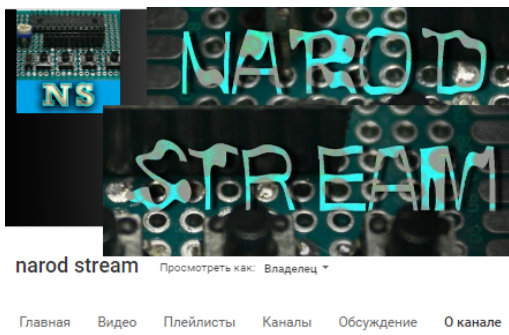


Сайт narodstream.ru

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](#)

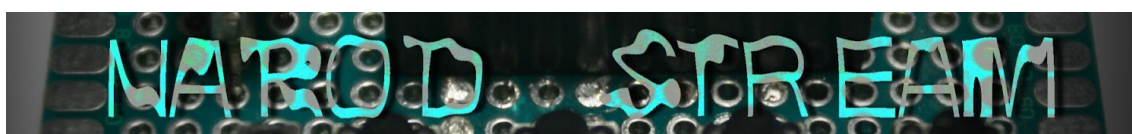
Последние ответы на форуме

-  [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: 41

[Главная](#) › [AVR Урок 34. Дисплей TFT 240×320 8bit. Часть 3](#)

Урок 34 Часть 3

Дисплей TFT 240×320 8bit

Мета

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

Уроки по



Яндекс Директ

Кулачковый переключатель Кулачковый переключатель! Производство! Цена на 15% ниже рынка. Прайс здесь

В **прошлой части** нашего урока мы написали ряд очень важных функций для работы с дисплеем, а также считали его идентификатор, что позволило нам стать уверенными в том, что мы с дисплеем нормально общаемся по 8-разрядной шине и что мы общаемся именно с тем дисплеем, который нам нужен.

Что ж, продолжим. Откроем файл ili9341.c и добавим в него две глобальные переменные для хранения ширины и высоты нашего дисплея в пикселях

```
#include "ili9341.h"
//
```

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

```
unsigned int X_SIZE = 0;
unsigned int Y_SIZE = 0;
unsigned long dt=0;
```

Пока они у нас будут равны нулю, а значения мы им присвоим в процессе инициализации.
Создадим функцию изменения ориентации дисплея

```
//-----
void TFT9341_SetRotation(unsigned char r)
{
}
//-----
```

У нас будет четыре вида ориентации. Два вертикальных — один обычный, один перевёрнутый и подобные два горизонтальных. Условимся обозначать из следующим образом:

- 0 — вертикальная обычная ориентация
- 1 — горизонтальная обычная ориентация
- 2 — вертикальная перевёрнутая ориентация
- 3 — горизонтальная перевёрнутая ориентация.

Теперь начнём писать тело нашей функции.
Посмотрим в даташите регистр 0x36

8.2.29. Memory Access Control (36h)

36h MADCTL (Memory Access Control)													
	D/CX	RDX	WFX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	HEX
Command	0	1	↑	XX	0	0	1	1	0	1	1	0	36h
Parameter	1	1	↑	XX	MY	MX	MV	ML	BGR	MH	0	0	00

This command defines read/write scanning direction of frame memory.
This command makes no change on the other driver status.

Bit	Name	Description
MY	Row Address Order	These 3 bits control MCU to memory write/read direction.
MX	Column Address Order	
MV	Row / Column Exchange	
ML	Vertical Refresh Order	LCD vertical refresh direction control.
BGR	RGB-BGR Order	Color selector switch control (0=RGB color filter panel, 1=BGR color filter panel)
MH	Horizontal Refresh ORDER	LCD horizontal refreshing direction control.

Note: When BGR bit is changed, the new setting is active immediately without update the content in Frame Memory again.
X = Don't care.

MV(Vertical refresh order bit)="0"

MV(Vertical refresh order bit)="1"

ML(Vertical refresh order bit)="0"

ML(Vertical refresh order bit)="1"

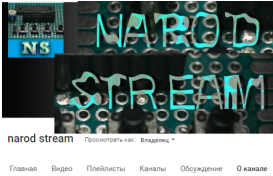
BGR(RGB-BGR Order control bit)="0"

BGR(RGB-BGR Order control bit)="1"

искать здесь ...

