Современные 32-разрядные ARM-микроконтроллеры серии STM32: последовательный интерфейс SPI

Олег Вальпа

Приведено описание последовательного интерфейса SPI 32-разрядных ARM-микроконтроллеров серии STM32 от компании STMicroelectronics. Рассмотрена архитектура, состав и назначение регистров конфигурирования SPI, а также предложены примеры программ для его инициализации и работы.

ВВЕДЕНИЕ

Интерфейс SPI (Serial Peripheral Interface — последовательный периферийный интерфейс) является высокоскоростным синхронным последовательным интерфейсом. Он обеспечивает обмен данными между микроконтроллером и различными периферийными устройствами, такими как АЦП, ЦАП, цифровые потенциометры, карты памяти, другие микросхемы и микроконтроллеры.

ОПИСАНИЕ ИНТЕРФЕЙСА SPI

Микроконтроллер STM32 [1] содержит два интерфейса SPI, которые обеспечивают передачу данных на частотах до 18 МГц. Один интерфейс SPI расположен на низкоскоростной шине APB1, работающей на тактовой частоте до 36 МГц, а другой — на высокоскоростной шине периферийных устройств APB2, которая работает на тактовой частоте до 72 МГц. Для увеличения эффективности передачи данных в микроконтроллере выделено два канала DMA.

По интерфейсу SPI можно связать ведущий микроконтроллер с одним или несколькими ведомыми устройствами. Схема подключения устройств по интерфейсу SPI показана на рисунке 1.

Одно из устройств должно быть определено ведущим (мастер), а остальные — ведомыми (подчиненные). Связь между устройствами осуществляется с помощью следующих линий связи:

- MOSI — выход данных для ведущего или вход данных для ведомого устройства;
- MISO — вход данных для ведущего или выход данных для ведомого устройства;
- SCK — сигнал общей синхронизации интерфейса.

Кроме того, ведущее устройство формирует один или несколько сигналов SS (slave select) для выбора ведомых устройств. При этом количество формируемых сигналов соответствует количеству ведомых устройств.

