Интерфейс SPI

У последовательного интерфейса SPI есть 2 основные функции - либо поддержка протокола/интерфейса SPI [2], либо audio-протокола I2S. По умолчанию выбрана функция SPI. Можно программно переключить режим из SPI в I2S.

Примечание: описание в этой статье - перевод раздела документации "28. Serial peripheral interface (SPI)" из даташита [1]. Вся приведенная информация об SPI относится к семейству микроконтроллеров STM32F4xx, если специально не указано нечто другое.

SPI. SPI позволяет реализовать полудуплексный или полнодуплексный, синхронный обмен последовательными данными с внешними устройствами. Этот интерфейс можно сконфигурировать либо как в режиме главного устройства (master), либо как в режиме подчиненного устройства (slave). В режиме главного устройства SPI формирует выходной сигнал тактов (SCK) для внешнего подчиненного устройства. Интерфейс позволяет также работать в конфигурации с несколькими главными устройствами на шине (конфигурация multimaster).

Возможный диапазон применений SPI очень широкий, включая двунаправленную передачу данных или надежную связь между устройствами, защищенную проверкой контрольной суммы (CRC).

I2S. I2S это также последовательный синхронный интерфейс. Он может работать по 4 различным стандартам, включая I2S Philips, стандарты с выравниванием данных MSB и LSB, и стандарт PCM. Возможна работа как slave или как master-устройства в полнодуплексном режиме (с использованием 4 выводов) или в полудуплексе (используется 3 вывода). Частота тактов от master может быть предоставлена для slave-устройства, когда I2S сконфигурирован как главное устройство.

Предупреждение: поскольку некоторые сигналы SPI1 и SPI3/I2S3 могут отображаться на выводы с функциями интерфейса JTAG (SPI1_NSS на JTDI, SPI3_NSS/I2S3_WS на JTDI и SPI3_SCK/I2S3_CK на JTDO), то для работы с этими сигналами можно поступить в следующими способами:

– отобразить сигналы SPI/I2S на другие выводы корпуса микроконтроллера

– запретить использование JTAG, и использовать для отладки приложения вместо него интерфейс SWD - до того, как были сконфигурированы выводы, перечисленные как ножки ввода/вывода SPI, или

– запретить оба интерфейса отладки, и JTAG, и SWD (применимо для уже отлаженных приложений).

Для дополнительной информации по ножкам JTAG/SWD см. секцию "8.3.2: I/O pin multiplexer and mapping" даташита [1] (или секцию "Мультиплексирование и отображение ножек I/O" статьи [4]).

Основные функции SPI

- *Полнодуплексная синхронная передача по 3 сигнальным линиям.*
- *Симплексная синхронная передача по 2 сигнальным линиям, с двунаправленной линией или однонаправленной линией данных.*
- *Формат фрейма 8 или 16 бит.*
- *Работа в режиме master или slave.*
- *Возможность поддержки режима Multimaster.*
- *8 режимов прескалера скорости для master (максимальная частота SCK тактов fPCLK/2).*
- *Максимальная частота тактов режима slave равна fPCLK/2 max.*
- *Ускоренный обмен для режимов master и slave.*
- *Аппаратное управление выборками NSS, или программное, доступное для обоих режимов master и slave. Благодаря этому можно динамически менять функционал master/slave.*
- *Программируемая полярность и фаза для тактов (CPOL, CPHA).*
- *Программируемый порядок следования старшинства бит - либо старшим битом вперед (MSB-first), либо младшим (LSB-first).*
- *Отдельные флаги для передачи и приема с возможностью прерываний.*
- *Флаг статуса занятости шины (SPI bus busy).*
- *Режим SPI TI.*
- *Функция аппаратного CRC для надежного обмена:*
  - *Значение CRC может быть передано как последний байт в режиме Tx.*
  - *Автоматическая проверка CRC для последнего принятого байта.*
- *Флаги ошибок Master mode fault, overrun и CRC error, с возможностью генерации прерывания.*
- *1-байтный буфер передачи и приема с возможностью DMA: обработка запросов передачи и приема (Tx request и Rx request).*

Основные функции I2S

- *Полнодуплексный обмен данными.*
- *Полудуплексный обмен (только передатчик или только приемник).*
- *Работа в режимах master или slave.*
- *8-битный программируемый линейный прескалер, чтобы достичь точных частот выборок звука (от 8 кГц до 192 кГц).*
- *Формат данных может быть 16 бит, 24 бита или 32 бита.*
- *Фрейм пакета, передаваемого по аудиоканалу, фиксирован на 16 бит (фрейм данных 16 бит) или на 32 бита (фрейм данных 16, 24, 32 бита).*

- *Программируемая полярность тактов (устойчивое состояние, steady state).*
- *Флаг недогрузки (underrun) в режиме передачи slave, флаг переполнения (overrun) в режиме приема (master и slave), флаг ошибки кадра (Frame Error) в режиме приема и передачи (только slave).*
- *16-битный регистр для передачи и приема с одним регистром данных для обоих каналов (левый и правый).*
- *Поддерживаемые протоколы I2S:*
  - *I2S Phillps standard*
  - *MSB-justified standard (left-justified)*
  - *LSB-justified standard (right-justified)*
  - *PCM standard (с короткой или длинной синхронизацией фрейма на 16-битном фрейме канала или 16-битном фрейме данных, расширенном до 32-битного фрейма канала).*
- *Порядок следования бит - старший бит всегда идет первым (MSB first).*
- *Возможность применения DMA для передачи и приема (данных шириной 16 бит).*
- *Может выводиться тактовая частота (master clock) для управления внешним аудио-компонентом. Коэффициент фиксирован на 256 x FS (здесь FS это частота аудиовыборок).*
- *Оба варианта I2S (I2S2 и I2S3) имеют выделенный блок PLL (PLLI2S), чтобы генерировать еще более точную тактовую частоту.*
- *Такты I2S (I2S2 и I2S3) могут быть получены от внешнего тактового сигнала, поступающего на вывод I2S_CKIN.*

[Общее описание SPI]

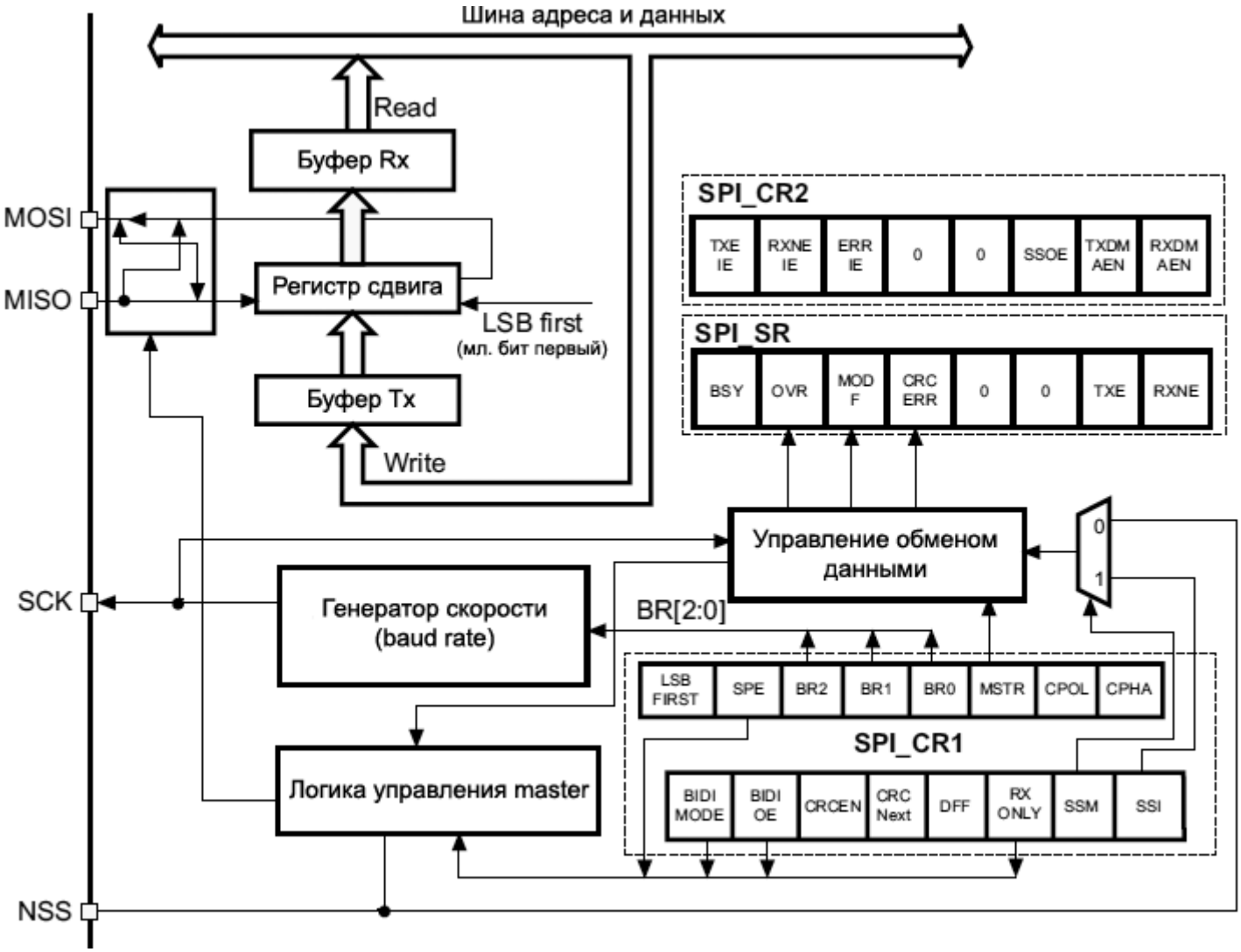

Рис. 246. Блок-схема SPI.

Обычно SPI подключается к внешним устройствам через 4 сигнала (они отображаются альтернативными функциями AF на определенные выводы микроконтроллера [3, 4]):

MISO: данные Master In / Slave Out. Этот сигнал может использоваться для передачи данных данных в режиме slave и приема данных в режиме master.

MOSI: Master Out / Slave In, данные в противоположном направлении. Этот сигнал может использоваться для передачи данных данных в режиме master и приема данных в режиме slave.

SCK: Serial Clock, тактовый выход для SPI master и тактовый вход для SPI slave.

NSS: Node Slave Select. Этот сигнал необязателен для использования, применяется как аппаратная выборка slave-устройства. Сигнал позволяет главному устройству шины SPI выбрать определенное подчиненное устройство, чтобы избежать конфликта на сигналах данных. Сигналы выборки NSS на главном устройстве могут также вырабатываться программно, управлением ножками GPIO. Ножка NSS также может использоваться как выход, если это разрешено (битом SSOE), и переключаться в лог. 0 при обращении к внешним подчиненным устройствам, если SPI работает в конфигурации master. Таким образом, все ножки NSS устройств, подключенных к сигналу Master NSS, видят лог. 0 и становятся подчиненными по низкому уровню, когда они сконфигурированы в аппаратном режиме выборки NSS. Когда конфигурируется режим master, и NSS конфигурируется как вход (MSTR=1 и SSOE=0), и если NSS переводится в лог. 0, то SPI входит в состояние отказа главного устройства (master mode fault state): бит MSTR автоматически очищается, и устройство конфигурируется в режиме подчиненного устройства, slave mode (см. далее раздел "Флаги ошибок").

Базовый пример взаимодействия между одним master и одним slave показан на рис. 247.

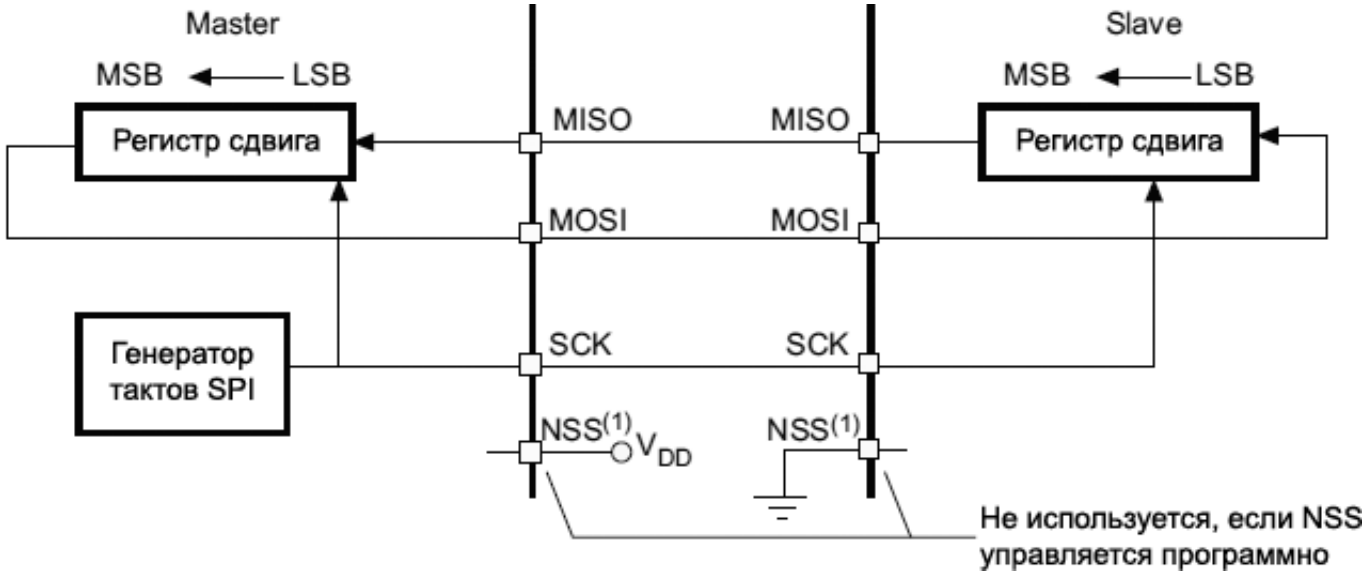

Рис. 247. Приложение, где соединена одиночная пара устройств master и slave.

Примечание (1): здесь вывод NSS сконфигурирован как вход.

Все одноименные выводы соединяются друг с другом. Таким образом, данные последовательно передаются между master и slave, одновременно в двух направлениях (по умолчанию старшим битом MSB вперед).

Обмен всегда инициируется со стороны master-устройства. Когда master передает данные slave-устройству через вывод MOSI, slave-устройство отвечает данными через вывод MISO. Это подразумевает полнодуплексный обмен, засинхронизированный одним тактовым сигналом SCK (этот сигнал генерирует master).

Управление выборкой NSS. Битом SSM может быть выбрано аппаратное или программное управление сигналом NSS.

- Программное управление NSS (SSM = 1). В этом случае информация об управлении подчиненным устройством предоставляется внутренне SSI в регистре SPI_CR1. Внешний вывод NSS остается свободным для других функций приложения.
- Аппаратное управление NSS (SSM = 0). Возможны 2 конфигурации выхода NSS (определяются битом SSOE в регистре SPI_CR2):

  – Разрешен выход NSS (SSM = 0, SSOE = 1). Эта конфигурация используется только когда устройство работает в режиме master. Сигнал NSS переводится в лог. 0, когда master запускает обмен, и NSS удерживается в лог. 0 до тех пор, пока SPI не будет запрещен.

  – Выход NSS запрещен (SSM = 0, SSOE = 0). Эта конфигурация позволяет работать нескольким главным устройствам на шине (multimaster). Для устройств, настроенных как slave, вывод NSS работает как классический вход NSS: подчиненное устройство выбирается, когда на NSS подается лог. 0, и его выборка снимается, когда на NSS подана лог. 1.

Фаза и полярность тактов. Программно можно выбрать 4 варианта взаимодействия по шине SPI, используя биты CPOL и CPHA в регистре SPI_CR1. Бит CPOL (Clock POLarity, полярность тактов) управляет устойчивым (т. е. исходным) состоянием сигнала тактов, когда данные не передаются. Этот бит влияет на оба режима, master и slave. Если CPOL=0, то ножка SCK в режиме ожидания находится в лог. 0. Если CPOL=1, то ножка SCK в режиме ожидания находится в лог. 1.

Если установлен бит CPHA (Clock PHAse, фаза тактов), то второй перепад на выводе SCK (спад в случае CPOL=0, фронт в случае CPOL=1) служит стробом бита данных (по умолчанию MSB). Данные защелкиваются при появлении второго перепада тактов. Если CPHA=0, то стробом служит первый перепад на выводе SCK (спад уровня при CPOL=1, нарастание уровня при CPOL=0). Данные защелкиваются при появлении первого перепада тактов.

Комбинация состояния бит CPOL и CPHA выбирает тактовый перепад для захвата данных.

Рис. 248 показывает передачу данных SPI для всех четырех возможных комбинаций бит CPHA и CPOL. Эта диаграмма может быть интерпретирована как диаграмма сигналов master или slave, где выводы SCK, MISO, MOSI соединены напрямую между устройствами master и slave. Порядок следования бит, когда старший бит (MSB) идет первым, выбран по умолчанию (бит LSBFIRST=0 в регистре SPI_CR1).

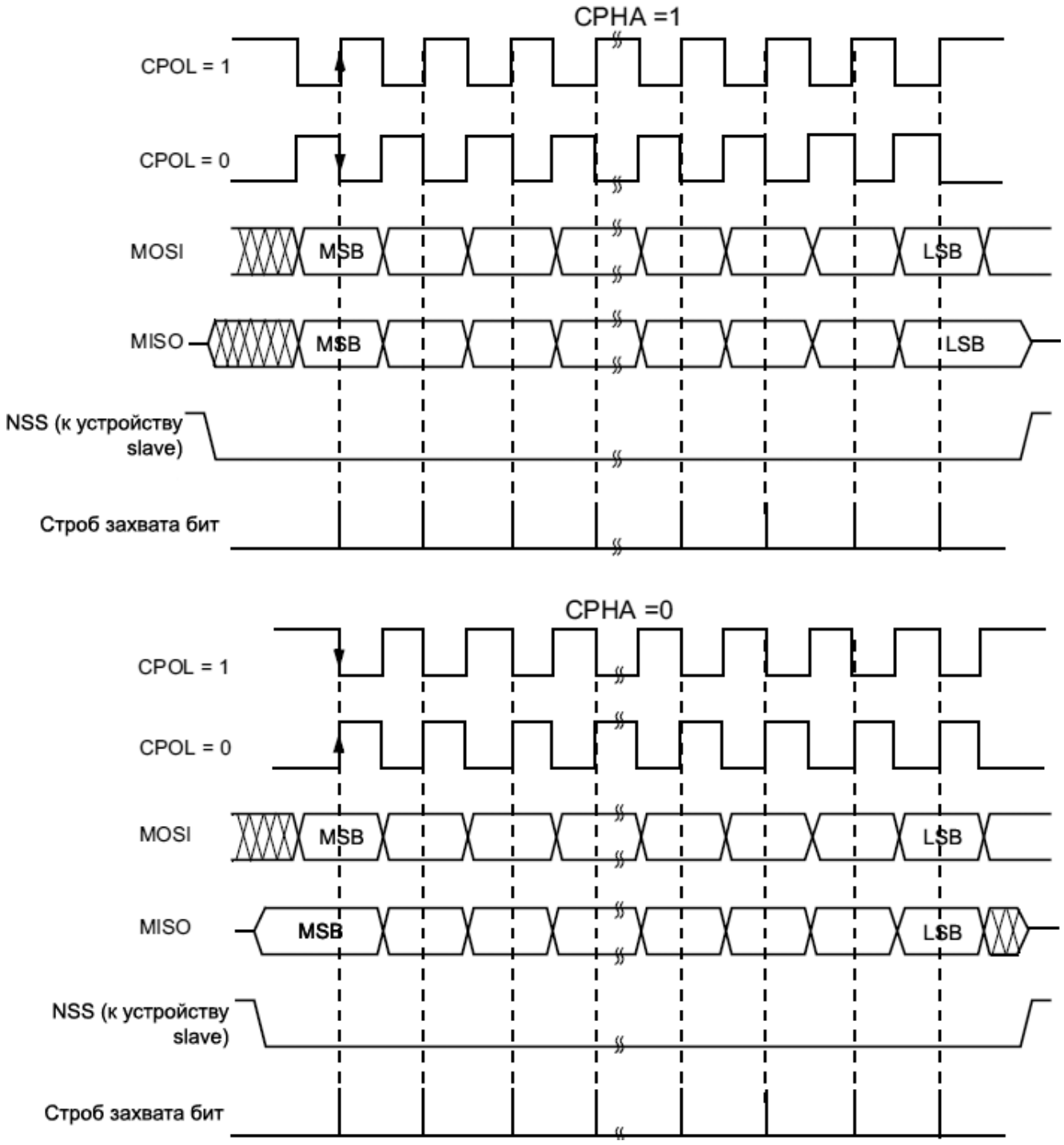

Рис. 248. Диаграммы сигналов для разных вариантов формата фрейма.

