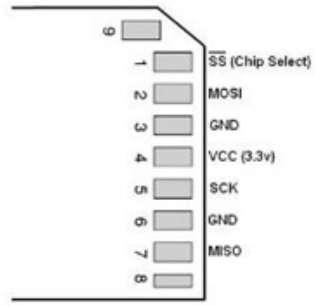
My Controller

Изучаем микроконтроллеры STM32

STM32 SD Card. Подключение

Распиновка карты памяти выглядит следующим образом:



Назначение выводов SD Card

вывод 1 (SS) – выбор кристалла (выбор при подаче нуля);

вывод 2 (MOSI) – для приема данных картой ;

выводы 3,6(GND) – общий вывод питания;

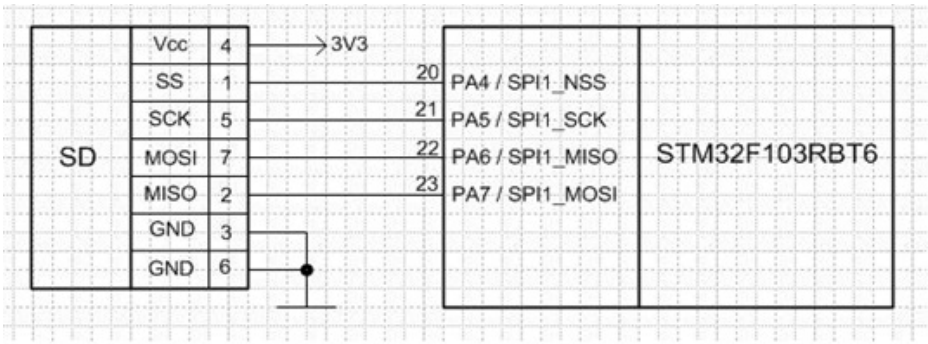
вывод 4(Vcc) – плюс питания (+3.3V);

вывод 5(SCK) – синхроимпульсы (подает контроллер);

вывод 7(MISO) – выдача данных из карты;

Как видим, для подключения к контроллеру необходимо 4 вывода последнего: три вывода используются для работы SPI и один для включения/выключения карты.

Я подключил карту памяти к контроллеру STM32F103RBT6 к первому модулю SPI. Ниже привожу часть схемы, на которой изображено это подключение:



Чтобы “общаться” с картой памяти, необходимо настроить интерфейс SPI. Для начала настроим его для работы без использования прерываний и DMA.

```
//*****************************************************************************************
//function  инициализация  SPI1                                                          //
//argument  none                                                                         //
//return    none                                                                         //
//*****************************************************************************************
void spi_init(void)
```

```
{
  RCC->APB2ENR |=  RCC_APB2ENR_AFIOEN;//включить тактирование альтернативных функций
  RCC->APB2ENR |=  RCC_APB2ENR_IOPAEN;//включить тактирование порта A

  //вывод управления SS: выход двухтактный, общего назначения,50MHz
  GPIOA->CRL   |=  GPIO_CRL_MODE4;    //
  GPIOA->CRL   &= ~GPIO_CRL_CNF4;     //
  GPIOA->BSRR   =  GPIO_BSRR_BS4;     //

  //вывод SCK: выход двухтактный, альтернативная функция, 50MHz
  GPIOA->CRL   |=  GPIO_CRL_MODE5;    //
  GPIOA->CRL   &= ~GPIO_CRL_CNF5;     //
  GPIOA->CRL   |=  GPIO_CRL_CNF5_1;   //

  //вывод MISO: вход цифровой с подтягивающим резистором, подтяжка к плюсу
  GPIOA->CRL   &= ~GPIO_CRL_MODE6;    //
  GPIOA->CRL   &= ~GPIO_CRL_CNF6;     //
  GPIOA->CRL   |=  GPIO_CRL_CNF6_1;   //
  GPIOA->BSRR   =  GPIO_BSRR_BS6;     //

  //вывод MOSI: выход двухтактный, альтернативная функция, 50MHz
  GPIOA->CRL   |=  GPIO_CRL_MODE7;    //
  GPIOA->CRL   &= ~GPIO_CRL_CNF7;     //
  GPIOA->CRL   |=  GPIO_CRL_CNF7_1;   //

  //настроить модуль SPI
  RCC->APB2ENR |= RCC_APB2ENR_SPI1EN; //подать тактирование
  SPI1->CR2     = 0x0000;             //
  SPI1->CR1     = SPI_CR1_MSTR;       //контроллер должен быть мастером,конечно
  SPI1->CR1    |= SPI_CR1_BR;         //для начала зададим маленькую скорость
  SPI1->CR1    |= SPI_CR1_SSI;
  SPI1->CR1    |= SPI_CR1_SSM;
  SPI1->CR1    |= SPI_CR1_SPE;        //разрешить работу модуля SPI
}
```

Пояснения к тексту программы.

В начале подаем тактирование порта A, к которому будет подключаться карта, и тактирование альтернативных функций, т.к. к выводам порта подключено переферийное устройство – модуль SPI.