На рисунке 2 приведена функциональная схема интерфейса SPI для двух

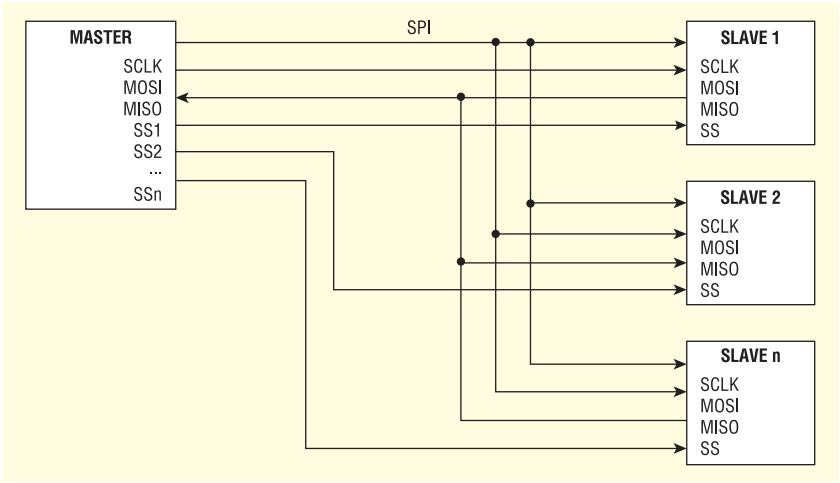

Рис. 1. Схема подключения устройств по интерфейсу SPI

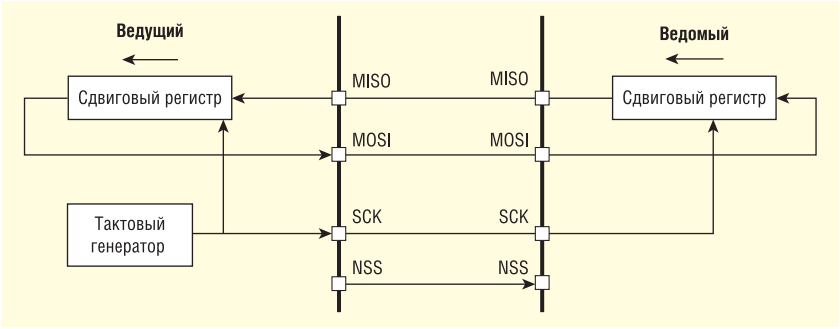

Рис. 2. Функциональная схема интерфейса SPI

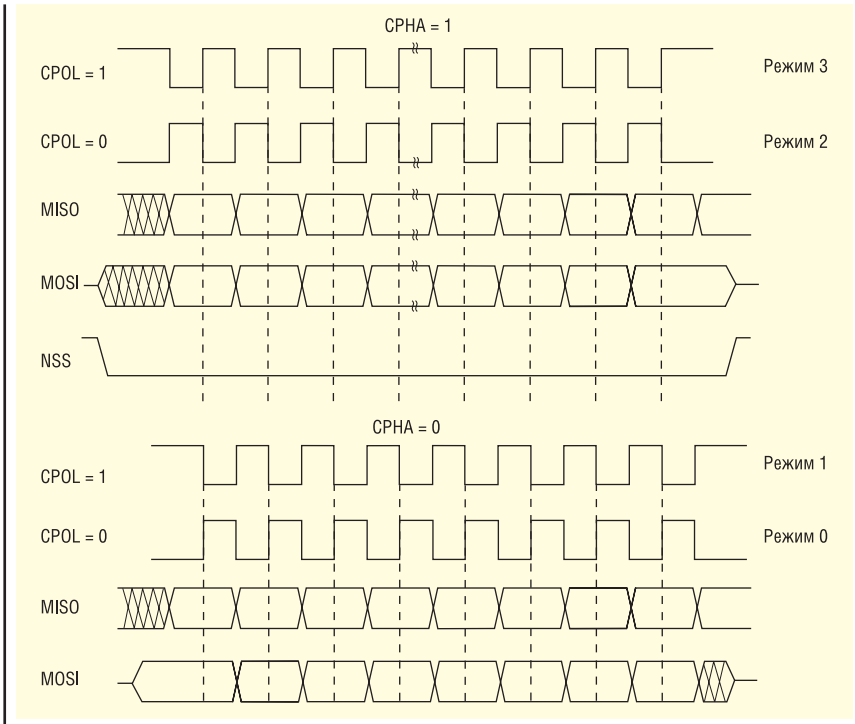

Рис. 3. Диаграмма режимов работы интерфейса SPI

подключенных друг к другу устройств, одно из которых определено как ведущее, а второе — как ведомое.

Как видно из этой функциональной схемы, передача данных осуществляется посредством линий MOSI и MISO. Сдвиговые регистры ведущего и ведомого устройства объединяются линиями связи в единый сдвиговый регистр. Процессом передачи данных управляет ведущее устройство (мастер), формируя одновременно тактовые импульсы через линию SCK. Одновременно с передачей данных от ведущего к ведомому устройству, происходит прием данных ведущим устройством от ведомого по кольцу. Таким образом, за один полный цикл сдвига всех разрядов регистра происходит обмен данными между двумя устройствами.

Существует четыре режима передачи данных по SPI, которые определяются полярностью и фазой тактового сигнала. Отличие режимов заключается в том, что активным уровнем сигнала синхронизации может быть единичный или нулевой потенциал, а запись данных может производиться по фронту или спаду импульса данного синхросигнала. Эти режимы интерфейса обозначаются цифрами 0, 1, 2 и 3. На рисунке 3 — диаграмма всех перечисленных режимов работы интерфейса SPI.

Микроконтроллер STM32 позволяет для каждого интерфейса SPI задать полярность и фазу тактового сигнала, определяя тем самым режим его работы. Кроме того, для микроконтроллера можно установить формат передачи данных 8-разрядными или 16-разрядными словами и определить порядок передачи данных — старшим или младшим битом вперед. Это позволяет микроконтроллеру с помощью обоих интерфейсов SPI обмениваться информацией с любыми другими SPI-устройствами.

АРХИТЕКТУРА SPI

Рассмотрим внутреннюю архитектуру SPI микроконтроллера STM32, которая представлена на рисунке 4.

Регистр сдвига представляет собой основной регистр, через который передаются и принимаются данные. Если интерфейс SPI работает в режиме ведущего устройства, то вход этого сдвигового регистра соединен с выводом MISO, а выход — с выводом MOSI. В режиме ведомого устройства происходит обратное переключение, которое регулирует блок управления.