Примечание: перед тем, как поменять биты CPOL/CPHA, периферийное устройство SPI должно быть запрещено путем сброса бита SPE. Master и slave должны быть запрограммированы в одинаковом режиме интервалов времени. Состояние ожидания SCK должно соответствовать выбранной полярности в регистре SPI_CR1 (лог. 1 на SCK при CPOL=1, или лог. 0 на SCK при CPOL=0). Формат данных фрейма (8 или 16 бит) выбирается битом DFF в регистре SPI_CR1, и это определяет длину данных во время передачи/приема.

Порядок следования бит. Формат фрейма также зависит от того, какой бит данных передается первым - старший (MSB) или младший (LSB). Это зависит от значения бита LSBFIRST в регистре SPI_CR1.

Длина фрейма. Элементарная порция данных, передаваемая по шине (фрейм, или кадр), может состоять либо из 8, либо из 16 бит, в зависимости от состояния бита DFF в регистре SPI_CR1.

Выбранный формат фрейма (порядок следования бит и длина данных) распространяется на прием и/или передачу.

Конфигурирование режима SPI slave

В конфигурации slave последовательные такты принимаются на выводе SCK от устройства master. Значение, установленное битами BR[2:0] в регистре SPI_CR1, никак не влияет на скорость передачи данных.

Примечание: рекомендуется разрешить SPI до того, как master начнет передавать тактовый сигнал. Если это не сделать, то может произойти нежелательная передача данных. Регистр данных slave-устройства должен быть готов перед поступлением первого перепада тактового сигнала, или перед окончанием осуществляющегося обмена. Важно, чтобы полярность тактов была установлена в состояние ожидания перед тем, как разрешена работа устройств slave и master.

Для конфигурирования SPI в режиме slave выполните следующую последовательность шагов:

1. *Установите значение бита DFF, чтобы выбрать 8- или 16-разрядный формат фрейма данных.*
2. *Значение бит CPOL и CPHA, чтобы определить один из 4 возможных вариантов поведения данных и тактов (см. рис. 248). Чтобы корректно передавать данные, биты CPOL и CPHA должны быть сконфигурированы одинаково у slave и у master. Этот шаг не требуется, когда битом FRF в регистре SPI_CR2 выбирается режим TI.*
3. *Выберите порядок следования бит (MSB-first или LSB-first) путем программирования бита LSBFIRST в регистре SPI_CR1. Установленный порядок бит должен быть такой же, как и у устройства master. Этот шаг не требуется, когда битом FRF в регистре SPI_CR2 выбирается режим TI.*
4. *В аппаратном режиме управления выборкой ножка NSS должна быть соединена с лог. 0 во время полной последовательности передачи фрейма. В программном режиме NSS установите бит SSM и очистите бит SSI в регистре SPI_CR1. Этот шаг не требуется, когда битом FRF в регистре SPI_CR2 выбирается режим TI.*
5. *Установите бит FRF в регистре SPI_CR2, чтобы выбрать режим протокола TI.*
6. *Очистите бит MSTR и установите бит SPE (оба эти бита находятся в регистре SPI_CR1), чтобы назначить выводам SPI альтернативные функции.*

В этой конфигурации ножка MOSI работает как вход данных, а ножка MISO как выход данных.

Передача. Передаваемая порция данных (8 или 16 бит) параллельно загружается в буфер Tx во время цикла записи.

Последовательность передачи начинается, когда slave-устройство получит тактовый сигнал, и тогда оно выведет свой первый бит данных (MSB или LSB, в зависимости от бита LSBFIRST) на ножку MOSI. Остальные биты (7 бит в случае 8-разрядного фрейма данных и 15 в случае 16-разрядного фрейма данных) загружаются в регистр сдвига. Установится флаг TXE в регистре SPI_SR при перемещении данных из буфера Tx в регистр сдвига, и будет сгенерировано прерывание, если установлен бит разрешения прерывания TXEIE в регистре SPI_CR2.

Прием. Для приемника, когда завершится процедура передачи данных:

- *Данные в регистре сдвига переместятся в буфер Rx, и установится флаг RXNE в регистре SPI_SR.*
- *Будет сгенерировано прерывание, если в регистре SPI_CR2 установлен бит разрешения прерывания RXNEIE.*

После последнего тактового перепада установится бит RXNE, копия принятой порции данных переместится из регистра сдвига в буфер Rx. При считывании регистра SPI_DR периферийное устройство SPI вернет это принятое значение.

Очистка бита RXNE осуществляется чтением регистра SPI_DR.

Протокол SPI TI. В режиме slave интерфейс SPI совместим с протоколом TI. Для совместимости с этим протоколом может быть установлен бит FRF в регистре SPI_CR2.

Полярность и фаза тактов принудительно устанавливаются в формат протокола TI, независимо от настроек регистра SPI_CR1. Управление NSS также устанавливается специфичным для протокола TI, что делает конфигурацию управления NSS через регистры SPI_CR1 и SPI_CR2 (состояние бит SSM, SSI, SSOE) прозрачной для пользователя.

В режиме Slave (рис. 249 TI mode - Slave mode, одиночная передача и рис. 250 TI mode - Slave mode, непрерывная передача), для управления моментом перехода состояния вывода MISO в Hi-Z (отключено) используется прескалер скорости (SPI baud rate prescaler). Может использоваться любая скорость, что позволяет добиться максимальной гибкости при определении этого момента времени. Однако скорость обычно устанавливается в соответствии с тактовой частотой SCK внешнего master. Время для перехода сигнала MISO в состояние HI-Z (освобождение шины SPI) зависит от внутренних ресинхронизаций и от значения скорости, установленного битами BR[2:0] в регистре SPI_CR1. Это время определяется по формуле:

$$\frac{tbaud\_rate}{2} + 4 \times tpclk < trelease < \frac{tbaud\_rate}{2} + 6 \times tpclk$$

Примечание: эта возможность недоступна для обмена Motorola SPI (FRF=0).

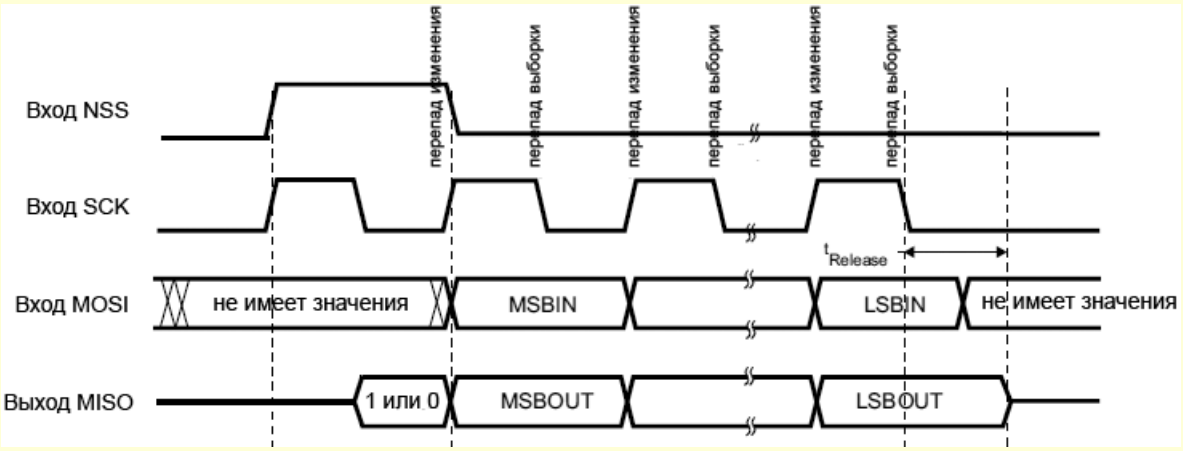

Рис. 249. TI mode - slave mode, одиночная передача.

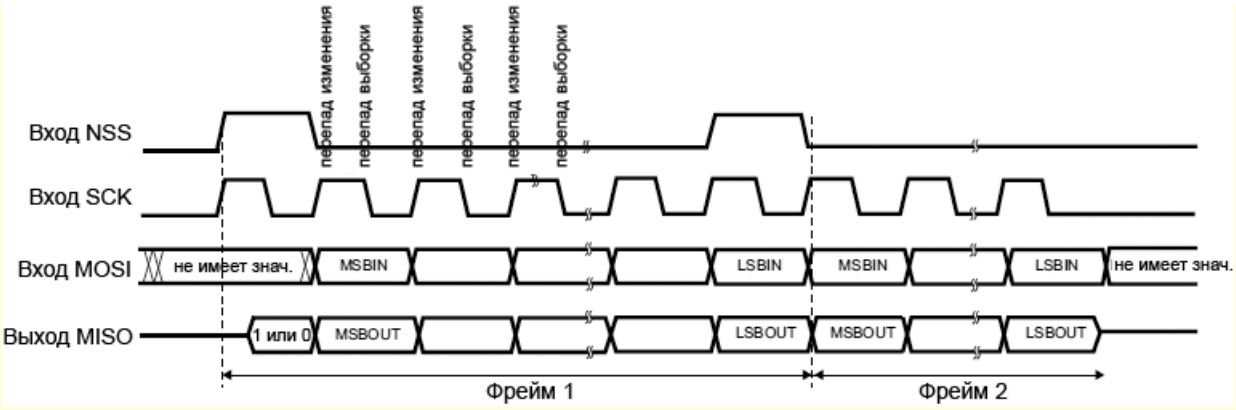

Рис. 250. TI mode - slave mode, непрерывная передача.

Чтобы детектировать ошибки фрейма TI в режиме "только передатчик" (Slave transmitter only mode) с использованием прерывания ошибки (Error interrupt, ERRIE = 1), SPI должен быть сконфигурирован 2-проводном однонаправленном режиме путем установки в регистре SPI_CR1 бита BIDIMODE и бита BIDIOE в лог. 1. Когда BIDIMODE=0, бит OVR установится в 1, потому что регистр данных никогда на считывается и всегда генерируется прерывание ошибки, в то время когда BIDIMODE=1, данные не принимаются и OVR никогда не установится.

Конфигурирование режима SPI master

В режиме главного устройства (master) SPI формирует тактовый сигнал. Для конфигурирования SPI в режиме master выполните следующую последовательность шагов:

1. *Определите битами BR[2:0] частоту тактов SCK (см. описание регистра SPI_CR1).*
2. *Значениями бит CPOL и CPHA выберите один из 4 вариантов взаимодействия сигналов тактов и данных (см. рис. 248). Этот шаг не требуется, когда выбран режим TI.*
3. *Сконфигурируйте бит DFF в регистре SPI_CR1 для выбора 8- или 16-разрядного формата данных фрейма.*
4. *Сконфигурируйте бит LSBFIRST в регистре SPI_CR1 для выбора формата фрейма. Этот шаг не требуется, когда выбран режим TI.*
5. *Если ножка NSS требуется в аппаратном режиме входа, то соедините NSS с сигналом высокого уровня во время выполнения последовательности передачи байта. В программном режиме NSS установите биты SSM и SSI в регистре SPI_CR1. Если ножка NSS нужна как выход, то нужно установить только бит SSOE. Этот шаг не требуется, когда выбран режим TI.*
6. *Установите бит FRF в регистре SPI_CR2, чтобы выбрать режим протокола TI.*
7. *Должны быть установлены биты MSTR и SPE (они останутся установленными только если вход NSS соединен с сигналом лог. 1).*

В этой конфигурации ножка MOSI работает как выход данных, и ножка MISO как вход данных.

Передача. Последовательность передачи начинается, когда в буфер Tx записан байт. Этот байт данных параллельно загружается в регистр сдвига (через внутреннюю шину) во время передачи первого бита, и затем биты выдвигаются последовательно через MOSI старшим битом (MSB) или младшими битом (LSB) вперед, в зависимости от состояния бита LSBFIRST в регистре SPI_CR1. Установится флаг TXE в момент передачи данных из буфера Tx в регистр сдвига, и будет сгенерировано прерывание, если установлен бит TXEIE в регистре SPI_CR2.

Прием. Для приемника, когда передача данных завершена:

- *Данные из регистра сдвига переносятся в буфер RX, и установится флаг RXNE.*
- *Будет сгенерировано прерывание, если установлен бит RXNEIE в регистре SPI_CR2.*

На последнем тактовом перепаде захвата бита установится бит RXNE, и копия принятого байта переместится в буфер Rx. Когда происходит считывание регистра SPI_DR, возвращается значение принятого байта из буфера.

Очистка бита RXNE происходит путем чтения регистра SPI_DR.

Непрерывный поток передачи может быть достигнут, если следующая порция данных для передачи была помещена в буфер Tx в момент, когда началась передача. Имейте в виду, что флаг TXE должен быть в состоянии 1 перед любой записью в буфер Tx.

Примечание: если master обменивается со slave-устройствами SPI, у которых должна быть отменена выборка между передачами, ножка NSS должна быть сконфигурирована как GPIO, или другой вывод GPIO должен использоваться для программного формирования сигнала выборки.

Протокол SPI TI. В режиме master интерфейс SPI совместим с протоколом TI. Для совместимости с этим протоколом может быть установлен бит FRF в регистре SPI_CR2.

Полярность и фаза тактов принудительно устанавливаются в формат протокола TI, независимо от настроек регистра SPI_CR1. Управление NSS также устанавливается специфичным для протокола TI, что делает конфигурацию управления NSS через регистры SPI_CR1 и SPI_CR2 (состояние бит SSM, SSI, SSOE) прозрачной для пользователя.

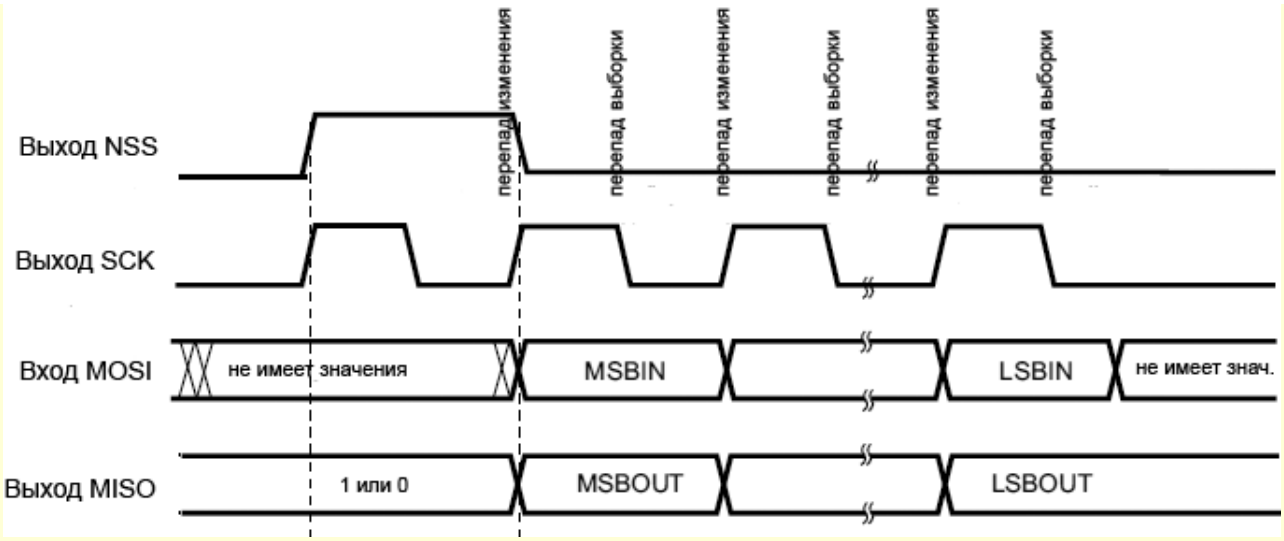

Рис. 251. TI mode - master mode, одиночная передача.

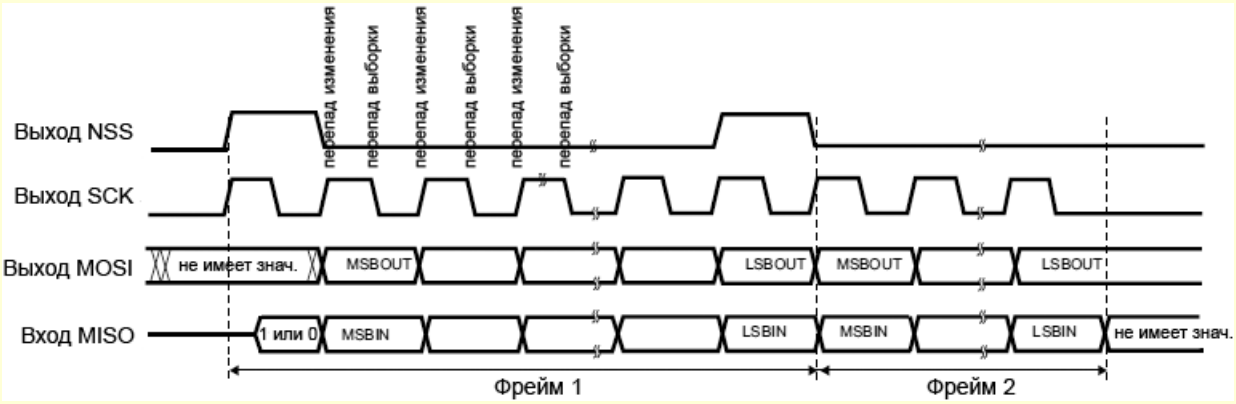

Рис. 252. TI mode - master mode, непрерывная передача.

Рисунки 251 и 252 показывают формы сигналов SPI master, когда выбран режим TI.

Конфигурирование SPI для полудуплекса

SPI может работать полудуплексом в двух конфигурация.

- *1 тактовый сигнал и 1 двунаправленный сигнал данных (BIDIMODE = 1).*
- *1 тактовый сигнал и 1 однонаправленный сигнал данных (BIDIMODE = 0, только прием или только передача).*

Двунаправленная линия данных. Этот режим выбирается установкой бита BIDIMODE в регистре SPI_CR1. Сигнал SCK используется как такты, и сигнал ножка MOSI используется для данных в режиме master, или для данных используется ножка MISO в режиме slave. Направление передачи (вход или выход) выбирается битом BIDIOE в регистре SPI_CR1. Когда этот бит установлен в лог. 1, линия данных работает как выход, иначе как вход.

Однонаправленная линия данных. В этом режиме приложение может использовать SPI либо только на передачу, либо только на прием.

- *Режим "только передача" (transmit-only) подобен режиму полного дуплекса (BIDIMODE=0, RXONLY=0): данные передаются через MOSI в режиме master или через MISO в режиме slave). Ножка приема (MISO в режиме master или MOSI в режиме slave) может использоваться как обычный порт ввода/вывода (GPIO). В этом случае приложению просто нужно игнорировать буфер Rx (если прочитать регистр данных SPI_DR, то в нем не будет содержаться принятое значение).*
- *В режиме "только прием" (receive-only) приложение может запретить функцию выхода SPI путем установки бита RXONLY в регистре SPI_CR1. В этом случае освобождается ножка передачи (MOSI в режиме master или MISO в режиме slave), и её можно использовать для других целей.*

Для старта обмена в режиме receive-only сконфигурируйте и разрешите SPI следующим образом:

- *В режиме master прием начнется немедленно и остановится, когда будет очищен бит SPE. Сброс бита SPE немедленно остановит текущий прием. В этом режиме не требуется считывать флаг занятости BSY. Он всегда установлен, когда происходит обмен SPI.*
- *В режиме slave блок SPI продолжит принимать, пока ножка NSS притянута к лог. 0 (или пока очищен бит SSI в режиме программной обработки NSS), и пока поступают такты SCK.*

[Процедуры приема и передачи]

Буферы Rx и Tx. Принятые данные сохраняются во внутреннем буфере Rx, в то время как при передаче данные предварительно сохраняются во внутренний буфер Tx, после чего они передаются.

Доступ на чтение к регистру SPI_DR возвратит буферизированное значение Rx, в то время как доступ на запись к SPI_DR сохранит записанные данные в буфер Tx.

Master в полном дуплексе (BIDIMODE=0, RXONLY=0). Процесс передачи данных начинается, когда данные записаны в регистр SPI_DR (буфер Tx). Затем данные параллельно загружается из буфера Tx в регистр сдвига, и начинается их последовательный вывод через ножку MOSI. Одновременно принимаемые через ножку MISO данные последовательно вдвигаются в регистр сдвига, и по окончании приема параллельно загружается в регистр SPI_DR (буфер Rx).

Однонаправленный master, только прием (BIDIMODE=0, RXONLY=1). Процесс начнется, как только SPE=1, активированный приемник начнет принимать данные через ножку MISO и последовательно вдвигать из в регистр сдвига. После окончания приема данные параллельно загружаются в регистр SPI_DR (буфер Rx).

Двунаправленный master в режиме передачи (BIDIMODE=1, BIDIOE=1). Процесс передачи начинается, когда данные записаны в регистр SPI_DR (буфер Tx). После этого данные параллельно загружаются из буфера Tx в регистр сдвига, одновременно с передачей первого бита. Все биты данных последовательно передаются через ножку MOSI. При этом никакие данные не принимаются.

Двунаправленный master в режиме приема (BIDIMODE=1, BIDIOE=0). Пороцесс начинается, как только SPE=1 и BIDIOE=0. Принятые через ножку MOSI данные последовательно вдвигаются в регистр сдвига, и затем параллельно загружаются в регистр SPI_DR (буфер Rx). Передатчик при этом не активируется, и через ножку MOSI данные не передаются.

Slave в полном дуплексе (BIDIMODE=0, RXONLY=0). Процесс начинается по мере поступления тактовых импульсов на ножку SCK, данные вдвигаются в ножку MOSI. Одновременно загруженные параллельно данные в буфер Tx будут выдвигаться через регистр сдвига наружу, появляясь на ножке выхода MISO. Программное обеспечение должно предварительно записать передаваемые данные в SPI_DR (буфер Tx) до того, как SPI master начнет процесс передачи.

Однонаправленный slave, только прием (BIDIMODE=0, RXONLY=1). Процесс начинается, когда slave получает тактовый сигнал, и первый бит поступит на вход MOSI. Принимаемые биты вдвигаются в регистр сдвига через ножку MOSI, пока не будут приняты все остальные биты. Передатчик при этом не активируется, и данные через ножку MISO не выдвигаются.

Двунаправленный slave при передаче (BIDIMODE=1, BIDIOE=1). Процесс начинается, когда slave получит сигнал тактов, первый бит в буфере Tx появляется на выходе MISO. Затем данные через регистр сдвига последовательно, по тактовым импульсам SCK выдвигаются через MISO, пока не будет выдвинут последний бит. Программное обеспечение должно предварительно записать передаваемые данные в SPI_DR (буфер Tx) до того, как SPI master начнет процесс передачи. Никакие данные не принимаются.

Двунаправленный slave на приеме (BIDIMODE=1, BIDIOE=0). Процесс начинается, когда slave получит сигнал тактов, и первый принимаемый бит присутствует на входе MOSI. Все поступающие по тактам SCK данные вдвигаются в регистр сдвига, вплоть до последнего бита, после чего данные параллельно переносятся из регистра сдвига в регистр SPI_DR (буфер Rx). Передатчик не активируется, и никакие данные на выходе MISO не появляются.

Обработка передачи и приема данных. Флаг TXE (Tx buffer empty, буфер передачи пуст) установится, когда записанные в буфер Tx данные были перенесены в регистр сдвига. Факт установки этого показывает, что внутренний буфер Tx готов к загрузке следующей порции данных. В момент установки TXE может генерироваться прерывание, если установлен бит TXEIE в регистре SPI_CR2. Очистка бита TXE выполняется путем записи в регистр SPI_DR.

Примечание: программа должна обеспечить состояние лог. 1 флага TXE перед любой попыткой записи в буфер Tx (регистр SPI_DR). Иначе операция записи перезапишет данные буфера передачи до того, как они были реально отправлены.

Флаг RXNE (Rx buffer not empty, буфер приема не пуст) установится на последнем тактовом перепаде приема бит, когда данные переносятся из регистра сдвига в буфер Rx (регистр SPI_DR). Факт установки этого флага показывает, что принятые данные готовы к чтению через регистр SPI_DR. В момент установки флага RXNE может быть сгенерировано прерывание, если установлен бит RXNEIE в регистре SPI_CR2. Очистка бита RXNE выполняется путем чтения регистра SPI_DR.

Для некоторых конфигураций может использоваться флаг BSY во время последней транзакции данных, чтобы ждать завершения транзакции.

Дуплекс. Для полнодуплексной процедуры передачи и приема в режиме master или slave (BIDIMODE=0, RXONLY=0) программа должна следовать следующей процедуре для передачи и приема данных (см. рис. 253 и рис. 254):

1. Разрешение SPI установкой бита SPE=1.
2. Запись первого передаваемого элемента данных в регистр SPI_DR (это очистит флаг TXE).
3. Ожидание, когда установится флаг TXE=1, и затем запись второго элемента даных для передачи. После этого ожидание RXNE=1 и чтение SPI_DR, чтобы получить первый принятый элемент данных (чтение SPI_DR очистит бит RXNE). Шаг 3 передачи/приема повторяется до тех пор, пока не будет принято n–1 элементов данных.
4. Ожидание RXNE=1, и затем чтение последнего принятого элемента данных.
5. Ожидание TXE=1, после чего ожидание BSY=0 перед запретом SPI.

Эта процедура может быть также реализована с помощью выделенных подпрограмм обработки прерываний (ISR), которые запускаются в момент установки флага RXNE или TXE.

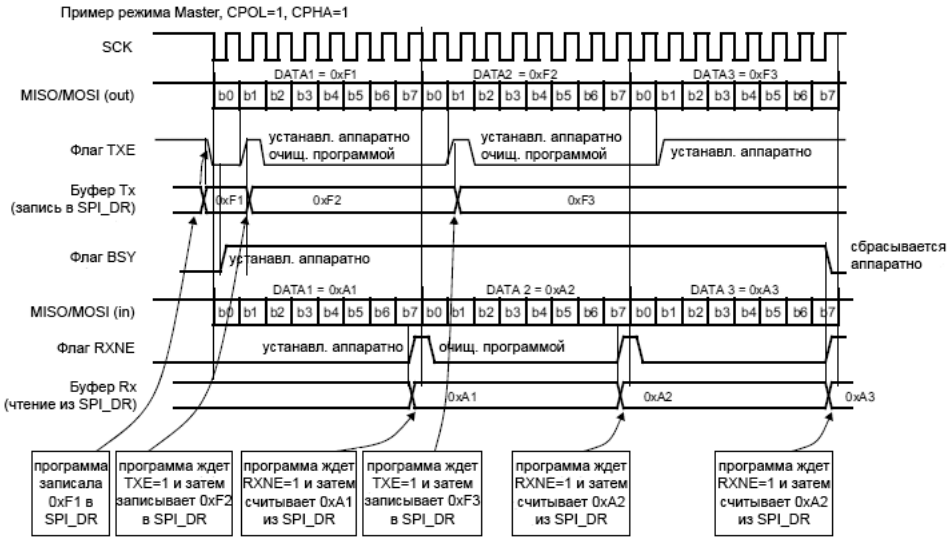

Рис. 253. Поведение TXE/RXNE/BSY в режиме Master / full-duplex (BIDIMODE=0, RXONLY=0), в случае непрерывной передачи.

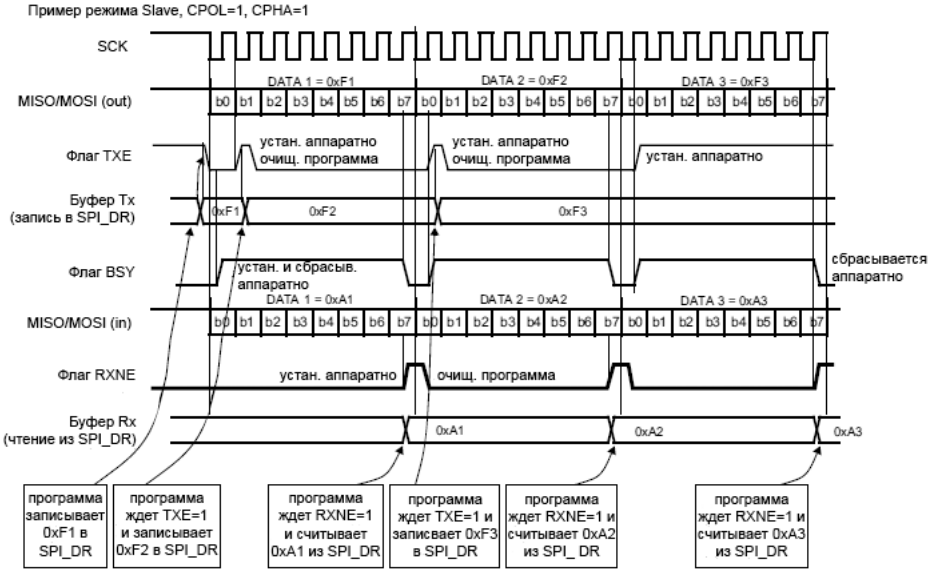

Рис. 254. Поведение TXE/RXNE/BSY в режиме Slave / full-duplex (BIDIMODE=0, RXONLY=0), в случае непрерывной передачи.

Обработка только передачи (BIDIMODE=0, RXONLY=0). В этом режиме процедура может быть сокращена, и бит BSY может использоваться для ожидания завершения передачи (см. рис. 255 и 256).

1. Разрешение SPI установкой SPE=1.
2. Запись первого элемента данных для для отправки в регистр SPI_DR (это очистит TXE).
3. Ожидание TXE=1 и запись следующего передаваемого элемента данных. Шаг 3 повторяется до тех пор, пока не будет передан каждый элемент данных.
4. После записи последнего элемента данных в SPI_DR осуществляется ожидание TXE=1, затем ожидание BSY=0, что покажет завершение передачи последнего элемента данных.

Эта процедура может быть также реализована с помощью выделенных подпрограмм обработки прерываний (ISR), которые запускаются в момент установки флага TXE.

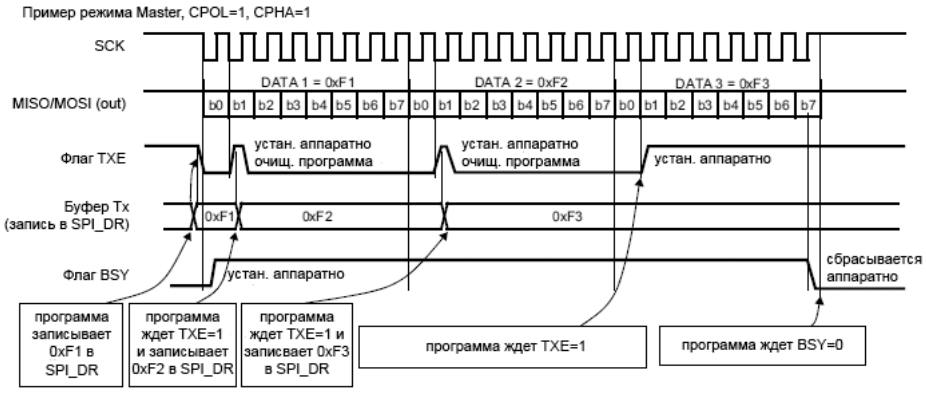

Рис. 255. Поведение TXE/BSY в режиме Master / transmit-only (BIDIMODE=0, RXONLY=0), в случае непрерывной передачи.

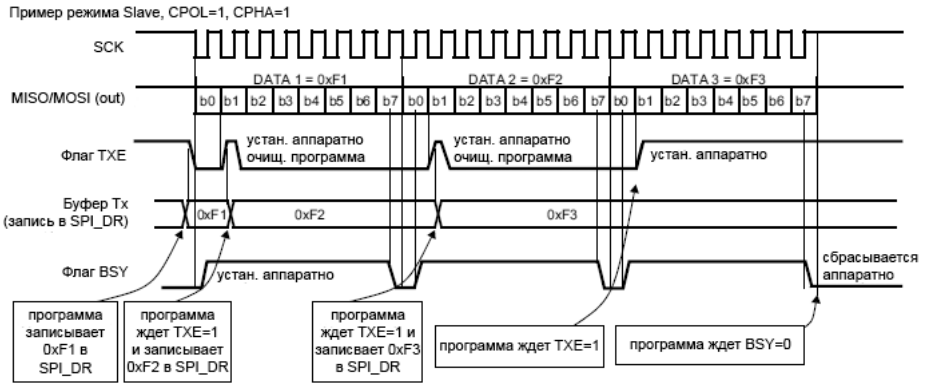

Рис. 256. Поведение TXE/BSY в режиме Slave / transmit-only (BIDIMODE=0, RXONLY=0), в случае непрерывной передачи.

Примечание: во время прерывистых коммуникаций существует задержка 2 периода тактов APB между операцией записи в SPI_DR и установкой бита BSY. Как следствие в режиме "только передача", необходимо сначала дождаться установки TXE, и затем ждать очистки BSY после записи последней порции данных. После передачи двух элементов данных в режиме "только передача", установится флаг OVR в регистре SPI_SR, когда принятые данные никогда не считываются.

Обработка двунаправленной передачи по одному сигналу данных (BIDIMODE=1, BIDIOE=1). В этом режиме процедура подобна процедуре "только передача" с тем исключением, что оба бита BIDIMODE и BIDIOE в регистре SPI_CR2 должны быть установлены перед разрешением SPI.

Однонаправленный прием (BIDIMODE=0, RXONLY=1). В этом случае может использоваться следующая упрощенная процедура (см. рис. 257):

1. Установка бит RXONLY в регистре SPI_CR1.
2. Разрешение SPI установкой в 1 бита SPE:
   a) В режиме master это немедленно активизирует генерацию тактов SCK, и данные будут последовательно приниматься до тех пор, пока SPI не будет запрещен (SPE=0).
   b) В режиме slave данные принимаются, когда устройство SPI master переведет сигнал NSS в лог. 0 и начнет генерировать такты SCK.
3. Ожидание RXNE=1, и затем чтение SPI_DR для получения принятых данных (это очистит бит RXNE). Шаг 3 повторяется до тех пор, пока не будет принят каждый элемент данных.

Описанная процедура может быть реализована с помощью выделенных процедур обработки прерывания (ISR) на каждой установке флага RXNE.

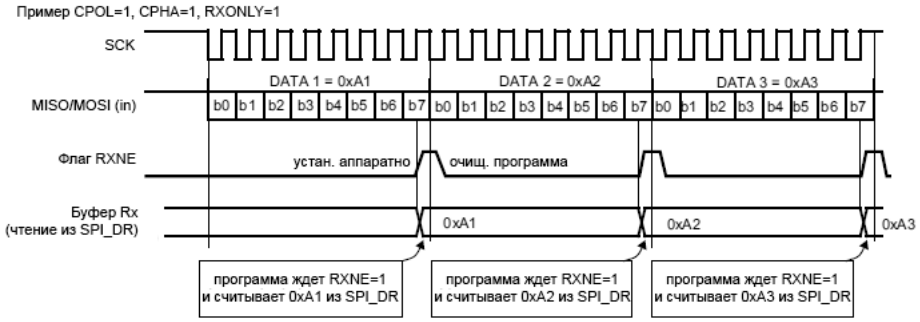

Рис. 257. Поведение RXNE в режиме receive-only (BIDIRMODE=0, RXONLY=1) в случае непрерывной передачи.

Примечание: если требуется запрет SPI после последней передачи, следуйте процедуре, описанной далее в секции "Запрет SPI".

Прием в двунаправленном режиме (BIDIMODE=1, BIDIOE=0). В этом режиме процедура подобна режиму "только прием" с тем исключением, что бит BIDIMODE должен быть установлен, и бит BIDIOE должен быть сброшен в регистре SPI_CR2 перед разрешением SPI.

Непрерывные и прерывистые передачи. Когда передаются данные в режиме master, если программа работает достаточно быстро, чтобы детектировать каждый момент установки флага TXE (или своевременно обрабатывать прерывание TXE) и немедленно записывать регистр SPI_DR до того, как текущая передача данных завершилась, то обмен данными считается непрерывным. В этом случае нет пауз в генерации тактов SPI, и бит BSY никогда не очищается передачами каждой порции данных.

Напротив, если программа работает недостаточно быстро, то могут быть перерывы в обмене данными. Тогда бит BSY очищается между передачами данных (см. рис. 258).

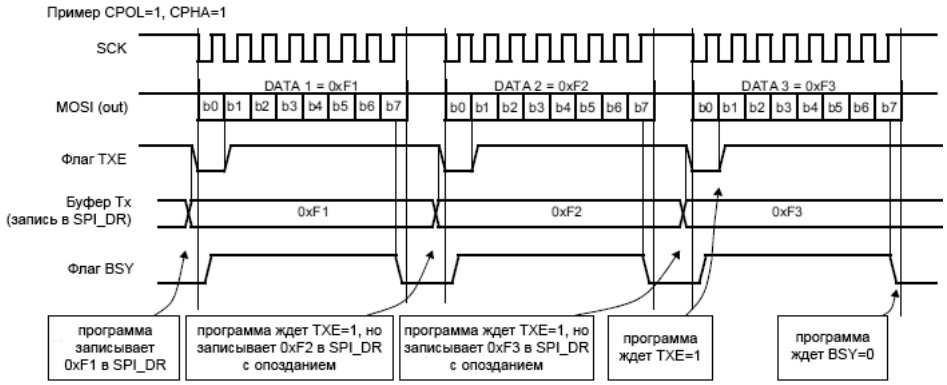

Рис. 258. Поведение TXE/BSY при передаче (BIDIRMODE=0, RXONLY=0) в случае прерывистой передачи.

В режиме master receive-only (RXONLY=1), обмен всегда непрерывный, и флаг BSY всегда считывается как 1.

В режиме slave непрерывность обмена определяется целиком устройством master шины SPI. В любом случае, даже если обмен непрерывный, флаг BSY переходит в лог. 0 между каждой передачей на минимальный интервал времени в 1 такт периода SPI (см. рис. 256).

Вычисление CRC

Для надежности коммуникаций реализованы калькуляторы CRC, отдельный для передачи данных и отдельный для приема. CRC вычисляется по программируемому полиному, последовательно на каждом бите. Вычисление происходит по перепаду выборки бита данных, который определяется битами CPHA и CPOL в регистре SPI_CR1.

Примечание: блок SPI предоставляет реализацию 2 видов стандартов вычисления CRC, что напрямую зависит от выбранного формата данных фрейма для передачи и/или приема: 8 бит данных (CRC8) и 16 бит данных (CRC16).

Вычисление CRC разрешается установкой бита CRCEN в регистре SPI_CR1. Это действие сбросит регистры CRC (SPI_RXCRCR и SPI_TXCRCR). В режимах full-duplex или transmit-only, когда передача управляется программно (режим CPU), необходимо записать бит CRCNEXT сразу после того, как последний передаваемый элемент данных был записан в SPI_DR. По окончанию этой последней передачи будет передано содержимое SPI_TXCRCR.