Дальше выполняем настройку выводов:

- вывод PA4 конфигурируем как двухтактный выход общего назначения для управления выбором карты;так как для выбора карты нужен ноль, мы его пока установим в единицу, а при обращении к карте будем его изменять;
- вывод PA5 конфигурируем как двухтактный выход для альтернативной функции, он будет использоваться модулем для подачи тактового сигнала SCK;
- вывод PA6 конфигурируем как вход цифровой с подтягивающим резистором для приема данных с карты памяти (MISO);
- вывод PA7 конфигурируем как двухтактный выход для альтернативной функции, через него модуль SPI будет выдавать данные (MOSI).

Ну и наконец выполняем настройку модуля SPI:

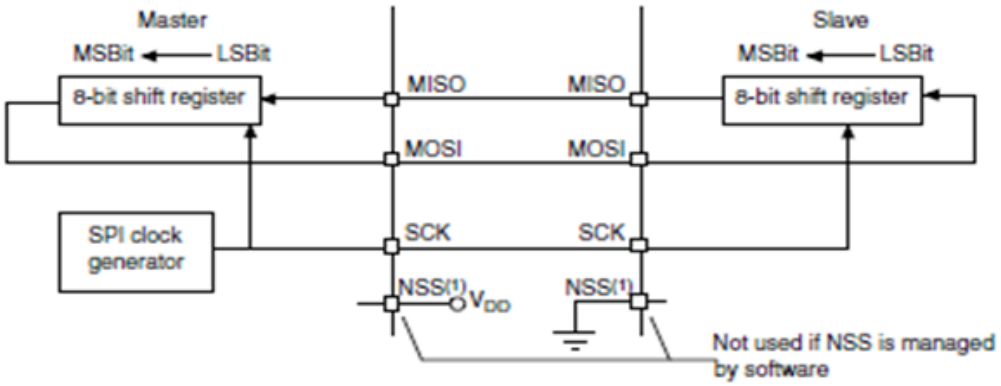
- подаем тактирование;
- включаем работу в режиме ведущего (мастера);
- задаем значение делителя скорости обмена (я задал наименьшую скорость, но можно увеличить, этим займемся дальше);
- разрешаем работу модуля

Остальные разряды управляющих регистров сброшены, настройки которые при этом получаются подходят для работы с картой памяти.

Функции передачи данных

При передаче данных автоматически осуществляется прием. Такова особенность интерфейса SPI.

Небольшая схема, приведенная ниже, хорошо это показывает:



Ведущий передает данные в регистр сдвига ведомого, а тот выталкивает свои данные ведущему. Поэтому правильно будет назвать функцию для передачи данных функцией обмена: передали байт, и приняли одновременно.

Назовем эту функцию spi_send. Выглядит она так:

```
//**************************************************************************************************
//function  обмен данными по SPI1                                                                  //
//argument  передаваемый байт                                                                      //
//return    принятый байт                                                                          //
//**************************************************************************************************
uint8_t spi_send (uint8_t data)
{
  while (!(SPI1->SR & SPI_SR_TXE));      //убедиться, что предыдущая передача завершена
  SPI1->DR = data;                       //загружаем данные для передачи
  while (!(SPI1->SR & SPI_SR_RXNE));     //ждем окончания обмена
  return (SPI1->DR);                     //читаем принятые данные
}
```

Возможно, строка №8 лишняя, но я решил "перестраховаться".

В строке №9 выполняется загрузка данных, что запускает процесс обмена.

Когда необходимо принять байт без передачи данных ведомому, можно воспользоваться следующей функцией:

```
//**************************************************************************************************
//function  прием байт по SPI1                                                                     //
//argument  none                                                                                   //
//return    принятый байт                                                                          //
//**************************************************************************************************
uint8_t spi_read (void)
{
  return spi_send(0xff);                 //читаем принятые данные
}
```

Можно выполнить проверку правильности настройки модуля SPI и его работоспособность без подключения внешнего устройства. Для этого необходимо замкнуть вывода MOSI и MISO, выполнить инициализацию и передать/принять байт функцией spi_send. Если передаваемый байт будет равен принятому, значит все нормально.

Итак, мы написали функции, необходимые для инициализации модуля SPI и работы с ним. Следующий этап — использовать их для связи с картой памяти. Об этом в следующей статье.

Рубрика: Карта памяти SD

Комментарии (22) на “STM32 SD Card. Подключение”

- *barteroff:*
  09.02.2012 в 15:34

  Огромное спасибо, помогли разобраться с подключение карты памяти!

  Ответить
- *vrr:*
  09.02.2012 в 23:49

  Мне пришлось перед влючением мастера (SPI1->CR1 = SPI_CR1_MSTR;) активизировать выход SS путем установки бита SSOE (SPI1->CR2 |= SPI_CR2_SSOE;) иначе при отладке выполняя активацию SPI (SPI1->CR1 |= SPI_CR1_SPE;) у меня слетал режим мастера. В итоге вышло: