Свежие комментарии

- Сергей к записи PIC Урок 3. Бегущие огни
- Narod Stream к записи PIC Урок 5. Таймеры
- Артем к записи PIC Урок 5. Таймеры
- Narod Stream к записи AVR Урок 13. ШИМ. Мигаем светодиодом плавно. Часть 1
- Narod Stream к записи STM Урок 10. HAL. Изучаем PWM (ШИМ). Мигаем светодиодами плавно

Форум. Последние ответы

- Narod Stream в Программирование МК STM32
  2 дн. назад
- Zandy в Программирование МК STM32
  2 дн., 6 час. назад
- Narod Stream в Программирование МК STM32
  1 неделя, 6 дн. назад
- Narod Stream в Программирование МК STM32
  1 неделя, 6 дн. назад
- fireweb в Программирование МК STM32
  2 нед., 2 дн. назад

Январь 2018

Пн	Вт	Ср	Чт	Пт	Сб	Вс
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				
« Дек						

Архивы

- Январь 2018
- Декабрь 2017
- Ноябрь 2017
- Октябрь 2017
- Сентябрь 2017
- Август 2017
- Июль 2017
- Июнь 2017
- Май 2017

AVR Урок 33. SPI. Карта SD. FAT. Часть 1

Posted on Январь 11, 2017 by Narod Stream Опубликовано в Программирование AVR — Нет комментариев ↓

Урок 33
Часть 1

SPI. Карта SD. FAT

Сегодня мы продолжим нашу любимую тему по интерфейсу SPI. Закончили мы с данной шиной **уроком по подключению** друг к другу контроллеров Atmega8a и ATTtiny2313. А сегодня мы по данному интерфейсу попробуем подключить к микроконтроллеру по данной шине карту памяти **SD (Secure Digital)**.
Данная карта может подключаться также по интерфейсу SDIO, но так как такой интерфейс не поддерживается аппаратно нашим контроллером, то в рамках данного занятия мы его касаться не будем. Нам интересен именно тип подключения по шине **SPI**, так как у нас уже есть неплохие накопленные знания по данной теме, а также аппаратная поддержка в контроллере, который мы программируем. Тем не менее мы посмотрим распиновку ножек карты по обоим типам

Ну, так как нас интересует второй тип, с ним и будем разбираться.
А разбираться тут особо не в чем. Все эти аббревиатуры нам известны. Здесь все стандартные ножки интерфейса SPI и ничего тут лишнего нет.

Мета

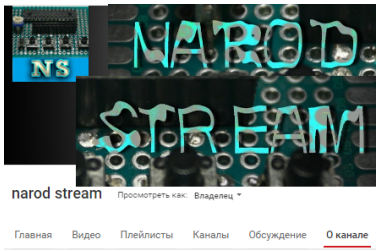
- Регистрация
- Войти
- RSS записей
- RSS комментариев
- WordPress.org

искать здесь …
Фильтровать

Уроки по программированию МК

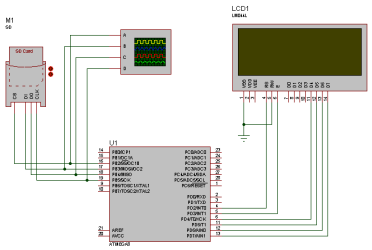
Программирование МК AVR
Программирование МК STM32
Программирование МК PIC
Тесты устройств и аксессуаров