Для передачи данных их необходимо записать в регистр передатчика. Принятые данные читаются из регистра приемника. Для программы существует один регистр с именем SPI_DR. При чтении этого регистра происходит обращение к регистру приемника, а при записи — к регистру передатчика.

Скорость обмена по SPI определяет блок генератора скорости, который задает частоту следования тактовых импульсов. Для этого предназначены разряды BR0, BR1 и BR2 регистра SPI_CR1. Три разряда предполагают наличие восьми значений скорости. Таким образом, скорость обмена данными по интерфейсу SPI для микроконтроллера STM32 с тактовой частотой 24 МГц может изменяться от 24 МГц/2=12 Мбод до 24 МГц/8=3 Мбод.

Вывод NSS (Negativ Slave Select) предназначен для выбора ведомого устройства. Этот сигнал можно уста-

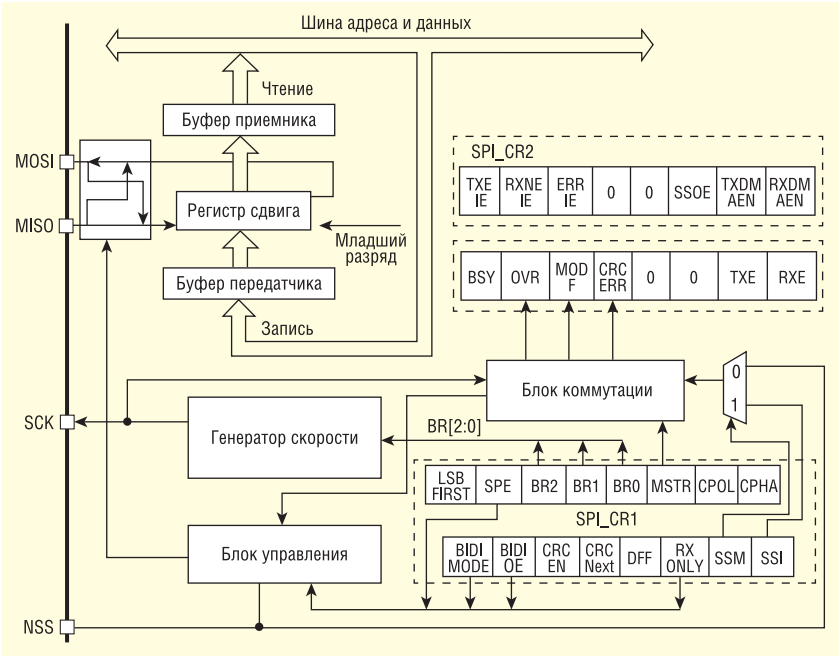

Рис. 4. Архитектура SPI микроконтроллера STM32

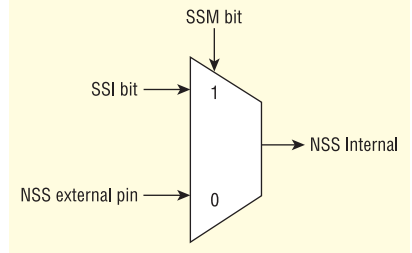

Рис. 5. Функциональная схема формирования сигнала NSS

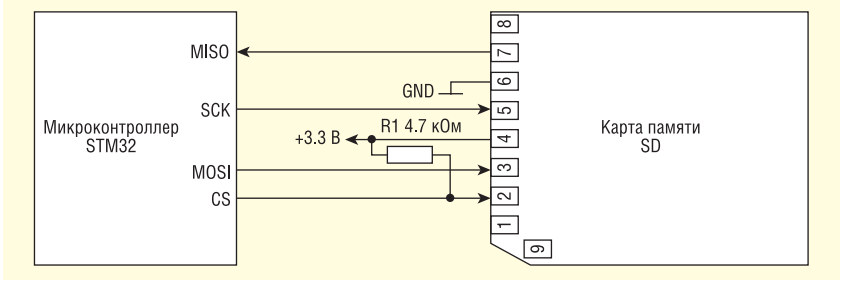

Рис. 6. Схема подключения карты памяти

новить программно или аппаратно. Функциональная схема формирования сигнала NSS отображена на рисунке 5.

На этой схеме показано формирование сигнала управления NSS Internal из внешнего сигнала NSS external pin или программного сигнала SSI bit. Выбор источника сигнала NSS производится с помощью программного разряда SSM. Разряды SSM и SSI принадлежат регистру управления SPI_CR1.