В режиме receive-only, и когда передача управляется программно (режим CPU), необходимо записать CRCNEXT после предпоследнего принятого элемента данных. CRC принимается сразу после приема последнего элемента данных, и выполняется проверка CRC. По окончании передачи данных и CRC установится флаг CRCERR в регистре SPI_SR, если была обнаружена ошибка во время передачи данных.

Если в буфере TX присутствуют данные, то значение CRC передается только после передачи последнего байта данных. Во время передачи CRC калькулятор CRC выключается, и значение регистра контрольной суммы остается неизменным.

Коммуникация SPI с применением CRC возможна при соблюдении следующей процедуры:

1. *Программирование значений CPOL, CPHA, LSBFirst, BR, SSM, SSI и MSTR.*
2. *Программирование полинома в регистр SPI_CRCPR.*
3. *Разрешение вычисления CRC установкой бита CRCEN в регистре SPI_CR1. Это также очистит регистры SPI_RXCRCR и SPI_TXCRCR.*
4. *Разрешение SPI установкой бита SPE в регистре SPI_CR1.*
5. *Запус обмена, и его продолжение до того момента пока все кроме последнего элемента данных не будут переданы или приняты.*
   - *– В режиме full-duplex или transmit-only, когда передачей управляет программа, при записи последнего элемента данных в буфер Tx устанавливается бит CRCNEXT в регистре SPI_CR1, показывая тем самым, что будет передана CRC после передачи последней порции данных.*
   - *– В режиме receive-only устанавливается бит CRCNEXT после приема предпоследнего элемента данных, чтобы подготовить SPI ко входу в фазу CRC по окончании приема последнего элемента данных. Во время передачи CRC вычисление CRC останавливается.*
6. *После передачи последнего элемента данных SPI входит в фазу передачи и проверки CRC. В режиме full-duplex или receive-only принятое значение CRC сравнивается со значением SPI_RXCRCR. Если значения не совпали, то установится флаг CRCERR в регистре SPI_SR, и может быть сгенерировано прерывание, если установлен бит ERRIE в регистре SPI_CR2.*

Когда SPI работает в режиме slave, будьте осторожны, чтобы разрешить вычисление CRC только когда такты стабилизировались, т. е. тактирование находится в устойчивом состоянии. Если этого не сделать, то может произойти неправильное вычисление CRC. Фактически CRC чувствительно ко входу тактов SCK slave, как только был установлен бит CRCEN, и при этом независимо от значения бита SPE.

На высоких скоростях будьте осторожны с передачей CRC. Поскольку количество используемых тактов CPU на фазе передачи CRC должно быть минимальным, насколько это возможно, запрещается вызывать программные функции последовательности передачи CRC, чтобы избежать ошибок в приеме последних данных и CRC. Фактически бит CRCNEXT должен быть записан перед окончанием передачи/приема последнего элемента данных.

Для высоких скоростей рекомендуется использовать режим DMA, чтобы избежать деградации производительности канала SPI из-за того, что доступ CPU снижает полосу пропускания SPI.

Когда устройства сконфигурированы как slave, и используется аппаратное управление NSS, то NSS нужно удерживать в лог. 0 между фазой данных и фазой CRC.

CRC и NSS. Когда SPI сконфигурирован в режиме slave с разрешенной функцией CRC, вычисление CRC осуществляется, даже если на вывод NSS подана лог. 1. Это может произойти в случае условий работы в окружении нескольких главных устройств SPI (multislave), когда master обмена обращается к slave-устройствам поочередно.

Между снятием выборки со slave-устройства (лог. 1 на NSS) и новой выборкой slave-устройства (лог. 0 на NSS), значение CRC должно быть сброшено на обоих сторонах - master и slave, чтобы заново синхронизировать процесс вычисления CRC для передачи данных.

Чтобы сбросить CRC, выполните следующую процедуру:

1. *Запретите SPI (SPE=0).*
2. *Очистите бит CRCEN.*
3. *Установите бит CRCEN.*
4. *Разрешите SPI (SPE=1).*

[Флаги статуса]

Существует 4 флага статуса, опрашивая которые приложение может полностью отслеживать состояние шины SPI.

Буфер передачи пуст (TXE). Когда этот флаг установлен, это показывает, что буфер Tx пуст, и готов к записи следующего передаваемого элемента данных. Флаг TXE очищается, когда произошла запись в регистр SPI_DR.

Буфер приема не пуст (RXNE). Когда этот флаг установлен, это показывает, что в буфере Rx присутствуют принятые, но еще не прочитанные данные. Флаг RXNE очищается чтением регистра SPI_DR.

Занятость (BSY). Этот флаг устанавливается и очищается аппаратно (запись в этот бит не дает никакого эффекта). Флаг BSY показывает состояние слоя коммуникации SPI.

Когда BSY=1, это показывает, что SPI занят обменом. Существует исключение только для режима master / двунаправленный прием (MSTR=1, BDM=1 и BDOE=0), когда BSY остается в лог. 0 во время приема.

Флаг BSY полезен для детектирования окончания транзакции, если программа хочет запретить SPI и войти в режим остановки (Halt mode), или запретить тактирование периферийного устройства (для снижения энергопотребления). Определение состояния занятости позволяет избежать повреждения передачи последней порции данных. Для этой цели нужно строго соблюдать описанную ниже процедуру.

Флаг BSY также полезен, чтобы избежать коллизий записи в системах multimaster.

Флаг BSY установится, когда запускается передача, кроме режима master/ двунаправленный прием (MSTR=1, BDM=1 и BDOE=0).

Флаг BSY очистится:

- Когда передача завершится (кроме режима master, если обмен непрерывный).
- Когда SPI запрещен.
- Когда произойдет ошибка режима master (master mode fault, MODF=1).

Когда обмен прерывистый, флаг BSY находится в лог. 0 между каждым обменом.

Когда обмен непрерывный:

- В режиме master флаг BSY должен удерживаться в лог. 1 между всеми передачами.
- В режиме slave флаг BSY переходит в лог. на минимальный интервал в 1 период тактов SPI между каждой передачей.

Примечание: не используйте флаг BSY, чтобы обрабатывать каждую передачу или прием. Вместо этого лучше использовать флаги TXE и RXNE.

[Запрет SPI]

Когда передача прерывается, приложение может остановить обмен запретом периферийного устройства SPI. Это делается очисткой бита SPE.

В некоторых конфигурациях запрет SPI и вход в режим остановки (Halt mode) во время текущей передачи может вызвать повреждение данных, и/или флаг BSY может стать ненадежным. Чтобы избежать подобных эффектов, рекомендуется соблюдать следующую процедуру запрета SPI.

Полный дуплекс master или slave (BIDIMODE=0, RXONLY=0):

1. Ожидание RXNE=1 для приема последних данных.
2. Ожидание TXE=1.
3. Ожидание BSY=0.4.

Запрет SPI (SPE=0) и, если требуется, вход в Halt mode (или запрет тактов периферийного устройства).

Однонаправленный режим (transmit-only), master или slave (BIDIMODE=0, RXONLY=0), либо двунаправленный режим передачи (BIDIMODE=1, BIDIOE=1). После записи последних данных в регистр SPI_DR:

1. Ожидание TXE=1.
2. Ожидание BSY=0.
3. Запрет SPI (SPE=0) и, если требуется, вход в Halt mode (или запрет тактов периферийного устройства).

Однонаправленный режим master receive-only (MSTR=1, BIDIMODE=0, RXONLY=1), или двунаправленный режим приема (MSTR=1, BIDIMODE=1, BIDIOE=0). Этом случай должен обрабатываться специальным образом, чтобы гарантировать, что SPI не инициировал новую передачу. Последовательность действий ниже допустима только для конфигурации SPI Motorola (FRF=0):

1. Ожидание предпоследнего RXNE=1 (принят n–1 элемент данных).
2. Ожидание в течение 1 такта SPI (с помощью пустого программного цикла задержки) перед запретом SPI (SPE=0).
3. Затем ожидание последнего RXNE=1 перед входом в Halt mode (или запретом тактов периферийного устройства).

Когда SPI сконфигурирован в режиме TI (FRF=1), необходимо соблюдать следующую процедуру, чтобы избежать генерации нежелательного импульса на NSS, когда SPI запрещен:

1. Ожидание предпоследнего RXNE=1 (принят n–1 элемент данных).
2. Запрет SPI (SPE=0) в следующем окне фрейма, с помощью программного цикла:

– после как минимум 1 импульса тактов SPI,

– перед началом передачи последнего бита данных.

Примечание: на приеме в режиме двунаправленного master (MSTR=1, BDM=1, BDOE=0), флаг BSY удерживается в лог. 0 во время передач.

В режиме slave receive-only (MSTR=0, BIDIMODE=0, RXONLY=1) или на приеме в двунаправленном режиме (MSTR=0, BIDIMODE=1, BIDOE=0).

1. Вы можете запретить SPI (записью SPE=0) в любой момент времени: текущая передача будет завершена перед тем, как SPI станет запрещенным.
2. Затем, если Вы хотите войти в Halt mode, то должны сначала подождать BSY=0 перед входом в Halt mode (или перед запретом тактирования периферийного устройства).

[Обмен SPI с помощью DMA]

Чтобы реализовать работу на максимальной скорости, в SPI должны поступать данные для передачи с минимальной задержкой. Точно так же требуется своевременно считывать данные из буфера Rx, чтобы избежать переполнения (overrun). Чтобы реализовать эту задачу и разгрузить процессор от действий по загрузке/считыванию данных, блок SPI поддерживает функцию DMA, реализующую простой протокол запроса/подтверждения.

Доступ DMA запрашивается, когда установлен бит разрешения запроса в регистре SPI_CR2. Должны выдаваться отдельные запросы для буферов Tx и Rx (см. рис. 259 и 260):

- При передаче запрос DMA выдается всякий раз, когда бит TXE установлен в 1. Затем DMA записывает данные в регистр SPI_DR (это очистит флаг TXE).
- На приеме запрос DMA выдается всякий раз, когда бит RXNE установлен в 1. Затем DMA считывает регистр SPI_DR (это очистит флаг RXNE).

Когда SPI используется только для передачи данных, можно разрешить только канал SPI Tx DMA. в этом случае установится флаг OVR, потому что принятые данные никогда не будут считываться.

Когда SPI используется только для приема данных, можно разрешить только канал SPI Rx DMA.

В режиме передачи, когда DMA записал все передаваемые данные (установился флаг TCIF в регистре DMA_ISR), может быть отслежено состояние флага BSY, чтобы гарантировать полное завершение обмена SPI. Это нужно для того, чтобы избежать повреждения передаваемых данных последней передачи перед запретом SPI или входом в режим остановки (Stop mode). Программа должна ждать TXE=1, и затем ждать BSY=0.

Примечание: во время прерывистых обменов существует задержка длительностью 2 такта APB между операцией записи в SPI_DR и установкой бита BSY. Как следствие, сначала необходимо дождаться TXE=1, и затем дождаться BSY=0 после записи последних данных.

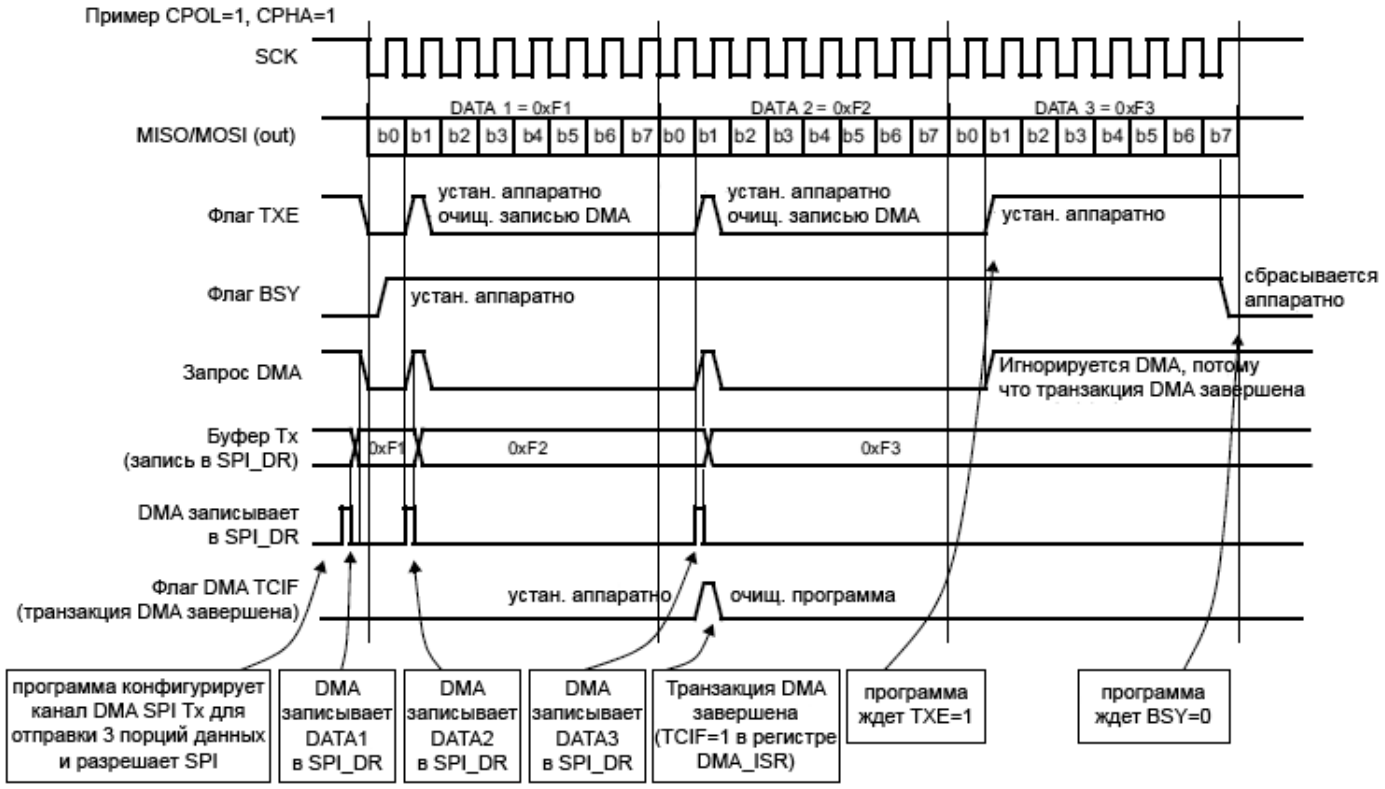

Рис. 259. Передача с помощью DMA.

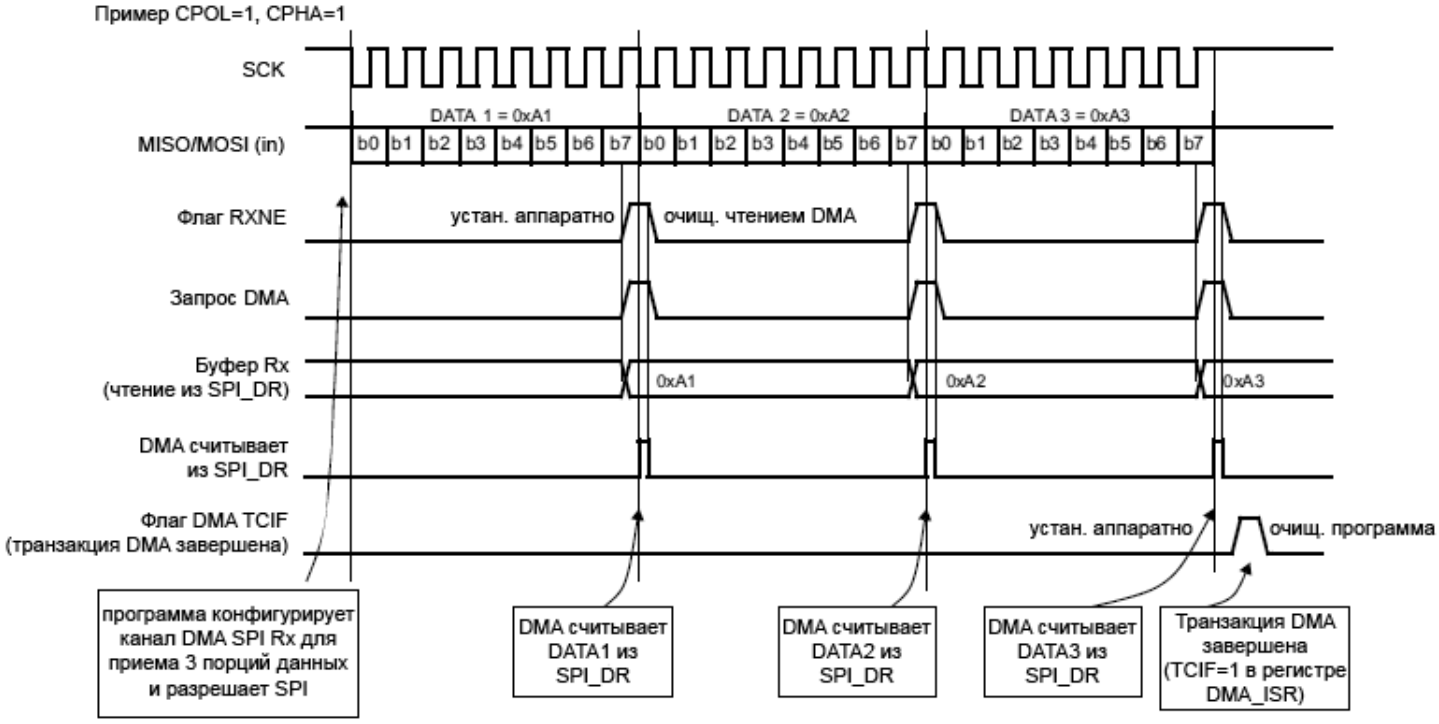

Рис. 260. Прием с помощью DMA.

DMA с использованием CRC. Когда разрешен обмен SPI с функцией CRC в режиме DMA, передача и прием CRC по окончании обмена происходят автоматически, без использования бита CRCNEXT. После приема CRC, CRC должна быть прочитана в регистре SPI_DR, чтобы очистить флаг RXNE.

По окончанию передач данных и CRC установится флаг CRCERR в регистре SPI_SR, если во время передачи обнаружилось повреждение данных (не совпала CRC на приеме).

[Флаги ошибок]

Master mode fault (MODF). Отказ режима master происходит, когда устройство master обнаружило, что на его вход NSS поступил лог. 0 (аппаратный режим NSS), или бит SSI в лог. 0 (программный режим NSS), при этом автоматически установится бит MODF. Отказ режима master влияет на периферийное устройство SPI следующим образом:

- Установится бит MODF, и генерируется прерывание SPI, если установлен бит ERRIE.
- Очистится бит SPE. Это заблокирует все выходы из устройства, и запретит интерфейс SPI.

- Очистится бит MSTR, это принудительно переведет устройство в режим slave.

Используйте следующую последовательность действий в программе, чтобы очистить бит MODF:

1. Выполните доступ на чтение или запись к регистру SPI_SR, когда установился бит MODF.
2. Выполните запись в регистр SPI_CR1.

Чтобы избежать конфликтов нескольких slave-устройств в системе, состоящей из нескольких MCU, ножка NSS должна быть подтянута к лог. 1 во время процедуры очистки бита MODF. После этой процедуры очистки биты SPE и MSTR могут быть восстановлены в их оригинальное состояние.

В целях безопасности аппаратура не позволяет установить биты SPE и MSTR, когда установлен бит MODF.

В slave-устройстве бит MODF не может быть установлен. Однако в конфигурации multimaster устройство может находиться в режиме slave с установленным битом MODF. В этом случае бит MODF показывает, что для управления системой мог произойти конфликт multimaster. Может использоваться ISR, чтобы корректно выполнить восстановление из этого состояния путем выполнения сброса или возврата в состояние по умолчанию.

Переполнение (OVR). Событие переполнения происходит, когда master отправил данные, и slave-устройство не очистило бит RXNE, который был установлен после приема предыдущих данных. Когда происходит переполнение:

- Установится бит OVR, и генерируется прерывание, если установлен бит ERRIE.