Заходите на канал Narod Stream

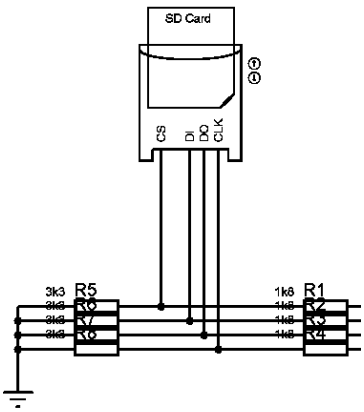


- Март 2017
- Февраль 2017
- Январь 2017
- Декабрь 2016
- Ноябрь 2016

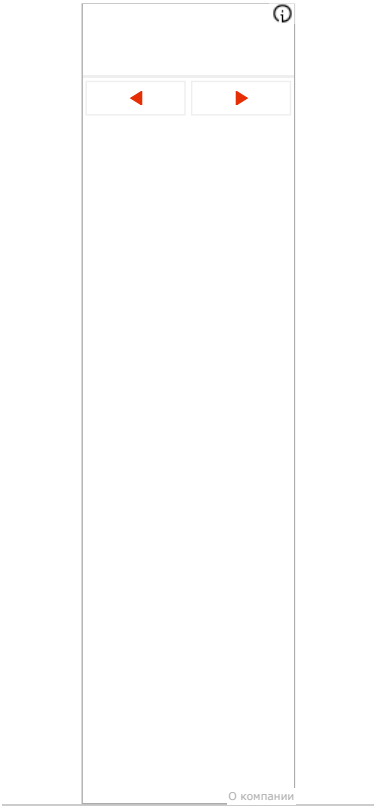
Теперь вообще про карту. Данная карта нам позволяет хранить данные, тип памяти у неё FLASH, который по сравнению с памятью типа EEPROM также является энергонезависимым, то есть при отключении питания данные никуда не пропадают, а остаются храниться. Также данная память имеет отличия, мы с ними познакомимся в процессе программирования. Одно из главных отличий то, что мы уже как в память EEPROM в данную память не можем записать один байт. Теоретически то конечно можем, но только запишутся туда либо только единички из нашего байта либо только нули в зависимости от типа FLASH — NOR или NAND. То есть прежде чем писать байт, нужно его стереть, а в силу организации данной памяти, стирать мы можем только блоками, вот и писать следовательно также только блоками. Но зато есть величайшее отличие от EEPROM — это цена. Она в разы дешевле, даже порой на порядки за одну хранящуюся единицу инфорамции (за мегабайт, за гигабайт). Поэтому у памяти FLASH как правило всегда гораздо больший объём информации.
Существуют 4 типа SD, но это мы изучим немного позднее.
Подключим данную карту пока в протеусе



Здесь всё просто. На самом деле не совсем так. Нужны ещё резисторы

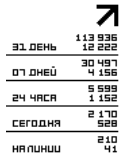


Данные резисторы нужны для того, чтобы обеспечить соответствующие уровни, так как карта питается от 3,3 вольт. Вообще по технической документации от 2,7 до 3,6 вольт.
Также в протеусе не указано, а на самом деле мы будем питать нашу карту от отдлеьного питания, поставив микросхему, преобразующую 5 вольт в 3,3 вольт.

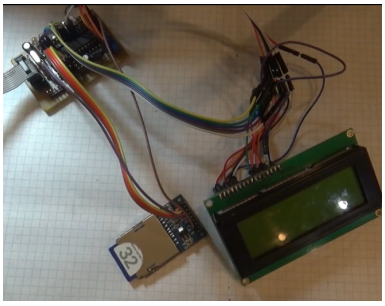


Рубрики

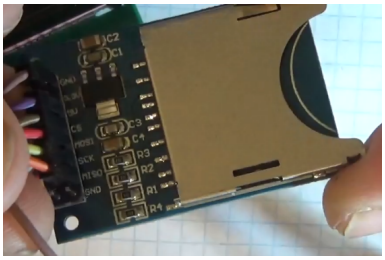
- 1-WIRE (3)
- ADC (6)
- DAC (4)
- GPIO (25)
- I2C (19)
- SPI (13)
- USART (8)
- Программирование AVR (131)
- Программирование PIC (6)
- Программирование STM32 (211)
- Тесты устройств и аксессуаров (1)



Вернее, мы не будем ничего ставить, а будем использовать готовый модуль, в котором уже всё установлено.
Также у нас подключен дисплей, как и на **прошлом занятии** по расширению функционала библиотеки дисплея.
Вот так у нас всё выглядит в практической схеме



Вот так вот выглядит модуль с держателем



Найти такой модуль можно везде, стоит он копейки. Тот модуль, который конектится по SDIO, стоит дороже. Мы видим также, что на модуле уже установлена микросхема для понижения напряжения до 3,3 вольта. А подключаем питание мы только на контакт 5 вольт, а на 3,3 не подключаем ничего

Также на модуле установлены все делители для уровней, то есть данный модуль рассчитан именно на подключение к 5-вольтовым устройствам.
А флеш-карту для тестов я откопал на 32 мегабайта, именно мегабайта а не гигабайта

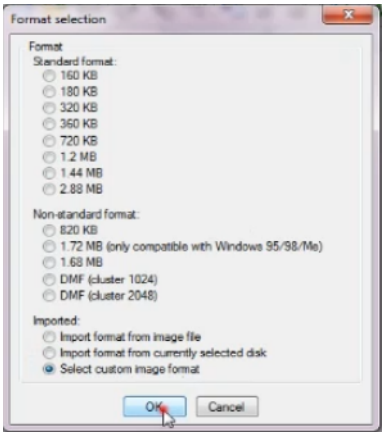
Данная флеш-карта была подарена вместе с каким-то фотоаппаратом и она нам лучше всего подойдёт для наших тестов, по крайней мере мы не будем думать, что тот или иной глюк у нас из-за слишком большого размера памяти на карте.
Код был весь взят также с прошлого занятия вместе с библиотекой дисплея, так как функцию, которую мы создали на прошлом уроке, мы будем очень активно использовать, только был конечно создан проект новый и назват соответственно **MYSD_SPI**.
Удалим ненужные строки, в main() у нас останется только во это

```
int main(void)
{
  unsigned int i;
  port_ini();
  LCD_ini(); //инициализируем
дисплей
  clearlcd();//очистим дисплей
  setpos(0,0);
  str_lcd("String 1");
  setpos(2,1);
  str_lcd("String 2");
  setpos(4,2);
  str_lcd("String 3");
  setpos(6,3);
  str_lcd("String 4");
  _delay_ms(2000);
  // for (i=0;i<=22;i++)
{str80_lcd(buffer2+i*20);_delay_ms(1
000);}
  while(1)
  {
  }
}
```

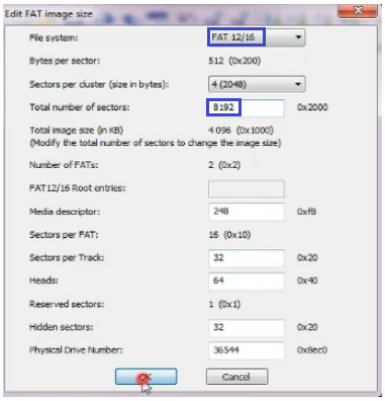
Так как мы посимвольно не бу выводить текст, то можно будет переменной обойтись типом char

```
unsigned char i;
```

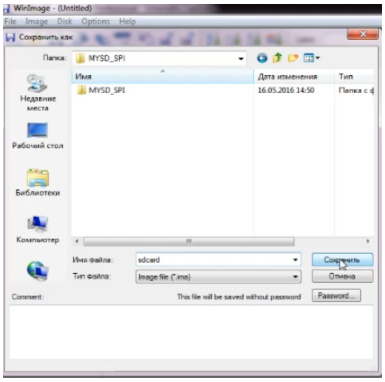
Теперь ещё один нюанс.
Чтобы нам работать с SD-картой в протеусе, нам мало добавить сам держатель с картой, необходимо также в его свойствах прикрепить файл образа флеш-карты.
Создать его не сложно. Одним из способов является создание с помощью программы WinImage.
Мы в ней стандартно создаём новый файл с помощью пункта меню File — > New. Выбираем в диалоге самый последний пункт и жмём "OK"



Для теста в протеусе нам вполне хватит размера 4 мегабайта, поэтому поменяем в следующем диалоге поле с номером секторов, а также выберем формат FAT12/16, потому что с 32-битной файловой системой немного другая специфика работы, и также нажмём "OK"

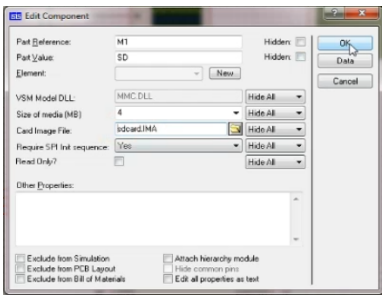


Вообще мы конечно можем оставить и FAT32, так как мы с файловой системой пока не работает, но в дальнейших частях занятия будет работа с файловой системой и мы будем именно работать с 12/16.
Затем мы сохраняем наш созданный файл с помощью пункта меню File -> Save As. И сохраняем мы его в ту папку, где у нас находится сохранённый проект протеуса. Назовём файл и нажмём "Сохранить"