Использование сигнала NSS для работы в режиме ведущего устройства имеет специфические особенности. Чтобы интерфейс SPI оставался в режиме мастера, необходимо удерживать сигнал NSS Internal в состоянии логической единицы. Если сигнал NSS Internal станет равным нулю, интерфейс сразу перейдет в режим ведомого устройства. Данная особенность применяется для организации мультимастерных систем, в которых арбитр может управлять сигналом NSS через внешний вывод. Изменение уровня сигнала на этом выводе с логической единицы на уровень логического нуля позволяет переключать интерфейс между режимами ведущего и ведомого устройства.

Если мультимастерный режим не требуется, следует программно блокировать этот вывод, записав логическую единицу в разряды SSI и SSM. После чего данный вывод можно использовать по другому назначению.

В режиме мастера можно сконфигурировать вывод NSS для аппаратной подачи сигнала SS ведомому устройству. Для этого используется разряд SSOE регистра SPI_CR2.

Для организации высокоскоростного обмена данными каждый SPI-интерфейс содержит два канала DMA. Один канал DMA предназначен для передачи данных, а второй — для сохранения принимаемых данных в памяти микроконтроллера. Использование DMA позволяет обмениваться данными на высокой скорости в двух направлениях под аппаратным управлением.

Кроме стандартных функций, интерфейс SPI микроконтроллера STM32 содержит два аппаратных блока вычисления циклического избыточного кода CRC. Один блок CRC используется для передаваемых данных, а второй — для принимаемых данных. Оба блока могут вычислять циклический избыточный код для 8- и 16-битных данных. Эта функция особенно необходима при подключении к SPI карт памяти типа MMC или SD. Схема подключения карты памяти к микроконтроллеру через интерфейс SPI приведена на рисунке 6.

На этой схеме также показано расположение контактов на карте памяти и их назначение.

ОПИСАНИЕ РЕГИСТРОВ

Для работы с интерфейсом SPI в микроконтроллере STM32 имеются специальные регистры. Формат этих регистров с названием входящих в них разрядов представлен в таблице.

Рассмотрим регистры, необходимые для работы SPI. К ним относятся:

Таблица. Формат регистров интерфейса SPI

Сдвиг	Регистр	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0×00	SPI_CR1	Резерв																BIDMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0×04	SPI_CR2	Резерв																								TXEE	RXNEIE	ERRIE	Резерв		SSDE	TXDMAEN	RXDMAEN
	Исх. значение																									0	0	0			0	0	0
0×08	SPI_CR	Резерв																								BSY	OVR	MODE	CRCERR	Резерв		TXE	RXNE
	Исх. значение																									0	0	0	0			1	0
0×0C	SPI_DR	Резерв																DR[15:0]															
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0×10	SPI_CRCPR	Резерв																CRCPOLY[15:0]															
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0×14	SPI_RXCRCR	Резерв																RXCRC[15:0]															
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0×18	SPI_TXCRRCR	Резерв																TXCRC[15:0]															
	Исх. значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Листинг 1

```
void Init_SPI(void)
{
// Включение тактирования
// Включить тактирование альтернативных функций
RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;
// Включить тактирование порта A
RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
// Настройка выводов согласно выполняемым функциям:
// Вывод управления NSS: выход двухтактный, общего назначения, 50 МГц
GPIOA->CRL |= GPIO_CRL_MODE4;
GPIOA->CRL &= ~GPIO_CRL_CNF4;
GPIOA->BSRR = GPIO_BSRR_BS4;
// Вывод SCK: выход двухтактный, альтернативная функция, 50 МГц
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, 50 МГц
GPIOA->CRL |= GPIO_CRL_MODE7;
GPIOA->CRL &= ~GPIO_CRL_CNF7;
GPIOA->CRL |= GPIO_CRL_CNF7_1;
// Настройка SPI
RCC->APB2ENR |= RCC_APB2ENR_SPI1EN; // Включить тактирование
SPI1->CR1 = 0x0000; // Очистить первый управляющий регистр
SPI1->CR1 |= SPI_CR1_DFF // Бит11 Формат данных 0-8бит 1-16бит
SPI1->CR1 |= SPI_CR1_SSM; // Бит9 SSM - выбирает источник сигнала NSS (0 – с внешнего вывода, 1 – программно);
SPI1->CR1 |= SPI_CR1_SSI; // Бит8 SSI - если SSM = 1 определяет значение NSS;
SPI1->CR1 |= SPI_CR1_LSBFIRST; // Бит7 LSBFIRST - задает способ передачи (0 - старшим, 1 – младшим разрядом
вперед) ;
SPI1->CR1 |= SPI_CR1_SPE; // Бит6 SPE - работа SPI (1 - вкл. 0 - откл.)
Бит3-5 BR[2:0] – делитель скорости обмена fPCLK/x (000:2, 001:4, 010:8, 011:16, 100:32, 101:64, 110:128,
111:256)
SPI1->CR1 |= SPI_CR1_BR_0 | SPI_CR1_BR_1 | SPI_CR1_BR_2; // Задать скорость fPCLK/x
SPI1->CR1 |= SPI_CR1_MSTR; // MSTR - делает модуль ведущим(1)/ведомым(0);
SPI1->CR1 |= SPI_CR1_CPOL; // CPOL - задает полярность тактового сигнала;
SPI1->CR1 |= SPI_CR1_CPHA; // CPHA - задает фазу тактового сигнала 0-\ 1-/;
SPI1->CR2 = 0x0000; // Очистить второй управляющий регистр
}
```