В этом случае содержимое буфера приемника не будет обновлено новыми принятыми данными, полученными от устройства master. Чтение регистра SPI_DR вернет эти данные. Все остальные переданные данные, которые прошли до чтения регистра SPI_DR, будут потеряны.

Очистка бита OVR осуществляется чтением из регистра SPI_DR, с последующим чтением регистра SPI_SR.

Ошибка CRC (CRCERR). Это флаг используется для проверки достоверности передачи данных, когда установлен бит CRCEN в регистре SPI_CR1. Флаг CRCERR в регистре SPI_SR установится, если значение контрольной суммы, принятое в регистре сдвига, не совпадает со значением регистра SPI_RXCRCR приемника.

Ошибка формата фрейма TI (FRE). Ошибка формата фрейма TI детектируется, когда появляется импульс NSS во время текущего обмена, когда SPI работает в режиме slave, и сконфигурирован для режима протокола TI. Когда произошла эта ошибка, установится флаг FRE в регистре SPI_SR. SPI не запрещается, когда происходит такая ошибка, импульс NSS игнорируется, и SPI ожидает следующего импульса NSS перед запуском новой передачи. Данные могут быть повреждены, поскольку детектирование ошибки может привести к потере 2 байт данных.

Флаг FRE очищается чтением регистра SPI_SR. Если установлен бит ERRIE, то при детектировании NSS будет вызвано прерывание. В этом случае SPI должен быть запрещен, потому что целостность данных больше не гарантируется, и обмен должен быть инициализирован заново устройством master, когда slave SPI снова будет разрешен.

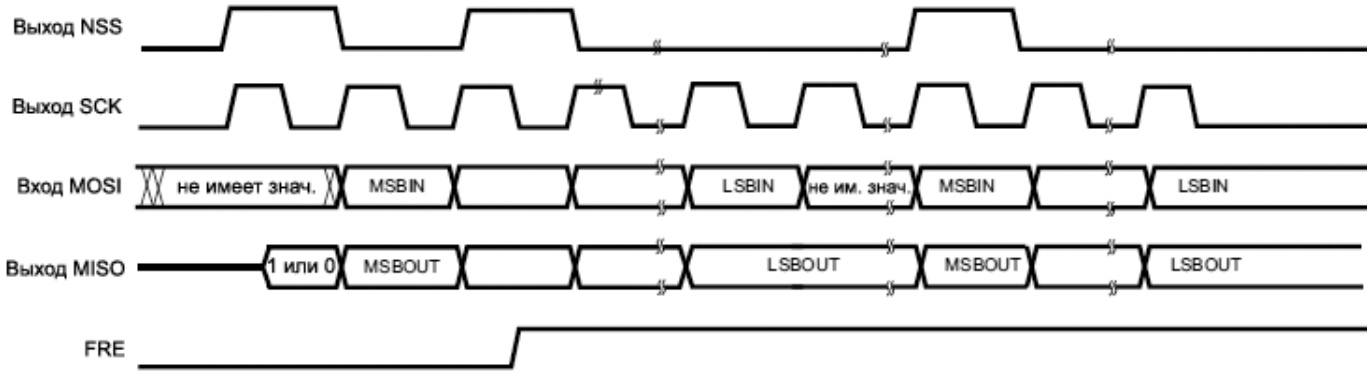

Рис. 261. Детектирование ошибки формата фрейма режима TI.

[Прерывания SPI]

Таблица 126. Запросы прерываний SPI.

Событие прерывания	**Флаг события**	**Бит разрешения прерывания**
Transmit buffer empty (буфер передачи пуст)	TXE	TXEIE
Receive buffer not empty (в буфере приема имеются готовые для чтения данные)	RXNE	RXNEIE
Master Mode fault (сбой режима master)	MODF	ERRIE
Overrun error (переполнение)	OVR	
CRC error (несовпадение контрольной суммы на приеме)	CRCERR	
TI frame format error (ошибка фрейма формата протокола TI)	FRE	

Функциональное описание I2S

SPI может функционировать как audio-интерфейс I2S, когда возможность I2S разрешена установкой бита I2SMOD в регистре SPI_I2SCFGR. Интерфейс I2S использует главным образом те же самые выводы, флаги и прерывания, что и традиционный SPI.

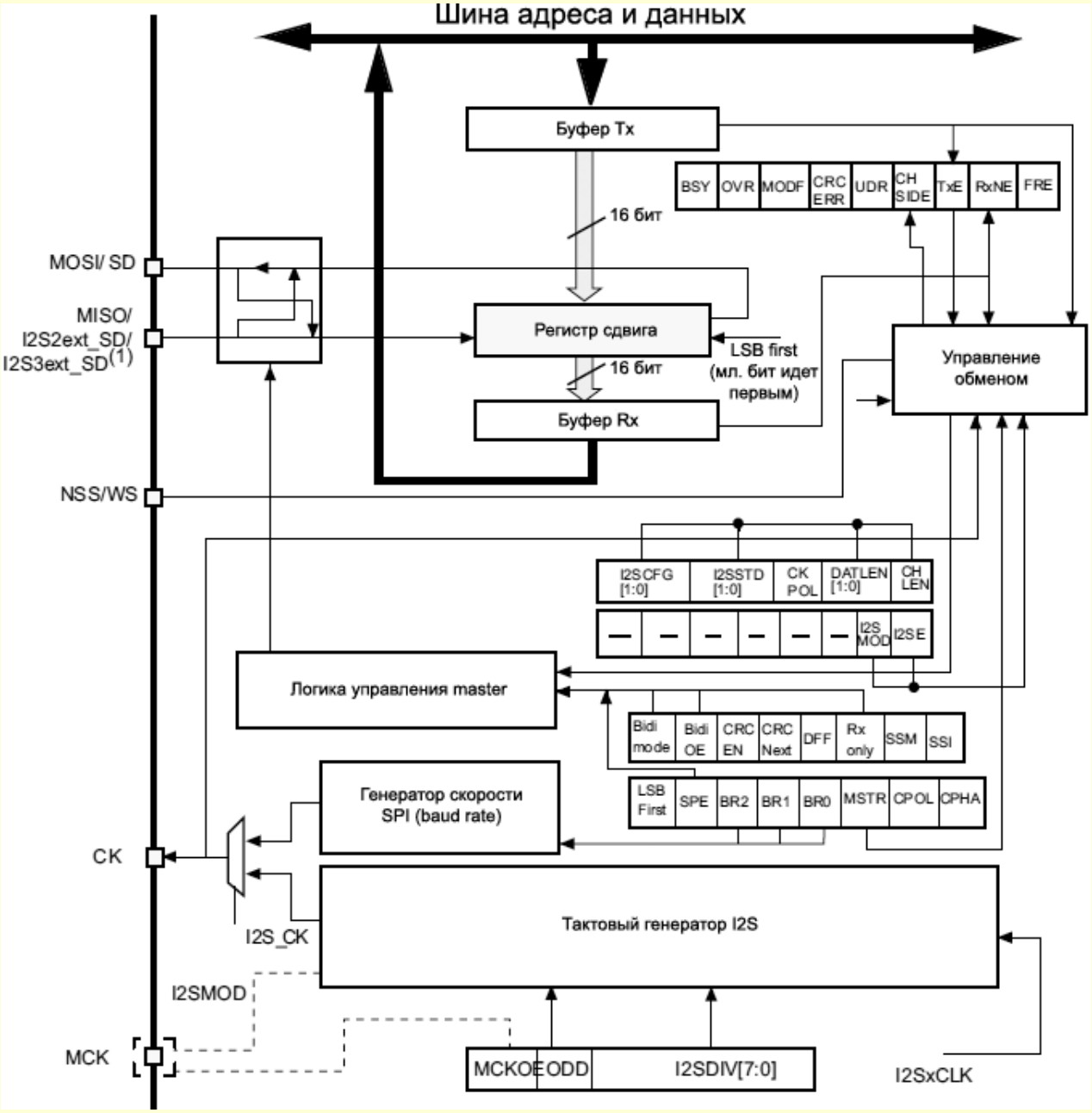

Рис. 262. Блок-схема аппаратуры периферийного устройства I2S.

Примечание (1): I2S2ext_SD и I2S3ext_SD это расширенные ножки SD, которые управляют режимом полного дуплекса I2S.

I2S совместно с SPI использует 3 общих ножки:

SD. Это последовательные данные (Serial Data), отображенные на ножку MOSI. Сигнал SD предназначен для передачи или приема двух мультиплексированных во времени канала данных (только в режиме полудуплекса).

WS. Выборка слова (Word Select), сигнал отображен на ножку NSS. WS управляет выводом данных в режиме master, и вводом данных в режиме slave.

CK. Тактовый сигнал (Serial Clock), отображенный на ножку SCK. В режиме master это выход тактов, в режиме slave вход тактов.

I2S2ext_SD и I2S3ext_SD. Это дополнительные сигналы (отображаются на ножку MISO), чтобы управлять режимом полного дуплекса I2S.

MCK. Это дополнительный сигнал, который может использоваться как выход тактов master для некоторых внешних audio-устройств (этот сигнал отображается на внешний вывод отдельно от ножек SPI). Такты MCK генерируются когда сконфигурирован режим I2S master, и когда установлен бит MCKOE в регистре SPI_I2SPR. Это дополнительный тактовый сигнал, который генерируется с частотой 256 x FS, где FS это частота выборок звука.

I2S использует свой собственный генератор тактов, чтобы формировать такты обмена, когда настроен режим master. Этот генератор тактов также служит источником выходных тактов master. В режиме I2S доступны 2 дополнительных регистра. Один из них SPI_I2SPR, связанный с конфигурацией тактового генератора, и другой SPI_I2SCFGR, традиционный регистр конфигурации I2S (определяет стандарт audio, режим slave/master, формат данных, пакет фрейма, полярность тактов и т. п.).

Регистр SPI_CR1 и все регистры CRC в режиме I2S не используются. Подобным образом не используется бит SSOE в регистре SPI_CR2 и биты MODF и CRCERR в регистре SPI_SR.

I2S использует для данных тот же регистр, что и SPI (SPI_DR), в режиме 16 бит.

[Полный дуплекс I2S]

Для поддержки I2S full duplex mode доступны 2 дополнительных экземпляра I2S, которые называют расширенными I2S (I2S2_ext, I2S3_ext), в дополнение к I2S2 и I2S3 (см. рис. 263). Первый полнодуплексный интерфейс I2S базируется на I2S2 и I2S2_ext, и второй на I2S3 и I2S3_ext.

Примечание: I2S2_ext и I2S3_ext используются только в режиме full-duplex.

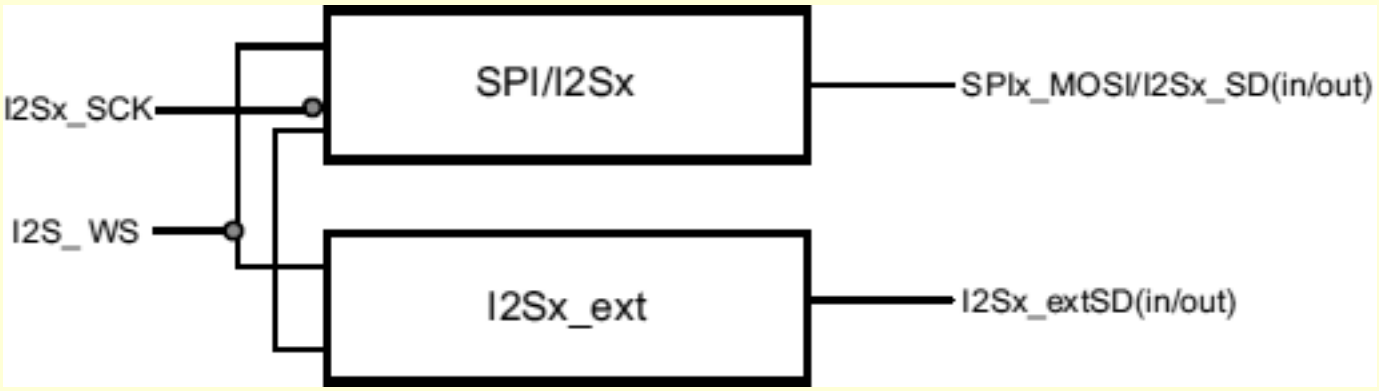

Рис. 263. Схема полного дуплекса I2S. Здесь буква x может быть заменена на 2 или 3.

I2Sx может работать в режиме master. В результате:

- *Только I2Sx может выводить SCK и WS в режиме полудуплекса.*
- *Только I2Sx может поставлять SCK и WS на I2S2_ext b I2S3_ext в режиме полного дуплекса.*

Расширенные I2S (I2Sx_ext) могут использоваться только в режиме полного дуплекса. I2Sx_ext может всегда работать в режиме slave.

Оба I2Sx и I2Sx_ext можно сконфигурировать как передатчики или как приемники.

[Поддерживаемые audio-протоколы]

Четырехпроводная шина может поддерживать только звуковые данные, которые обычно разделены во времени по 2 каналам: левый и правый. Однако существует только один 16-битный регистр (SPI_SR) для передачи и приема. Таким образом, программа должна записать в регистр данных адекватное значение, соответствующее рассматриваемому каналу, или прочитать данные из регистра данных, чтобы идентифицировать соответствующий канал путем проверки бита CHSIDE в регистре SPI_SR. Левый канал всегда отправляется первым, за ним идет правый канал (для протокола PCM бит CHSIDE не имеет никакого значения).

Доступно 4 варианта фрейма пакета. Данные могут быть отправлены в одном из форматов:

- *16-битные данные, упакованные в 16-битный фрейм*
- *16-битные данные, упакованные в 32-битный фрейм*
- *24-битные данные, упакованные в 32-битный фрейм*
- *32-битные данные, упакованные в 32-битный фрейм*

При использовании 16-битных данных, расширенных по 32-битному пакету, первые 16 бит (MSB) это значащие биты, 16-бит LSB принудительно обнуляются без каких-либо необходимых действий со стороны программы или запроса DMA (происходит только одна операция чтения/записи).

24-битные и 32-битные фреймы данных нуждаются в 2 операциях чтения или записи SPI_DR со стороны CPU, или двух операциях DMA, если приложение предпочло использовать DMA. Для 24-битного фрейма данных 8 не значащих бита аппаратно расширяются до 32 бит путем дополнения нулевыми битами.

Для всех форматов данных и стандартов обмена старший значащий бит (MSB) всегда отправляется первым.

Интерфейс I2S поддерживает 4 audio-стандарта, что конфигурируется битами I2SSTD[1:0] и PCMSYNC в регистре SPI_I2SCFGR.

[Audio-стандарты I2S]

I2S Philips. Для этого стандарта сигнал WS используется для того, чтобы показать, какой канал передается. Он активируется на 1 тактовый цикл CK перед первым доступным битом (MSB).

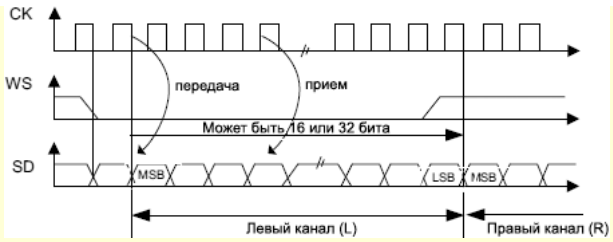

Рис. 264. Диаграммы протокола I2S Philips, полная точность 16/32 бита, CPOL=0.

Данные на выходе защелкиваются по спаду уровня CK (для передатчика), и считываются по фронту CK (для приемника). Сигнал WS также защелкивается по спаду уровня CK.

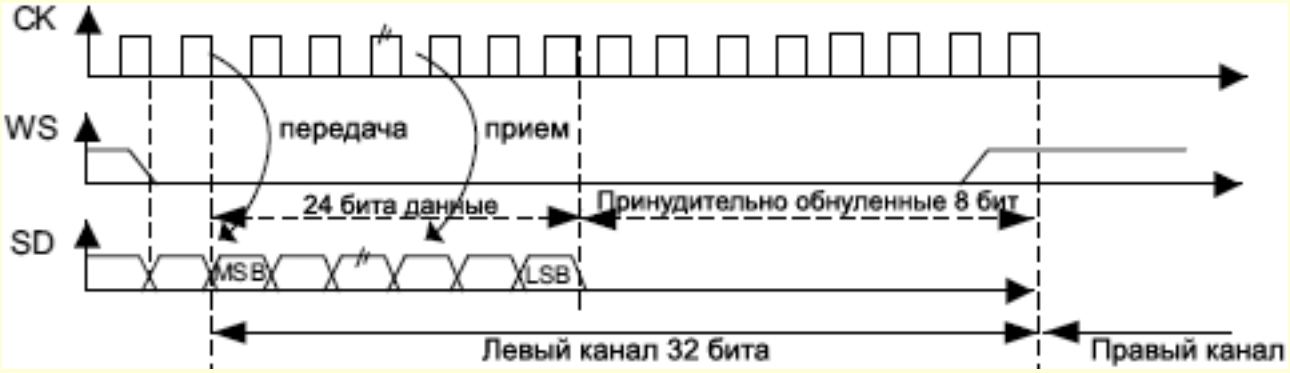

Рис. 265. Диаграммы протокола I2S Philips, 24-битный фрейм, CPOL=0.

Этот режим требует 2 операций чтения или записи с регистром SPI_DR.

- *В режиме передачи, если отправляется 0x8EAA33 (24-бита):*

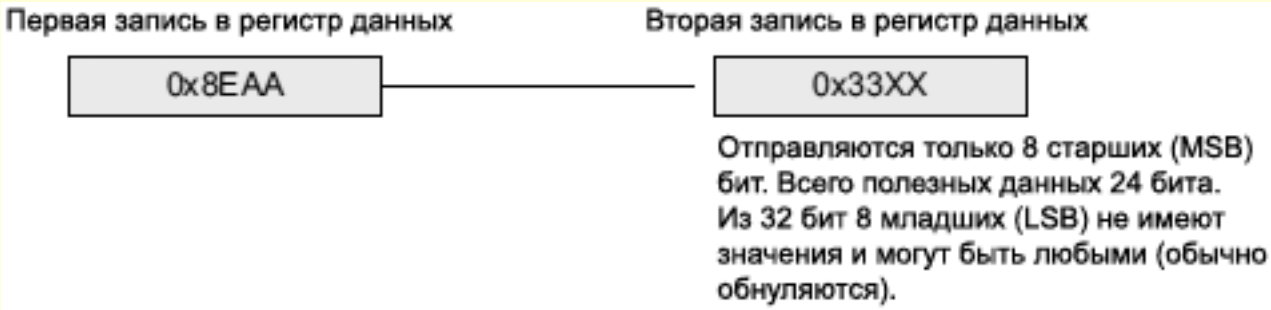

Рис. 266. Передача 0x8EAA33.

- *В режиме приема, если принимается 0x8EAA33:*

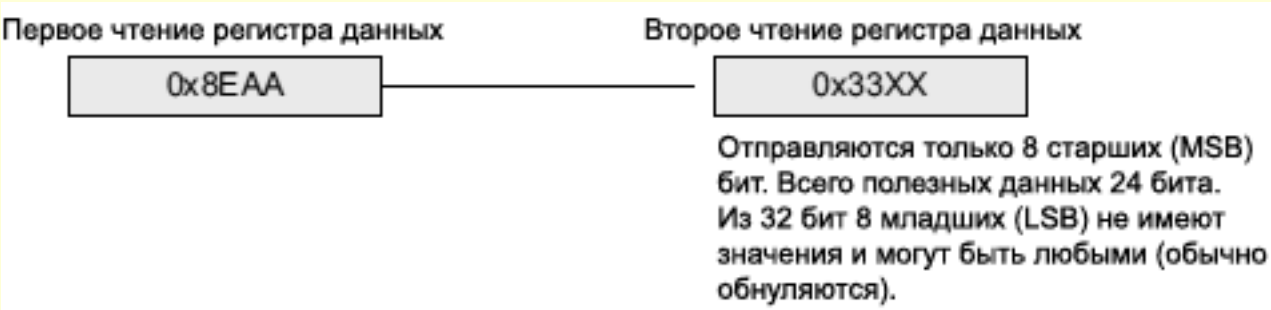

Рис. 267. Прием 0x8EAA33.

