
| [Ссылки](#) | [Форум](#) | [Помощь](#)



Google



© 2017 Narod Stream