Фильтровать

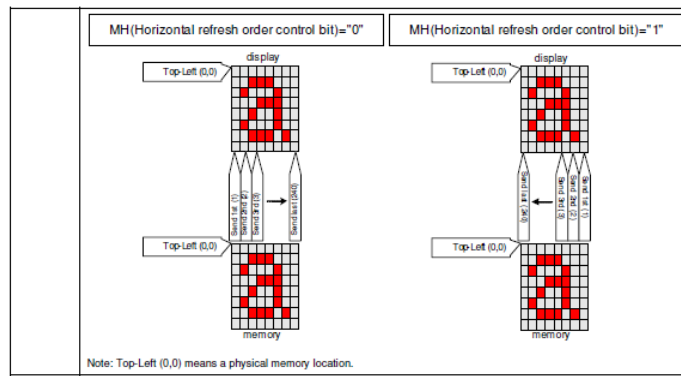
Заходите на канал Narod Stream



Главная Видео Плейлисты Каналы Обсуждение 0 канал

Архивы

- Октябрь 2017
- Сентябрь 2017
- Август 2017



- [Июль 2017](#)
- [Июнь 2017](#)
- [Май 2017](#)
- [Апрель 2017](#)
- [Март 2017](#)
- [Февраль 2017](#)
- [Январь 2017](#)
- [Декабрь 2016](#)
- [Ноябрь 2016](#)

Как мы видим в нотации к данному регистру, что мы можем менять ориентацию в зависимости от состояния определённых битов в параметре.

Поэтому сначала отправим адрес регистра в контроллер дисплея, а затем создадим ветвление в зависимости от вызываемого режима ориентации

```
void TFT9341_SetRotation(unsigned char r)
{
    TFT9341_SendCommand(0x36);
    switch(r)
    {
        case 0:
            break;
        case 1:
            break;
        case 2:
            break;
        case 3:
            break;
    }
}
```

Теперь отправим параметр. Содержимое его будет зависеть от определённых битов ориентации. Биты MH и ML всегда будут в нуле, BGR всегда в единице, а вот значение битов MV, MX и MY будут меняться.

При обычной вертикальной ориентации (входной параметр ноль) в единице из этих трёх битов будет только MX (Column Address Order).

Поэтому напомним первый кейс следующим образом, заодно и присвоим нашим глобальным переменным размеров экрана соответствующие значения

```
case 0:
    TFT9341_SendData(0x48);
    X_SIZE = 240;
    Y_SIZE = 320;
    break;
```

При горизонтальной ориентации (входной параметр 1) в единице будет бит MV (Row / Column Exchange).

Напомним следующий кейс

```
case 1:
    TFT9341_SendData(0x28);
```

	88 083
31.05.17	9 064
07.06.17	22 948
24.06.17	2 994
30.06.17	3 036
01.07.17	713
СЕГОДНЯ	1 398
НА ПЯТНИЦУ	381
	84
	24

```
X_SIZE = 320;
Y_SIZE = 240;
break;
```

При вертикальной перевёрнутой ориентации (входной параметр 2) в единице будет уже бит MY (Row Address Order).

Напишем кейс

```
case 2:
TFT9341_SendData(0x88);
X_SIZE = 240;
Y_SIZE = 320;
break;
```

При горизонтальной перевёрнутой ориентации (входной параметр 3) в единице будут все три бита

```
case 3:
TFT9341_SendData(0xE8);
X_SIZE = 320;
Y_SIZE = 240;
break;
```

Продолжим нашу инициализацию. Допишем в функцию инициализации следующий код, который лишний раз перезагрузит программно наш дисплей, а вернее его контроллер

```
str_lcd(str);
TFT9341_SendCommand(0x01); //Software Reset
```

Следующая команда уже будет требовать данные (целых 5 параметров)

```
TFT9341_SendCommand(0x01); //Software Reset
TFT9341_SendCommand(0xCB); //Power Control A
TFT9341_SendData(0x39);
TFT9341_SendData(0x2C);
TFT9341_SendData(0x00);
TFT9341_SendData(0x34);
TFT9341_SendData(0x02);
```

Ну, я не буду сюда постить весь даташит, дабы не засорять страницу и базу данных, буду объяснять вкратце.

Данный регистр, в которые мы отправили столько данных — это регистр настроек или управления. Все биты, которые мы там включили, вы можете посмотреть в даташите.

Есть ещё один подобный регистр, в который мы отправим три аргумента

```
TFT9341_SendData(0x02);
TFT9341_SendCommand(0xCF); //Power Control B
TFT9341_SendData(0x00);
```

```
TFT9341_SendData(0xC1);  
TFT9341_SendData(0x30);
```

Следующие 2 команды вносят настройки в регистр управления различными таймингами

```
TFT9341_SendData(0x30);  
TFT9341_SendCommand(0xE8); //Driver timing control A  
TFT9341_SendData(0x85);  
TFT9341_SendData(0x00);  
TFT9341_SendData(0x78);  
TFT9341_SendCommand(0xEA); //Driver timing control B  
TFT9341_SendData(0x00);  
TFT9341_SendData(0x00);
```

Также заполним ещё ряд регистров, назначения которых я объяснил в комментариях к каждой команде. Также где-то посередине кода мы вызовем функцию ориентации дисплея и инициализируем там обычный вертикальный режим с индексом 0. На этом этапе у нас и проинициализируются наши глобальные переменные размера нашего экрана

```
TFT9341_SendData(0x00);  
TFT9341_SendCommand(0xED); //Power on Sequence control  
TFT9341_SendData(0x64);  
TFT9341_SendData(0x03);  
TFT9341_SendData(0x12);  
TFT9341_SendData(0x81);  
TFT9341_SendCommand(0xF7); //Pump ratio control  
TFT9341_SendData(0x20);  
TFT9341_SendCommand(0xC0); //Power Control 1  
TFT9341_SendData(0x10);  
TFT9341_SendCommand(0xC1); //Power Control 2  
TFT9341_SendData(0x10);  
TFT9341_SendCommand(0xC5); //VCOM Control 1  
TFT9341_SendData(0x3E);  
TFT9341_SendData(0x28);  
TFT9341_SendCommand(0xC7); //VCOM Control 2  
TFT9341_SendData(0x86);  
TFT9341_SetRotation(0);  
TFT9341_SendCommand(0x3A); //Pixel Format Set  
TFT9341_SendData(0x55); //16bit  
TFT9341_SendCommand(0xB1);  
TFT9341_SendData(0x00);  
TFT9341_SendData(0x18); // Частота кадров 79 Гц  
TFT9341_SendCommand(0xB6); //Display Function Control  
TFT9341_SendData(0x08);  
TFT9341_SendData(0x82);  
TFT9341_SendData(0x27); //320 строк  
TFT9341_SendCommand(0xF2); //Enable 3G (пока не знаю что это за режим)  
TFT9341_SendData(0x00); //не включаем  
TFT9341_SendCommand(0x26); //Gamma set  
TFT9341_SendData(0x01); //Gamma Curve (G2.2) (Кривая цветовой гаммы)  
TFT9341_SendCommand(0xE0); //Positive Gamma Correction  
TFT9341_SendData(0x0F);  
TFT9341_SendData(0x31);  
TFT9341_SendData(0x2B);
```

```
TFT9341_SendData(0x0C);
TFT9341_SendData(0x0E);
TFT9341_SendData(0x08);
TFT9341_SendData(0x4E);
TFT9341_SendData(0xF1);
TFT9341_SendData(0x37);
TFT9341_SendData(0x07);
TFT9341_SendData(0x10);
TFT9341_SendData(0x03);
TFT9341_SendData(0x0E);
TFT9341_SendData(0x09);
TFT9341_SendData(0x00);
TFT9341_SendCommand(0xE1); //Negative Gamma Correction
TFT9341_SendData(0x00);
TFT9341_SendData(0x0E);
TFT9341_SendData(0x14);
TFT9341_SendData(0x03);
TFT9341_SendData(0x11);
TFT9341_SendData(0x07);
TFT9341_SendData(0x31);
TFT9341_SendData(0xC1);
TFT9341_SendData(0x48);
TFT9341_SendData(0x08);
TFT9341_SendData(0x0F);
TFT9341_SendData(0x0C);
TFT9341_SendData(0x31);
TFT9341_SendData(0x36);
TFT9341_SendData(0x0F);
```

Выйдем из спящего режима и включим дисплей

```
TFT9341_SendData(0x0F);
TFT9341_SendCommand(0x11); //Выйдем из спящего режим
_delay_ms(150);
TFT9341_SendCommand(0x29); //Включение дисплея
TFT9341_SendData(0x2C);
_delay_ms(150);
}
```

На этом инициализация закончена.

В **следующей части** нашего занятия мы напишем ещё несколько функций для работы с дисплеем и попробуем залить экран определенным цветом.



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)



Назначение перевода Я хочу поодержать проект Narod Stream

Сумма руб.

Отправить

2 комментария на “AVR Урок 34. Дисплей TFT 240×320 8bit. Часть 3”



Михаил:

Август 27, 2017 в 6:30 дп

Посмотрел инициализацию дисплея, есть замечания.

1. TFT9341_SendCommand(0xB6); //Display Function Control

Команде необходимы 4 параметра.

2. TFT9341_SendCommand(0xF2); //Enable 3G (пока не знаю что это за режим)

Отключена по умолчанию.

3. TFT9341_SendCommand(0x29); //Включение дисплея

TFT9341_SendData(0x2C);

_delay_ms(150);

Команда без параметров.

У меня не совсем корректно выполняются примеры, которые Вы рассматриваете в следующих частях урока, поэтому решил посмотреть инициализацию. Удачи!

[Ответить](#)



Andr125:

Сентябрь 20, 2017 в 7:18 дп

У меня тоже Ваш пример в Proteus не работает. Работает все, что до заливки прямоугольников. Дальше — ничего.

[Ответить](#)

Добавить комментарий

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

Источник	
	▼
▼	
Стили ▼	Формат... ▼
Шрифт ▼	
Ра... ▼	
▼	

Имя *

Е-mail *

Сайт

7 × 4 =

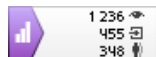


Отправить комментарий

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