Когда 16-битный фрейм данных, расширенный до 32-битного фрейма канала, выбирается во время фазы конфигурирования I2S, требуется только один доступ к SPI_DR. 16 оставшихся бит аппаратно обнуляются в 0x0000, чтобы расширить данные до 32-битного формата.

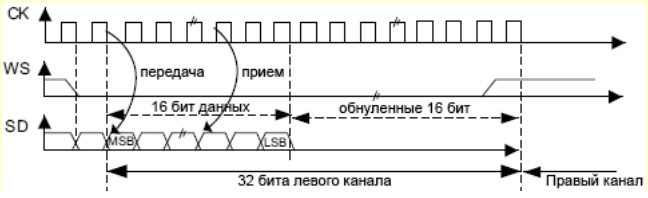

Рис. 268. Стандарт I2S Philips, 16 бит расширяются до фрейма пакета 32 бита, CPOL = 0.

Если передаваемые или принимаемые данные равны 0x76A3 (0x76A30000 расширены до 32 бит), то необходимая операция показана на рис. 269.

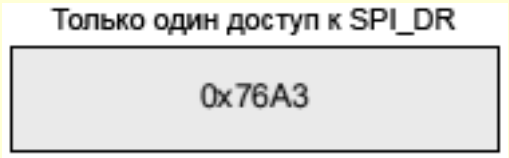

Рис. 269. Пример доступа к SPI_DR.

При передаче каждый раз, когда MSB записывается в SPI_DR, установится флаг TXE, и будет сгенерировано прерывание, если это разрешено, чтобы загрузить новое значение SPI_DR для отправки. Это происходит даже тогда, когда биты 0x0000 еще не отправлены, потому что это делается аппаратно.

Для приема установится флаг RXNE, и если это разрешено, будет сгенерировано прерывание, когда принята первая половина слова, 16 бит MSB.

Таким образом предоставляется больше времени между двумя операциями записи или чтения, что предотвращает ситуации недогрузки (underrun) или переполнения (overrun), в зависимости от направления передачи данных.

MSB justified. Для этого стандарта сигнал WS генерируется одновременно с первым битом данных MSB.

Рис. 270. MSB justified 16 или 32 бита, длина полной точности, CPOL=0.

Данные на выходе защелкиваются по спаду уровня CK (для передатчика), и считываются по фронту нарастания уровня (для приемника).

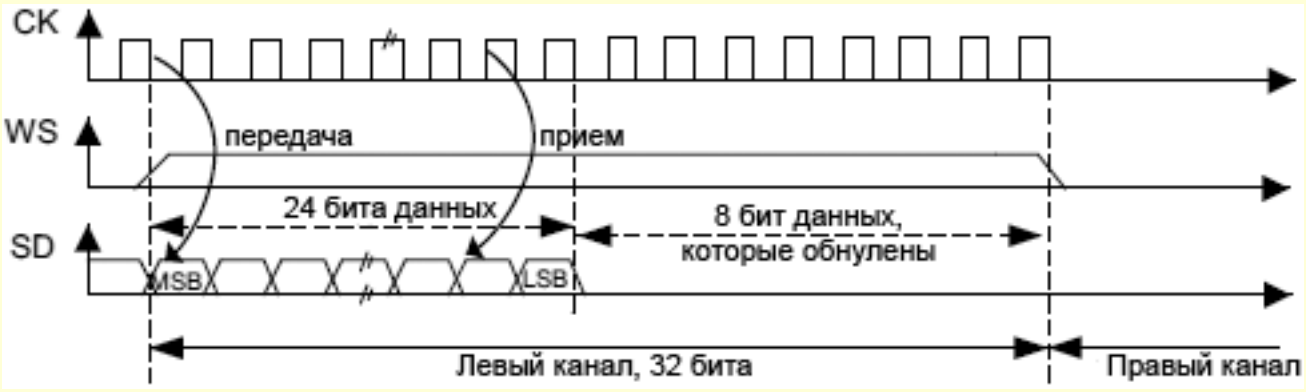

Рис. 271. MSB justified 24 бита, CPOL=0.

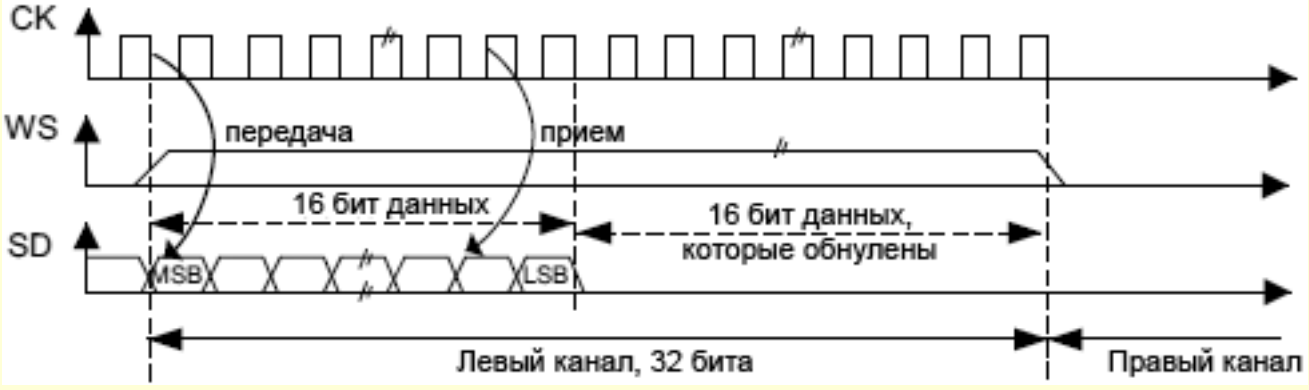

Рис. 272. MSB justified 16 бит, расширенные до 32-битного фрейма пакета, CPOL=0.

LSB justified. Этот стандарт подобен стандарту MSB justified (нет отличий для форматов 16 бит и 32 бита полной точности).

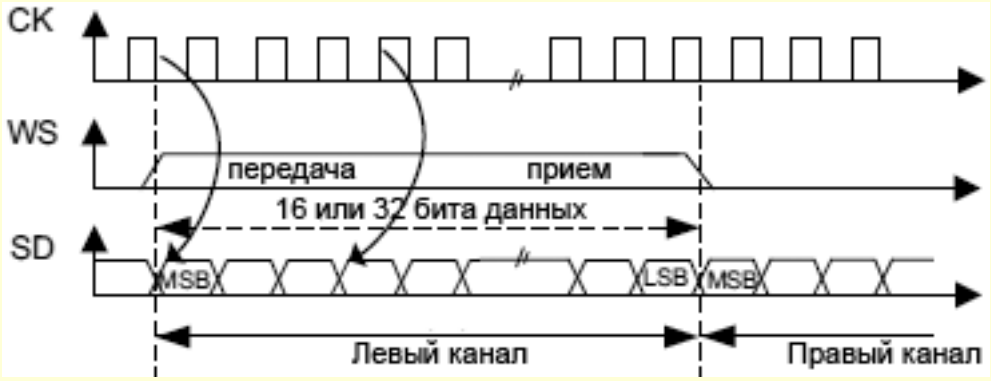

Рис. 273. LSB justified 16 бит или 32 бита полной точности, CPOL=0.

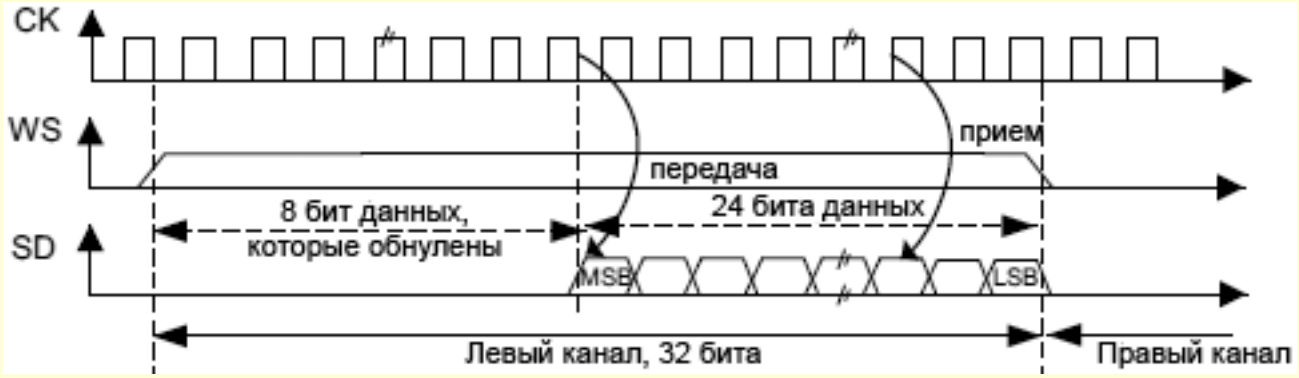

Рис. 274. LSB justified 24 бита, CPOL=0.

- *В режиме передачи: если передаются данные 0x3478AE, требуются 2 операции записи в регистр SPI_DR со стороны программы или DMA. Ниже на рис. 275 показаны эти операции.*

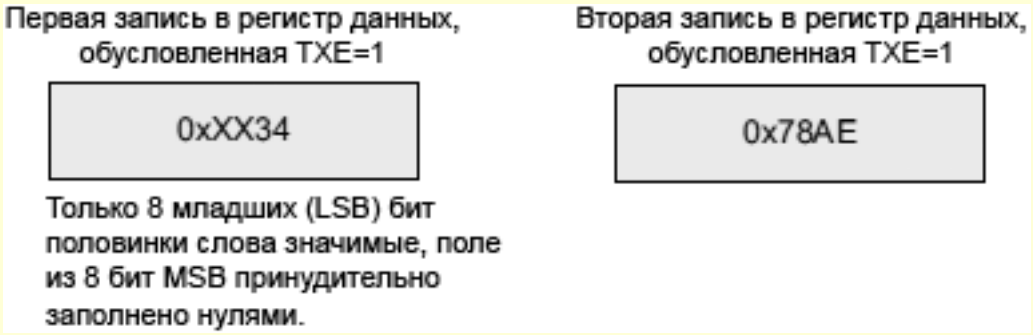

Рис. 275. Операции, необходимые для передачи 0x3478AE.

- *В режиме приема: если принимаются данные 0x3478AE, требуются 2 следующие друг за другом операции чтения регистра SPI_DR на каждом событии установки бита RXNE.*

Рис. 276. Операции, необходимые для приема 0x3478AE.

Когда 16-битный фрейм данных, расширенный до 32-битного фрейма канала, выбран на фазе конфигурации I2S, то требуется только один доступ к SPI_DR. Остальные 16 бит принудительно обнуляются аппаратурой в 0x0000, чтобы расширить данные до 32-битного формата.

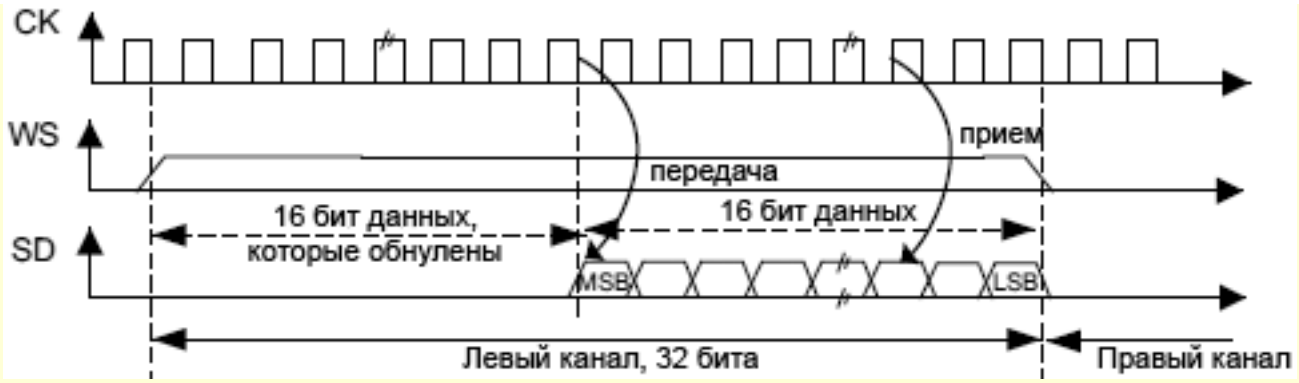

Рис. 277. LSB justified, 16-битные данные, расширенные до 32-битного фрейма пакета, CPOL = 0.

Если передаваемые или принимаемые данные 0x76A3 (0x000076A3 расширено до 32 бит), то необходимая операция показана на рис. 278.

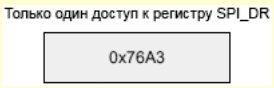

Рис. 278. Пример LSB justified 16 бит, расширенного до 32 битного пакета фрейма.

В режиме передачи, когда установлен бит TXE, приложение должно записать передаваемые данные (в нашем примере 0x76A3). Сначала передается поле 0x0000 (расширение до 32 бит). TXE установится снова, как только значимые данные (0x76A3) были отправлены по is sent on SD.

В режиме приема RXNE установится, как только было принято слово со значимыми данными (но не поле 0x0000).

Таким образом, предоставляется больше времени между двумя операциями записи или чтения, чтобы предотвратить ситуации недогрузки или переполнения.

PCM. Для стандарта PCM не нужна информация о каналах. Доступны 2 режима PCM (короткий фрейм и длинный фрейм), что конфигурируется битом PCMSYNC в регистре SPI_I2SCFGR.

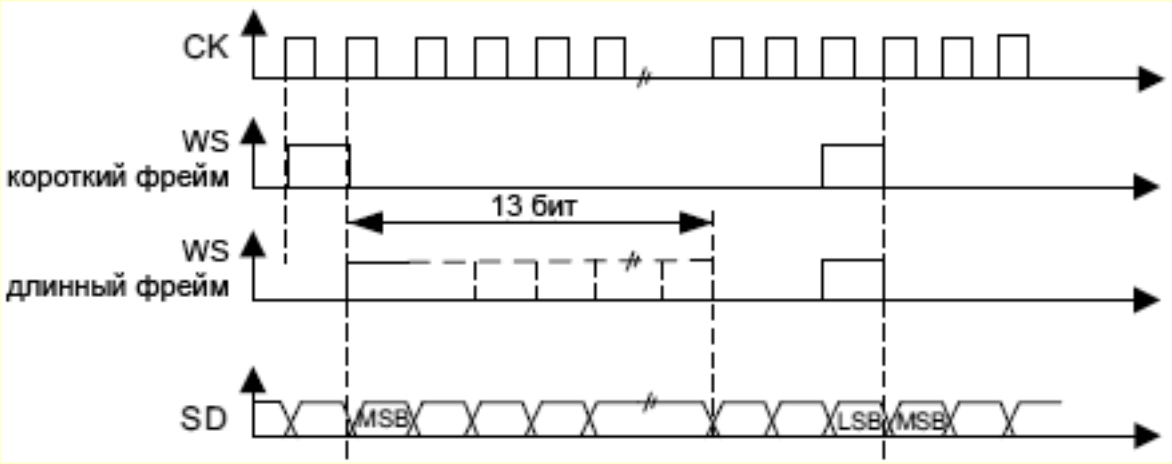

Рис. 279. Диаграмма сигналов стандарта PCM (16-бит).

Для синхронизации длинного фрейма время установки сигнала WS зафиксировано на 13 бит в режиме master. Для синхронизации короткого фрейма время сигнала синхронизации WS установлено длительностью в 1 бит.

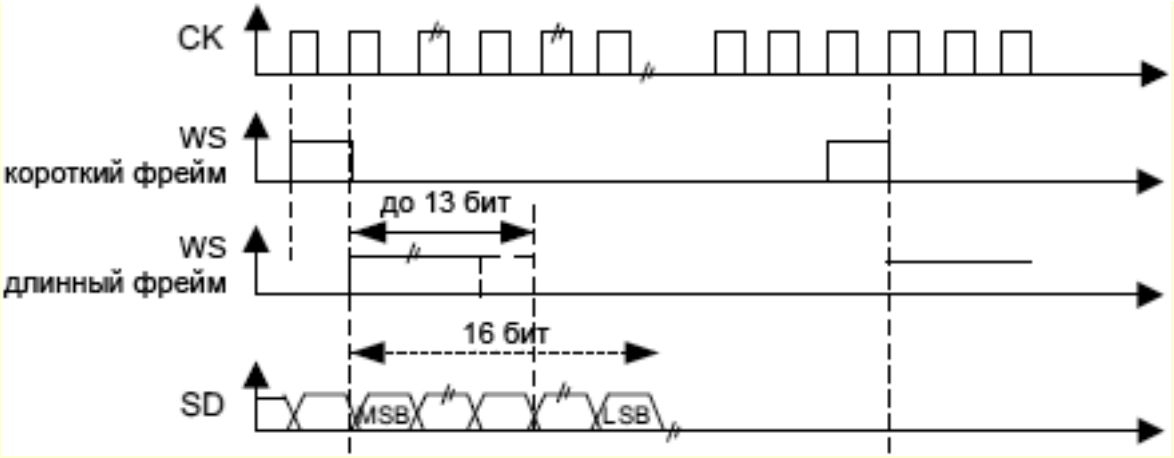

Рис. 280. Диаграмма сигналов стандарта PCM (16-бит, расширенные до 32-битного фрейма пакета).

Примечание: для обоих режимов (master и slave) и для обоих вариантов синхронизации (короткая и длинная), необходимо указать количество бит между двумя последовательными порциями данных (и таким образом между двумя сигналами синхронизации) с помощью бит DATLEN и CHLEN в регистре SPI_I2SCFGR, даже если используется режим slave.

[Генератор тактов I2S]

Скорость следования бит I2S (bitrate) определяет полосу пропускания потока на линии данных I2S и частоту тактов I2S.

I2S bitrate = количество бит на канал x количество каналов x частота выборок звука

Для 16-битного audio (16 бит на канал), с левым и правым каналом (2 канала), I2S bitrate вычисляется следующим образом:

I2S bitrate = 16 x 2 x FS

Здесь FS это частота выборок в секунду для оцифрованных данных звука. Если длина пакета 32 бита, то получится:

I2S bitrate = 32 x 2 x FS

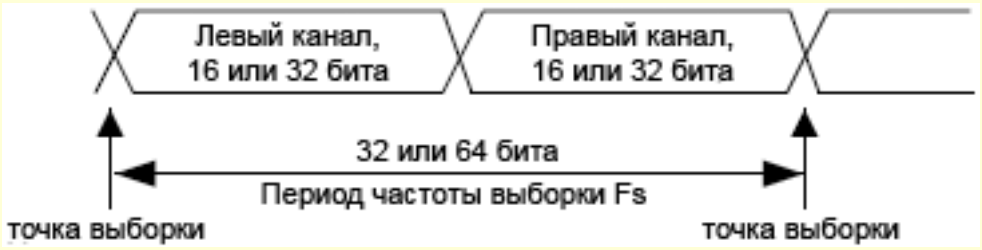

Рис. 281. Определение частоты выборок звука FS.

Когда сконфигурирован режим master, требуется предпринять специальные действия, чтобы правильно запрограммировать линейный делитель для обеспечения обмена на желаемой частоте выборок звука.

На рис. 282 представлена архитектура коммуникационного тактирования. Чтобы достичь высокого качества звука, источник тактов I2SxCLK может быть либо выходом PLLI2S (с коэффициентом деления R), либо внешней тактовой частотой (отображается на ножку I2S_CKIN).