- SPI_CR1 — первый управляющий регистр;
- SPI_CR2 — второй управляющий регистр;
- SPI_SR — регистр статуса;
- SPI_DR — регистр данных;
- SPI_CRCPR — регистр, содержащий полином для вычисления CRC;
- SPI_RXCRCR — регистр, содержащий CRC принятых данных;
- SPI_TXCRCR — регистр, содержащий CRC передаваемых данных.

Некоторые из этих регистров используются для работы в режиме I²S.

Регистр SPI_CR1 является первым управляющим регистром интерфейса SPI. Он имеет следующие управляющие разряды:

- 0 CPHA задает фазу тактового сигнала;
- 1 CPOL устанавливает полярность тактового сигнала;
- 2 MSTR назначает режим работы интерфейса (0 — ведомый, 1 — ведущий);
- 5...3 0BR [2:0] задают скорость обмена (000 — fPCLK/2, 001 — fPCLK/4, 010 — fPCLK/8, 011 — fPCLK/16, 100 — fPCLK/32, 101 — fPCLK/64, 110 — fPCLK/128, 111 — fPCLK/256);
- 6 SPE управляет интерфейсом (0 — отключает, 1 — включает);
- 7 LSBFIRST задает направление передачи (0 — младшим разрядом вперед, 1 — старшим разрядом вперед);

- 8 SSI определяет значение NSS при SSM = 1;
- 9 SSM выбирает источник сигнала NSS (0 — с внешнего вывода, 1 — программно от разряда SSI);
- 10 RX ONLY совместно с битом BIDIMODE определяет направление передачи в однонаправленном режиме;
- 11 DFF определяет формат данных (0–8 бит, 1–16 бит);
- 12 CRCNEXT управляет передачей кода CRC (0 — данные, 1 — CRC);
- 13 CRCEN регулирует аппаратное вычисление CRC (0 — запрещено, 1 — разрешено). Для корректной операции этот бит должен записываться только при отключенном интерфейсе SPI, когда SPE = 0;
- 14 BIDIOE совместно с битом BIDIMODE управляет двунаправленным режимом работы интерфейса (0 — прием, 1 — передача);
- 15 BIDIMODE управляет двунаправленным режимом работы интерфейса (0 — двухпроводный однонаправленный режим, 1 — однопроводной двунаправленный режим).

SPI_CR2 является вторым управляющим регистром интерфейса SPI и имеет следующие разряды, которые управляют:

- 0 RXDMAEN — запросом DMA для приемника (0 — запрещает, 1 — разрешает);
- 1 TXDMAEN – запросом DMA для передатчика (0 — запрещает, 1 — разрешает);
- 2 SSOE — сигналом NSS в режиме мастера (0 — запрещает, 1 — разрешает);
- 4...3 — зарезервированы;
- 5 ERRIE — прерыванием в случае ошибки (0 — запрещает, 1 — разрешает);
- 6 RXNEIE — прерыванием приема данных (0 — запрещает, 1 — разрешает);
- 7 TXEIE — управляет прерыванием передачи данных (0 — запрещает, 1 — разрешает);
- 15...8 — зарезервированы.