Также затем нужно будет убедиться, что данный файл у нас получился не с аттрибутом "только для чтения" и после этого мы уже сможем его подключить в протеусе. Надо будет вручную вписать имя

файла, так как протеус требует какой-то свой формат и наш файл будет просто не виден



Путь нам никакой не нужен, так как файл у нас находится в папке с проектом. Жмём "ОК".
Инициализация шины нам не нужна, так как у нас SPI будет программный, с аппаратным флеш-карты работают корректно не все, то нам не надо будет использовать никаких регистров. Аппаратный конечно, лучше, но чтобы уяснить работу протокола досконально, надо ещё поработать и с программным, то есть подрыгать ножками портов. Вообще, глядя на схему, может показаться, что у нас всё аппаратно, так как я именно такие ножки выбрал, это потому, что я просто так выбрал, чтобы впоследствии когда-то может быть кто-то попытается всё-таки поработать с аппаратной реализацией шины.
Добавим макроподстановки для ножек порта

```
#include "main.h"
#define MOSI 3
#define MISO 4
#define SCK 5
#define SS 2
```

Добавим код для инициализации ножек в функцию инициализации портов

```
void port_ini(void)
{
  PORTD=0x00;
  DDRD=0xFF;
  PORTB|=(1<<SS)|(1<<MISO)|(1<<MOSI);
  DDRB|=(1<<SS)|(1<<MOSI)|(1<<SCK);
}
```

Мы оставляем на вход ножку MISO, так как по умолчанию все биты в регистре равны нулю, и мы его просто не трогает. Также мы включаем сразу высокий уровень в MOSI и SS, а к MISO подтягиваем резистор.
Напишем функцию передачи байта по шине SPI

```
void SPI_SendByte (unsigned char
byte)
{
}
```

Добавим переменную для цикла и сам цикл

```
void SPI_SendByte (unsigned char
byte)
```

```
{
  unsigned char i;
  for (i=0;i<8;i++) //движемся по
битам байта
  {
  }
}
```

Я думаю, понятно почем мы считаем до 8, так как битов мы передаём именно 8.
Ну и начнём их передавать потихоньку. Проверим сначала самый левый бит, выделив его из всего байта маскированием, и, если он у нас равен 1, то выставим 1 и на шине MOSI, а если 0 — то не трогаем шину

```
for (i=0;i<8;i++) //движемся по
битам байта
{
  if ((byte&0x80)==0x00)//проверяем
левый бит
  PORTB&=~(1<<MOSI); //если 0, то
выставляем 0 на шине
  else PORTB|=(1<<MOSI); //если 1,
то выставляем 1
```

Затем мы сдвинем наш байт влево на 1, чтобы старшим (левым) битом у нас стал следующий бит

```
else PORTB|=(1<<MOSI); //если 1, то
выставляем 1
byte<<=1; //сдвигаем влево, в
сторону старшего для проверки
следующего бита
```

Дрыгнем ножкой SCK, чтобы сформировать импульс тактирования

```
    byte<<=1; //сдвигаем влево, в
сторону старшего для проверки
следующего бита
    PORTB|=(1<<SCK); //фронт на
ножке SCK
    asm("nop"); //1 такт подождём
    PORTB&=~(1<<SCK); //спад на
ножке SCK
  }
}
```

Я думаю, для одной части урока мы уже сделали достаточно, а код для приёма байта по шине мы будем писать уже в **следующей части** урока.

Техническая документация на Secure Digital

Программатор, модуль SD и дисплей можно приобрести здесь:
Программатор (продавец надёжный)
USBASP USBISP 2.0
Модуль карты SD SPI
Дисплей LCD 20×4

Смотреть ВИДЕОУРОК
(нажмите на картинку)

Post Views: 481

‹ AVR Урок 32. Дисплей LCD 20×4. Расширяем функционал
AVR Урок 33. SPI. Карта SD. FAT. Часть 2 ›

Добавить комментарий

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

Комментарий

Имя *

E-mail *

Сайт

девять + = **11**

Отправить комментарий

Наверх