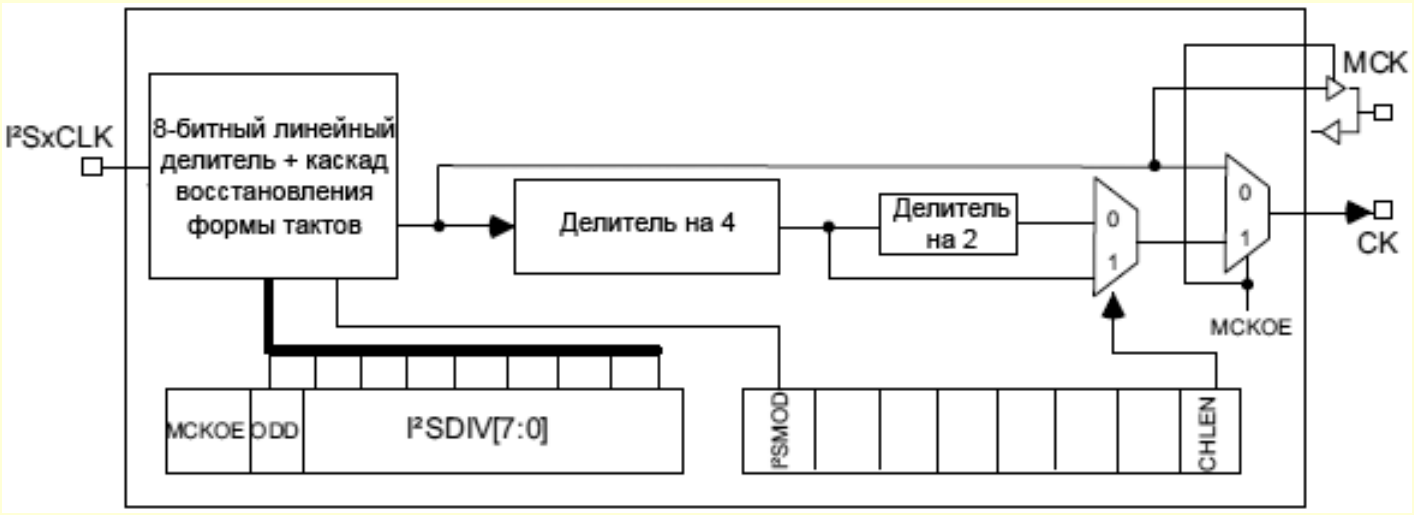

Рис. 282. Архитектура генератора тактов I2S. Здесь буква x может быть заменена на 2 или 3.

Частота выборок звука FS(audio sampling frequency) может быть выбрана 192 кГц, 96 кГц или 48 кГц. Чтобы достичь желаемой частоты, нужно запрограммировать линейный делитель в соответствии с формулами, приведенными ниже.

Когда частоту тактов генерирует master (MCKOE=1 в регистре SPI_I2SPR):

*FS = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)*8)], когда фрейм канала 16-битный*

*FS = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)*4)], когда фрейм канала 32-битный*

Когда частоту тактов master запрещена (MCKOE=0 в регистре SPI_I2SPR):

*FS = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD))], когда фрейм канала 16-битный*

*FS = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD))], когда фрейм канала 32-битный*

В таблице 127 предоставлены примеры точных значений для различных конфигураций тактирования (возможны и другие конфигурации, обеспечивающие оптимальную точность синхронизации).

Таблица 127. Точность частоты выборок звука FS (для PLLM VCO = 1 МГц или 2 МГц).

Такты MCK	Целевая F_S (Гц)	Формат данных	PLLI2SN	PLLI2SR	I2SDIV	I2SODD	Реальная F_S (Гц)	Ошибка
Запрещены	8000	16 бит	192	2	187	1	8000	0.0000%
		32 бита	192	3	62	1	8000	0.0000%
	16000	16 бит	192	3	62	1	16000	0.0000%
		32 бита	256	2	62	1	16000	0.0000%
	32000	16 бит	256	2	62	1	32000	0.0000%
		32 бита	256	5	12	1	32000	0.0000%
	48000	16 бит	192	5	12	1	48000	0.0000%
		32 бита	384	5	12	1	48000	0.0000%
	96000	16 бит	384	5	12	1	96000	0.0000%
		32 бита	424	3	11	1	96014.49219	0.0151%
	22050	16 бит	290	3	68	1	22049.87695	0.0006%
		32 бита	302	2	53	1	22050.23438	0.0011%
	44100	16 бит	302	2	53	1	44100.46875	0.0011%
		32 бита	429	4	19	0	44099.50781	0.0011%
	192000	16 бит	424	3	11	1	192028.9844	0.0151%
		32 бита	258	3	3	1	191964.2813	0.0186%
Разрешены	8000	Не имеет значения	256	5	12	1	8000	0.0000%
	16000		213	2	13	0	16000.60059	0.0038%
	32000		213	2	6	1	32001.20117	0.0038%
	48000		258	3	3	1	47991.07031	0.0186%
	96000		344	2	3	1	95982.14063	0.0186%
	22050		429	4	9	1	22049.75391	0.0011%
	44100		271	2	6	0	44108.07422	0.0183%

[Режим I2S master]

I2S может быть сконфигурирован следующим образом:

- *Режим master для передачи или приема, полудуплекс с использованием I2Sx.*
- *Режим master для передачи или приема, полный дуплекс с использованием I2Sx и I2Sx_ext).*

Это означает, что последовательные такты генерируются master на выводе CK, как и сигнал выборки слова WS (Word Select). Частота MCK может выводиться или нет, в зависимости от бита MCKOE в регистре SPI_I2SPR.

Процедура программирования:

1. *Выберите значение бит I2SDIV[7:0] регистра SPI_I2SPR, чтобы определить частоту последовательных тактов, обеспечивающих нужную частоту выборок звука. Также должен быть определен бит ODD в регистре SPI_I2SPR.*
2. *Выберите бит CKPOL, чтобы определить уровень стабильности (когда данные не передаются) для тактов обмена. Установите бит MCKOE в регистре SPI_I2SPR, если частота тактов MCK должна быть предоставлена для внешних звуковых компонентов DAC/ADC (значения I2SDIV и ODD должны быть вычислены в зависимости от состояния выхода MCK, см. выше секцию "Генератор тактов I2S").*
3. *Установите бит I2SMOD в регистре SPI_I2SCFGR, чтобы активировать функционал I2S и выберите стандарт I2S битами I2SSTD[1:0] и PCMSYNC, длину данных битами DATLEN[1:0] и количество бит на канал конфигурированием*

бита CHLEN. Выберите также режим I2S master и направление данных (передатчик или приемник) битами I2SCFG[1:0] регистра SPI_I2SCFGR.

4. *Если необходимо, выберите все потенциальные источники прерывания и функции DMA записью регистра SPI_CR2.*
5. *Бит I2SE в регистре SPI_I2SCFGR должен быть установлен.*

WS и CK конфигурируются в режиме выхода. MCK также работает как выход, если установлен бит MCKOE в регистре SPI_I2SPR.

Передача. Последовательность передачи начинается, когда половина слова записана в буфер Tx.

Предполагается, что первые данные, записанные в буфер Tx, соответствуют данным левого канала. Когда данные передались из буфера Tx в регистр сдвига, установится TXE, и данные, соответствующие правому каналу, должны быть записаны в буфер Tx. Флаг CHSIDE показывает, какой канал передается. Это имеет значение, когда флаг TXE установлен, потому что флаг CHSIDE обновляется, когда TXE переходит в лог. 1.

Полный фрейм должен рассматриваться как передача данных левого канала, за которым следует передача данных правого канала. Не бывает частичный фрейм, где отправлен только левый канал.

Половина слова данных параллельно загружается в 16-битный регистр сдвига во время передачи первого бита, и затем данные последовательно выдвигаются на вывод MOSI/SD, старший бит (MSB) идет первым. Флаг TXE установится после каждого перемещения данных из буфера Tx в регистр сдвига, и в этот момент может быть сгенерировано прерывание, если установлен бит TXEIE в регистре SPI_CR2.

Подробнее про операции записи в зависимости от выбранного стандартного режима I2S см. выше секцию "Поддерживаемые audio-протоколы I2S".

Чтобы гарантировать непрерывную передачу данных звука, важно успевать записывать в SPI_DR следующую порцию данных по окончании текущей передачи.

Чтобы выключить I2S путем очистки I2SE, важно подождать TXE=1 и BSY=0 (точно так же, как это требуется для запрета SPI).

Прием. Рабочий режим такой же, как и для передачи, кроме шага 3, где битами I2SCFG[1:0] должна быть установлена конфигурация режима master приема.

Независимо от длины данных или длины канала, аудиоданные принимаются 16-битными пакетами. Это значит, что каждый раз, когда буфер Rx заполнен очередной порцией данных, установится бит RXNE, и будет сгенерировано прерывание, если установлен бит RXNEIE в регистре SPI_CR2. В зависимости от конфигурации длины данных и канала, поступление значения выборки звука для левого или правого канала может привести к одному или двум приемам данных в буфере Rx.

Очистка бита RXNE выполняется чтением регистра SPI_DR.

Бит CHSIDE обновляется на каждом приеме. Это чувствительно для сигнала WS, генерируемого блоком I2S.

Более подробно про операции чтения в зависимости от выбранного стандарта I2S см. выше секции "Поддерживаемые audio-протоколы I2S" и "Audio-стандарты I2S".

Если данные были приняты, когда предыдущие принятые данные не были пока прочитаны, генерируется переполнение с установкой флага ошибки OVR. Если в регистре SPI_CR2 установлен бит ERRIE, будет сгенерировано прерывание, сигнализирующее об ошибке.

Для выключения I2S требуются специальные действия, чтобы гарантировать, что I2S корректно завершил цикл передачи, без запуска новой передачи данных. Эта последовательность выключения зависит от длин данных и канала, и от выбранного режима звукового протокола. Возможны следующие варианты.

- *16-битные данные, расширенные на 32-битную длину канала (DATLEN=00, CHLEN=1), режим LSB justified (I2SSTD=10):*
  - *a) Подождать предпоследнюю установку RXNE=1 (принята выборка n–1 звука).*
  - *b) Затем подождать 17 периодов тактов I2S (с помощью программного цикла).*
  - *c) Запретить I2S (I2SE=0).*
- *16-битные данные, расширенные на 32-битную длину канала (DATLEN=00, CHLEN=1), режимы MSB justified, I2S или PCM (соответственно I2SSTD=00, I2SSTD=01 или I2SSTD=11:*
  - *a) Подождать последнюю установку RXNE=1.*
  - *b) Затем подождать 1 период тактов I2S (с помощью программного цикла).*
  - *c) Запретить I2S (I2SE=0).*
- *Для всех других комбинаций DATLEN и CHLEN, независимо от выбранного битами I2SSTD режима звука, выполните следующую последовательность действий для выключения I2S:*
  - *a) Подождать предпоследнюю установку RXNE=1 (принята выборка n–1 звука).*
  - *b) Затем подождать 1 период тактов I2S (с помощью программного цикла).*
  - *c) Запретить I2S (I2SE=0).*

Примечание: во время передач флаг BSY должен удерживаться в состоянии лог. 0.

[Режим I2S slave]

I2S может быть сконфигурирован следующим образом:

- *Режим slave для передачи или приема, полудуплекс с использованием I2Sx.*

- *Режим slave для передачи или приема, полный дуплекс с использованием I2Sx и I2Sx_ext).*

Настройка режима в основном следует тем же правилам, описанным выше для конфигурации I2S master. В режиме slave интерфейс I2S не генерирует такты, сигнал тактов и сигнал WS поступают от внешнего master. Таким образом, пользователю не нужно конфигурировать тактовые сигналы I2S.

Процедура конфигурирования:

1. *Установите бит I2SMOD в регистре SPI_I2SCFGR, чтобы достичь функционала I2S, и выберите стандарт I2S битами I2SSTD[1:0], длину данных битами DATLEN[1:0] и количество бит на канал для фрейма битом CHLEN. Также выберите режим (передача или прием) для slave битами I2SCFG[1:0] в регистре SPI_I2SCFGR.*
2. *Если необходимо, выберите все потенциальные источники прерывания и функции DMA записью регистра SPI_CR2.*
3. *Бит I2SE в регистре SPI_I2SCFGR должен быть установлен.*

Передача. Последовательность передачи начнется, когда внешний master device пошлет тактовый сигнал, и когда сигнал NSS_WS запросит передачу данных. Устройство slave должно быть разрешено до того, как внешний master начнет обмен данными. Регистр данных I2S должен быть загружен до того, как master инициирует обмен.

Для режимов MSB justified и LSB justified, первая записанная в регистр SPI_DR порция данных соответствует левому каналу. Когда начнется обмен, данные переместятся из буфера Tx в регистр сдвига. Затем установится флаг TXE, чтобы запросить данные правого канала, которые нужно записать в регистр данных I2S (регистр SPI_DR).

Флаг CHSIDE показывает, какой канал передается. В сравнении с режимом передачи master, в режиме slave бит CHSIDE чувствителен к сигналу WS, поступающему снаружи от внешнего master. Это значит, что для устройства slave нужно быть готовым передать первые данные до того, как master начнет генерировать такты. Установка WS=1 соответствует первой передаче левого канала.

Примечание: I2SE должен быть записан как минимум на 2 такта PCLK до того, как на сигнал CK начнут поступать такты от master.

Половина слова данных параллельно загружается (по внутренней шине) в 16-битный регистр сдвига во время передачи первого бита, и далее последовательно выдвигается наружу на ножку MOSI/SD, при этом старший бит (MSB) идет первым. Установится флаг TXE после каждого перемещения данных из буфера Tx в регистр сдвига, при этом генерируется прерывание, если установлен бит TXEIE в регистре SPI_CR2.

Обратите внимание, что флаг TXE должен быть проверен на наличие лог. 1 перед любой попыткой записи в буфер Tx (в регистр SPI_DR).

Более подробно про операции чтения в зависимости от выбранного стандарта I2S см. выше секции "Поддерживаемые audio-протоколы I2S" и "Audio-стандарты I2S".

Чтобы обеспечить непрерывную передачу данных, важно успевать записывать регистр SPI_DR следующей порцией данных для передачи до момента окончания передачи текущей порции. Установится флаг недогрузки (underrun) и может быть сгенерировано прерывание, если данные не были вовремя записаны в регистр SPI_DR перед первым тактовым перепадом следующего обмена данными. Это укажет программному обеспечению на то, что переданные данные неверны. Если установлен бит ERRIE в регистре SPI_CR2, то будет сгенерировано прерывание, когда в регистре SPI_SR установится в лог. 1 флаг недогрузки UDR. В этом случае важно выключить I2S и перезапустить передачу данных, начиная с левого канала (хотя необходимые действия могут зависеть от требований приложения).

Для выключения I2S путем очистки бита I2SE важно подождать TXE=1 и BSY=0 (точно так же, как это делается для запрета SPI).

Прием. Рабочий режим настраивается так же, как и передача, кроме отличий на шаге 1 (см. вышеописанную процедуру в секции "Режим I2S slave. Передача"), где должен быть установлен режим приема slave битами I2SCFG[1:0] регистра SPI_I2SCFGR.

Независимо от длины данных или длины канала, звуковые данные принимаются пакетами по 16 бит. Это значит, что каждый раз, когда заполняется буфер Rx, устанавливается флаг RXNE в регистре SPI_SR, и генерируется прерывание, если установлен бит RXNEIE в регистре SPI_CR2. В зависимости конфигурации от длины данных и длины канала, значение звука, принятое для левого или правого канала, может требовать приема одной или двух порций 16-битных данных в буфер Rx.

Флаг CHSIDE обновляется всякий раз, когда принята новая порция данных, которую нужно прочитать через регистр SPI_DR. Этот флаг чувствителен к внешнему сигналу WS, который поступает от внешнего master.

Очистка бита RXNE производится чтением регистра SPI_DR.

Более подробно про операции чтения в зависимости от выбранного стандарта I2S см. выше секции "Поддерживаемые audio-протоколы I2S" и "Audio-стандарты I2S".

Если приняты новые данные, когда ранее принятые данные еще не были прочитаны, то генерируется переполнение (overrun), установится флаг OVR. Если был установлен бит ERRIE в регистре SPI_CR2, будет сгенерировано прерывание, сигнализирующее об ошибке.

Чтобы выключить I2S в режиме приема, бит I2SE должен быть очищен немедленно после получения последней установки бита RXNE=1.

Примечание: у внешнего master должна быть возможность отправить/принять данные пакетами по 16 или 32 бита через звуковой канал.

[Флаги статуса I2S]

Приложению предоставлены 3 флага статуса, чтобы оно могло отслеживать состояние шины I2S.

Занятость (BSY). Флаг BSY устанавливается и очищается аппаратно (запись в этот флаг не дает никакого эффекта). Он показывает состояние слоя физического обмена I2S.

Когда BSY=1, это показывает занятость шины I2S обменом данными. Здесь есть одно исключение - в режиме приема master (I2SCFG = 11), когда флаг BSY удерживается во время приема в состоянии лог. 0.

Флаг BSY полезен для детектирования окончания передачи, если программе нужно запретить I2S. Это позволяет избежать повреждения последней передачи. Для корректного запрета I2S настоятельно рекомендуется выполнить описанную ниже процедуру. Итак, флаг BSY установится в лог. 1, когда запустилась передача, кроме режима I2S master receiver.

Флаг BSY очищается:

- *Когда передача завершилась (кроме режима master transmit, если подразумевается непрерывная передача данных).*
- *Когда I2S запрещен.*

Когда обмен непрерывный:

- *В режиме master transmit флаг BSY остается в лог. 1 во время всех передач.*
- *В режиме slave флаг BSY переходит в лог. 0 на 1 период тактов I2S между каждой передачей.*

Примечание: не используйте флаг BSY для обработки каждой передачи данных или каждого приема. Лучше для этой цели использовать флаги TXE и RXNE.

Буфер передачи пуст (TXE). Будучи установленным, этот флаг показывает, что буфер Tx пуст, и готов к загрузки в него новых данных для передачи. Флаг TXE сбросится, когда Tx уже содержит передаваемые данные, которые были переданы в регистр сдвига. Также флаг TXE сбросится, кода I2S запрещен (I2SE=0).

Буфер приема не пуст (RXNE). Будучи установленным, этот флаг показывает, что в буфере Rx находятся принятые данные, которые еще не были прочитаны через регистр SPI_DR. Флаг RXNE сбросится, когда считывается регистр SPI_DR.

Флаг левого или правого канала (CHSIDE). В режиме передачи этот флаг обновляется всякий раз, когда TXE переходит в лог. 1. CHSIDE показывает, какому каналу принадлежат данные, передаваемые через сигнал SD. В случае ошибки недогрузки (underrun) в режиме передачи slave этот флаг ненадежен, и нужно выключить I2S и снова его включить для возобновления корректного обмена данными.

В режиме приема флаг CHSIDE обновляется, когда данные приняты в SPI_DR. Флаг CHSIDE показывает, для какого канала были приняты данные. Обратите внимание, что в случае ошибки (наподобие переполнения OVR) значение этого флага становится бессмысленным, и I2S должен быть сброшен путем запрета и повторного разрешения (с процедурой конфигурирования, если необходимо что-то поменять).

Флаг CHSIDE не имеет значения для стандарта PCM (для обоих режимов короткого и длинного фрейма).

Когда установлен флаг OVR или UDR в регистре SPI_SR, и при этом также установлен бит ERRIE в регистре SPI_CR2, генерируется прерывание. Это прерывание может быть очищено чтением регистра статуса SPI_SR (после очистки источника прерывания).

[Флаги ошибки I2S]

Всего имеется 3 флага ошибки I2S.

Недогрузка (UDR). В режиме передачи slave этот флаг установится при поступлении первого тактового импульса от master, если предварительно не были загружены передаваемые данные путем записи в регистр SPI_DR. Флаг доступен когда установлен бит I2SMOD в регистре SPI_I2SCFGR. Может быть сгенерировано прерывание, если установлен бит ERRIE в регистре SPI_CR2.

Бит UDR очищается операцией чтения регистра SPI_SR.