Регистр статуса SPI_SR включает в себя следующие разряды:

- 0 RXNE устанавливается, если в буфере приемника есть принятые данные;
- 1 TXE — устанавливается, если буфер передатчика пуст и готов принять новые данные;
- 2, 3 — зарезервированы;
- 4 CRCERR устанавливается при ошибке CRC при приеме данных;
- 5 MODF устанавливается, когда в режиме мастера к сигналу NSS прикладывается низкий потенциал;
- 6 OVR — флаг переполнения, устанавливается при приеме новых данных, если предыдущие не были прочитаны;
- 7 BSY — флаг занятости, устанавливается, если интерфейс занят обменом данных или буфер данных передатчика не пустой.

Регистр данных SPI_DR состоит из 16 разрядов данных. В этот регистр данные записываются для передачи и читаются из него при приеме.

Регистры SPI_CRCPR, SPI_RXCRCR и SPI_TXCRCR используются для вычисления контрольной суммы CRC для обнаружения ошибок при приеме и передаче данных.

Более подробное описание назначения всех регистров USART и их разрядов можно найти в источнике [2].

ПРОГРАММИРОВАНИЕ SPI

Рассмотрим процедуры инициализации и работы с интерфейсом SPI.

Для настройки интерфейса SPI в режим ведущего устройства необходимо выполнить следующие действия:

1. Задать скорость обмена с помощью разрядов BR [2:0] регистра SPI_CR1. 2. Задать полярность и фазу тактового сигнала с помощью разрядов CPOL и CPHA регистра SPI_CR1. 3. Задать формат данных (8 или 16 бит) с помощью разряда DFF регистра SPI_CR1.

4. Задать направление передачи данных (первым или последним битом вперед) с помощью разряда LSBFIRST регистра SPI_CR1.

5. Если вывод NSS будет использоваться для выбора устройства, следует установить разряд SSOE регистра SPI_CR1.

6. Установить разряды MSTR и SPE регистра SPI_CR1 для перевода интерфейса SPI в режим ведущего и его включения.

В листинге 1 приведена функция инициализации интерфейса SPI в соответствии с описанным выше алгоритмом.

После вызова и выполнения данной функции настройка интерфейса будет завершена.

Для передачи данных через интерфейс SPI необходимо загрузить передаваемые данные в регистр SPI_DR. Эта процедура выполняется с помощью команды:

```
SPI1->DR = data_tx; // Загрузить данные в регистр для передачи через SPI.
```

Окончание передачи контролируется проверкой разряда TXE регистра SPI_SR:

```
while (!(SPI1->SR SPI_SR_TXE)); // Ожидание окончания передачи.
```

Одновременно с передачей происходит прием данных в регистр SPI_DR. Принятые данные считываются из регистра данных с помощью команды:

```
data_rx = SPI1->DR; // Считать принятые данные
```

Для проверки работоспособности интерфейса SPI в режиме мастера достаточно соединить выводы MISO и MOSI между собой и сравнить переданные данные с полученными. Если они совпадают, это значит, что интерфейс работает правильно.

Перед передачей и приемом данных необходимо сформировать сигналы выбора для того устройства, с которым будет производиться обмен. Если ведомое устройство одно, то можно использовать сигнал выбора NSS. Если же ведомых устройств несколько, то придется для каждого из них формировать индивидуальный сигнал выбора. В качестве источников таких сигналов могут выступать свободные выводы портов GPIO. Например, если для формирования сигнала выбора используется вывод порта A4, то активация и дезактивация данного сигнала осуществляется с помощью следующих команд:

```
GPIOA->BSRR = GPIO_BSRR_BR4; // Установить сигнал выбора SS (A4) = 0.
GPIOA->BSRR = GPIO_BSRR_BS4; // Установить сигнал выбора SS (A4) = 1.
```

Между этими командами следует расположить описанные выше команды передачи и чтения через интерфейс.

Литература:

1. https://www.st.com.
2. http://www.st.com/web/en/resource/technical/document/reference_manual/CD00246267.pdf.

СНУ

* Статья перепечатана из журнала «Современная электроника», № 1, 2014 г., с разрешения редакции, тел. +7 (495) 232-00-87, www.soel.ru