Переполнение (OVR). Этот флаг установится, когда приняты данные, и ранее принятые данные еще не были прочитаны через регистр SPI_DR. В результате новые поступившие данные окажутся потерянными. Может быть сгенерировано прерывание, если установлен бит ERRIE в регистре SPI_CR2.

В этом случае содержимое буфера приема не обновляется новыми принятыми данными, которые поступили от внешнего передатчика. Операция чтения регистра SPI_DR вернет предыдущие корректно принятые данные. Все другие переданные друг за другом данные окажутся потерянными.

Очистка флага OVR осуществляется операцией чтения регистра SPI_DR с последующим чтением регистра SPI_SR.

Ошибка фрейма (FRE). Этот флаг установится аппаратно, если I2S сконфигурирован в режиме slave. FRE установится, если внешний master поменял сигнал WS в момент, когда slave не ожидал этого изменения. Если синхронизация потеряна, то для восстановления синхронизации с внешним master выполните следующие шаги:

1. *Запретите I2S.*
2. *Заново разрешите его, когда на сигнале WS был определен корректный уровень (WS=1 в режиме I2S, или WS=0 для режимов MSB justified, LSB justified или PCM).*

Рассинхронизация между устройствами master и slave может произойти из-за шума во внешнем окружении, появившемся как сигнал помехи на сигнале тактов CK или сигнале WS. Может быть сгенерировано прерывание, если установлен бит ERRIE. Флаг рассинхронизации (FRE) очищается чтением регистра статуса.

[Прерывания I2S]

Таблица 128. Запросы прерывания I2S.

Событие прерывания	Флаг события	Бит разрешения прерывания
Transmit buffer empty (буфер передачи пуст)	TXE	TXEIE
Receive buffer not empty (в буфере приема имеются готовые для чтения данные)	RXNE	RXNEIE
Overrun error (переполнение)	OVR	ERRIE
Underrun error (недогрузка)	UDR	
Frame error (ошибка фрейма)	FRE	

[Функции DMA для I2S]

DMA работает точно так же, как и в режиме SPI, для I2S нет никаких отличий. Однако в режиме I2S недоступна функция CRC, поскольку для I2S нет системы защиты/проверки целостности данных.

[Регистры SPI и I2S]

К регистрам периферийного устройства SPI осуществляется доступ через половинку слова (16 бит) или через слово (32 бита), хотя все регистры 16-битные.

Таблица 129. Карта регистров SPI и их значение после сброса.

Смещ.	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
0x00	SPI_CR1	зарезервировано																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	*LSBF/IRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Значение сброса																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CR2	зарезервировано																								TXEIE	RXNEIE	ERRIE	FRF	зарезерв.	*SSOE*	TXDMAEN	RXDMAEN
	Значение сброса																									0	0	0	0		0	0	0
0x08	SPI_SR	зарезервировано																							FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Значение сброса																								0	0	0	0	0	0	0	1	0
0x0C	SPI_DR	зарезервировано																DR[15:0]															
	Значение сброса																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPI_CRCPR	зарезервировано																CRCPOLY[15:0]															
	Значение сброса																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	SPI_RXCRCR	зарезервировано																RxCRC[15:0]															
	Значение сброса																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPI_TXCRCR	зарезервировано																TxCRC[15:0]															
	Значение сброса																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPI_I2SCFGR	зарезервировано																				I2SMOD	I2SE	I2SCFG		PCMSYNC	зарезерв.	I2SSTD		CKPOL	DATLEN		CHLEN
	Значение сброса																					0	0	0	0	0		0	0	0	0	0	0
0x20	SPI_I2SPR	зарезервировано																						MCKOE	ODD	I2SDIV							
	Значение сброса																							0	0	0	0	0	0	0	0	1	0

Смещение адреса указано относительно базового адреса, который можно узнать в даташите на микроконтроллер, либо из его файла заголовка. Например, для STM32F427 и STM32F429 базовые адреса приведены в таблице ниже.

SPIx	Базовый адрес
SPI1	0x40013000
SPI2 / I2S2	0x40003800
SPI3 / I2S3	0x40003C00
SPI4	0x40013400
SPI5	0x40015000
SPI6	0x40015400

Буква x в имени регистра заменяется на цифру, обозначающую номер интерфейса SPI. Например для STM32F429 цифра может меняться на значения от 1 до 6 (соответствует интерфейсам SPI1 .. SPI6).

Условные обозначения функций бит

В описании функций регистров используются следующие сокращения:

read/write (rw) Программа может читать и записывать эти биты.

read-only (r) Программа может только читать эти биты.

write-only (w) Программа может только записывать в этот бит. Чтение бита вернет значение сброса.

read/clear (rc_w1) Программа может прочитать бит, а также сбросить его путем записью 1. Запись 0 не дает никакого эффекта.

read/clear (rc_w0) Программа может прочитать бит, а также сбросить его путем записью 0. Запись 1 не дает никакого эффекта.

read/clear by read (rc_r) Программа может прочитать этот бит. Чтение этого бита автоматически сбросит его в 0. Запись 0 в бит не дает никакого эффекта.

read/set (rs) Программа может прочитать, а также установить этот бит. Запись 0 в бит не дает никакого эффекта.

read-only write trigger (rt_w) Программа может прочитать этот бит. Запись 0 или 1 вызовет появление события (триггер), но не окажет никакого влияния на значение бита.

toggle (t) Программа может только переключить этот бит записью 1. Запись 0 не дает никакого эффекта.

Reserved (Res.) Зарезервированный бит, его значение должно сохраняться на значении сброса.

SPI control register 1 (SPIx_CR1)

Этот регистр управления не используется в режиме I2S.

Смещение адреса: 0x00

Значение сброса: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDIMODE rw	**BIDIOE** rw	**CRCEN** rw	**CRCNEXT** rw	**DFF** rw	**RXONLY** rw	**SSM** rw	**SSI** rw	**LSBFIRST** rw	**SPE** rw	**BR[2:0]** rw			**MSTR** rw	**CPOL** rw	**CPHA** rw

BIDIMODE (бит 15): BIDIrectional data MODe Enable, разрешение двунаправленного режима работы по одной линии данных.

0: 2-проводный однонаправленный режим обмена данными.
1: 1-проводный двунаправленный режим обмена данными.

BIDIOE (бит 14): Output enable in bidirectional mode, разрешение выхода в двунаправленном режиме. Этот бит в комбинации с битом BIDIMODE выбирает направление передачи в двунаправленном режиме.

0: выход запрещен (только прием).
1: выход разрешен (только передача).

В режиме master используется ножка MOSI, в то время как в режиме slave используется ножка MISO.

CRCEN (бит 13): Hardware CRC calculation Enable.

0: вычисление CRC и его проверка запрещены.
1: CRC разрешена.

Примечание: для корректного функционирования этот бит должен быть записан только когда SPI запрещен (SPE=0).

CRCNEXT (бит 12): CRC transfer NEXT, следующий элемент данных несет в себе CRC.

0: фаза данных (не фаза CRC).
1: следующая передача или прием несут в себе CRC (фаза CRC).

Примечание: когда SPI сконфигурирован для полного дуплекса, или для режимов только передачи бит CRCNEXT должен быть записан сразу после записи последней порции данных в регистр SPI_DR. Когда SPI сконфигурирован только в режиме приема, бит CRCNEXT должен быть установлен после предпоследней принимаемой порции данных. Этот бит должен оставаться сброшенным, когда транзакциями управляет DMA.

DFF (бит 11): Data Frame Format, формат данных фрейма.

0: для передачи/приема выбран 8-битный формат фрейма.
1: для передачи/приема выбран 16-битный формат фрейма.

Примечание: для корректного функционирования этот бит должен записываться только когда SPI запрещен(SPE=0).

RXONLY (бит 10): Receive only, только прием. Этот бит в комбинации с битом BIDIMODE выбирает направление данных в однонаправленном 2-проводном режиме. Этот бит также полезен в системе с несколькими slave-устройствами. Когда к определенному slave не осуществляется доступ, не будут повреждены выходные данные другого slave, к которому доступ осуществляется.

0: полный дуплекс (одновременные передача и прием).

1: выход запрещен (режим "только прием").

SSM (бит 9): Software Slave Management. Когда бит SSM установлен, действие уровня на входе выборки NSS заменяется на действие значения бита SSI.

0: программное управление slave-устройством запрещено (выборкой slave-устройства управляет внешний сигнал NSS).

1: программное управление slave-устройством разрешено (выборкой slave-устройства управляет бит SSI).

SSI (бит 8): Slave Select Internal. Этот бит работает только когда установлен бит SSM. Тогда значение бита SSI действует как выборка NSS, и внешний уровень ножки NSS игнорируется.

LSBFIRST (бит 7): формат фрейма в контексте порядка следования старшинства бит.

0: старший бит (MSB) в потоке данных идет первым.

1: младший бит (LSB) в потоке данных идет первым.

Примечание: этот бит не должен изменяться во время обмена данными.

SPE (бит 6): SPI Enable, разрешение работы SPI. Этот бит не влияет на режим работы I2S.

0: периферийное устройство SPI запрещено.

1: периферийное устройство SPI разрешено.

Для корректного запрета SPI следуйте процедуре, описанной выше в разделе "Запрет SPI".

BR[2:0] (биты 5:3): Baud Rate control, управление скоростью передачи.

000: fPCLK/2
001: fPCLK/4
010: fPCLK/8
011: fPCLK/16
100: fPCLK/32
101: fPCLK/64
110: fPCLK/128
111: fPCLK/256

Примечание: эти биты не должны изменяться во время обмена данными.

MSTR (бит 2): MaSTeR selection, выбор режима работы - главное или подчиненное устройство шины SPI.

0: конфигурация Slave.

1: конфигурация Master.

Примечание: этот бит не должен изменяться во время обмена данными.

CPOL (бит 1): Clock POLarity, полярность тактов.

0: CK в лог. 0 в режиме ожидания.

1: CK в лог. 1 в режиме ожидания.

Примечание: этот бит не должен изменяться во время обмена данными.

CPHA (бит 0): Clock PHAse, фаза тактов.

0: первый перепад тактов соответствует первому захвату уровня данных.

1: второй перепад тактов соответствует первому захвату уровня данных.

Примечание: этот бит не должен изменяться во время обмена данными

SPI control register 2 (SPIx_CR2)

Смещение адреса: 0x04

Значение сброса: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
зарезервировано								**TXEIE** rw	**RXNEIE** rw	**ERRIE** rw	**FRF** rw	зарез.	**SSOE** rw	**TXDMAEN** rw	**RXDMAEN** rw

Биты 15:8 зарезервированы и должны удерживаться в состоянии сброса (все нули).

TXEIE (бит 7): Tx buffer Empty Interrupt Enable, разрешение прерывания при опустошении буфера передачи.

0: прерывание TXE маскировано (запрещено).

1: прерывание TXE разрешено. Будет сгенерирован запрос прерывания, когда установится флаг TXE.

RXNEIE (бит 6): RX buffer Not Empty Interrupt Enable, разрешение прерывания при появлении новых данных в буфере приема.

0: прерывание RXNE маскировано (запрещено).

1: прерывание RXNE разрешено. Будет сгенерирован запрос прерывания, когда установится флаг RXNE.

ERRIE (бит 5): ERRor Interrupt Enable, разрешение прерывания ошибки. Этот бит управляет генерацией прерывания, когда произошло одно из событий ошибки (флаги CRCERR, OVR, MODF в режиме SPI, флаг FRE в режиме TI и флаги UDR, OVR, FRE в режиме I2S).

0: прерывание ошибки маскировано (запрещено).

1: прерывание ошибки разрешено.

FRF (бит 4): FRame Format, формат фрейма.

0: режим SPI Motorola (один из 4 режимов SPI, выбираемый битами CPOL и CPHA).

1: режим SPI TI (полярность и фаза тактов фиксированы, биты CPOL и CPHA не оказывают влияния).

Примечание: этот бит в режиме I2S не используется.

Бит 3 зарезервирован, и аппаратно удерживается в состоянии сброса (лог. 0).

SSOE (бит 2): SS Output Enable, разрешение выхода выборки slave-устройства.

0: выход SS запрещен в режиме master, и узел SPI может работать в конфигурации с несколькими устройствами master на шине (multimaster).

1: выход SS разрешен в режиме master, и когда узел SPI разрешен. SPI не может работать в конфигурации с несколькими устройствами master на шине (multimaster).

Примечание: этот бит не используется в режимах I2S и SPI TI.

TXDMAEN (бит 1): TX buffer DMA ENable, разрешение буфера DMA для передачи. Когда этот бит установлен, делается запрос DMA всякий раз, когда установлен флаг TXE.

0: буфер Tx DMA запрещен.

1: буфер Tx DMA разрешен.

RXDMAEN (бит 0): RX buffer DMA ENable, разрешение буфера DMA для приема. Когда этот бит установлен, делается запрос DMA всякий раз, когда установлен флаг RXNE.

0: буфер Rx DMA запрещен.

1: буфер Rx DMA разрешен.

SPI status register (SPIx_SR)

Смещение адреса: 0x08
Значение сброса: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
зарезервировано							**FRE** r	**BSY** r	**OVR** r	**MODF** r	**CRCERR** rc_w0	**UDR** r	**CHSIDE** r	**TXE** r	**RXNE** r

Биты 15:9 зарезервированы и должны удерживаться в состоянии сброса (все нули).

FRE (бит 8): FRame format Error, ошибка формата фрейма.

0: не было ошибки формата фрейма.

1: произошла ошибка формата фрейма.

Этот флаг устанавливается аппаратно и очищается программой, когда считывается значение регистра SPI_SR.

Примечание: флаг FRE используется, когда SPI работает в режиме TI slave или I2S slave (см. раздел "Флаги ошибок").

BSY (бит 7): BuSY flag, флаг занятости интерфейса.

0: SPI (или I2S) не занят.

1: SPI (или I2S) занят обменом, или буфер Tx не пуст. Этот флаг устанавливается и очищается аппаратурой.

Примечание: флаг BSY должен использоваться с осторожностью, см. разделы "Флаги статуса" и "Запрет SPI".

OVR (бит 6): OVerRun flag, флаг переполнения.

0: не было переполнения.

1: произошло переполнение.

Этот флаг устанавливается аппаратурой, и сбрасывается последовательностью действий в программе (см. раздел "Флаги ошибок").

MODF (бит 5): MODe Fault, не соответствие режима.

0: не было сбоя режима.
1: произошел сбой режима.

Этот флаг устанавливается аппаратурой, и сбрасывается последовательностью действий в программе (см. раздел "Флаги ошибок").

Примечание: этот бит не используется в режиме I2S.

CRCERR (бит 4): CRC ERRor flag, флаг ошибки контрольной суммы.

0: принятое значение CRC совпало со значением SPI_RXCRCR.
1: принятое значение CRC не совпало со значением SPI_RXCRCR.

Флаг CRCERR устанавливается аппаратурой, и сбрасывается программой при записи в него лог. 0.

Примечание: этот бит не используется в режиме I2S.

UDR (бит 3): UnDerRun flag, флаг недогрузки данных.

0: не было недогрузки.
1: произошла недогрузка.

Этот флаг устанавливается аппаратурой, и сбрасывается последовательностью действий в программе (см. раздел "Флаги ошибок").

Примечание: этот бит не используется в режиме SPI.

CHSIDE (бит 2): CHannel SIDE, левый или правый канал.

0: передается или принимается левый канал (L).
1: передается или принимается левый канал (R).

Примечание: этот бит не используется в режиме SPI и не имеет смысла в режиме PCM.

TXE (бит 1): Transmit buffer empty, буфер передачи пуст.

0: буфер Tx не пуст.

1: буфер Tx пуст.

RXNE (бит 0): Receive buffer not empty, в буфере приема находятся принятые, но пока не считанные данные.

0: буфер Rx пуст.

1: буфер Rx не пуст.

SPI data register (SPIx_DR)

Смещение адреса: 0x0C

Значение сброса: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0] rw															

DR[15:0] (биты 15:0): Data Register, регистр данных, принятые или передаваемые данные.

Регистр данных внутри микроконтроллера делится на 2 буфера - один для записи (буфер передачи, Tx), и другой для чтения (буфер чтения, Rx). Запись в регистр данных приведет к записи в буфер Tx, и чтение из регистра данных вернет значение, которое находится в буфере Rx.

Примечания, применимые для режима SPI:

1. *1. В зависимости от бита выбора формата данных фрейма (бит DFF в регистре SPI_CR1), данные отправляются или принимаются либо как 8 бит, либо 16 бит. Этот выбор должен быть сделан перед разрешением SPI, чтобы обеспечить корректность корректную работу SPI.*
2. *2. Для 8-битного фрейма данных буферы имеют размер только 8 бит, и для передачи/приема используется младшая (LSB) половинка регистра данных (SPI_DR[7:0]). В режиме приема старшая половинка (MSB) регистра данных (SPI_DR[15:8]) принудительно сбрасывается в 0.*
3. *3. Для 16-битного формата данных фрейма используются 16-битные буферы, поэтому для передачи/приема задействуется регистр данных целиком (SPI_DR[15:0]).*

SPI CRC polynomial register (SPIx_CRCPR)

Этот регистр не используется в режиме I2S.

Смещение адреса: 0x10

Значение сброса: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0] rw															

CRCPOLY[15:0] (биты 15:0): CRC POLYnomial register. Этот регистр определяет полином для вычисления CRC. Значение полинома 0007h используется по умолчанию после сброса. Если необходимо, то может быть сконфигурирован любой другой полином.

Примечание: эти биты не используются в режиме I2S.

SPI RX CRC register (SPIx_RXCRCR)

Этот регистр не используется в режиме I2S.

Смещение адреса: 0x14

Значение сброса: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0] r															

RXCRC[15:0] (биты 15:0): RX CRC register, регистр контрольной суммы приема.

Когда разрешено аппаратное вычисление CRC, биты RxCRC[15:0] содержат вычисленное значение контрольной суммы, проведенное над ранее последовательно принятыми данными. Этот регистр сбрасывается, когда бит CRCEN в регистре SPI_CR1 записан в лог. 1. CRC вычисляется последовательно, по полиному, запрограммированному в регистре SPI_CRCPR.

Разрядность контрольной суммы зависит от установленной длины фрейма. Когда используется 8-битный формат фрейма (DFF=0 в регистре SPI_CR1), в регистре RXCRC используются только младшие 8 бит, где вычисляется контрольная сумма по стандарту CRC8. Все 16 бит регистра RXCRC используются в том случае, когда формат фрейма 16-битный (DFF=1 в регистре SPI_CR1), в этом случае используется стандарт CRC16.

Примечание: чтение этого регистра, когда установлен флаг BSY, приведет к возврату некорректного значения.

SPI TX CRC register (SPIx_TXCRCR)

Этот регистр не используется в режиме I2S.

Смещение адреса: 0x18

Значение сброса: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0] r															

TXCRC[15:0] (биты 15:0): TX CRC register, регистр контрольной суммы передачи.

Когда вычисление CRC разрешено, биты TxCRC содержат вычисленное значение CRC от последовательно переданных байт. Этот регистр сбрасывается, когда в бит CRCEN регистра SPI_CR1 записана 1. CRC вычисляетя последовательно, по полиному, запрограммированному в регистре SPI_CRCPR.

Разрядность контрольной суммы зависит от установленной длины фрейма. Когда используется 8-битный формат фрейма (DFF=0 в регистре SPI_CR1), в регистре TXCRC используются только младшие 8 бит, где вычисляется контрольная сумма по стандарту CRC8. Все 16 бит регистра TXCRC используются в том случае, когда формат фрейма 16-битный (DFF=1 в регистре SPI_CR1), в этом случае используется стандарт CRC16.

Примечание: чтение этого регистра, когда установлен флаг BSY, приведет к возврату некорректного значения.

SPI/I2S configuration register (SPIx_I2SCFGR)

Смещение адреса: 0x1C

Значение сброса: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
зарезервировано				**I2SMOD** rw	**I2SE** rw	**I2SCFG** rw		**PCMSYNC** rw	зарез.	**I2SSTD** rw		**CKPOL** rw	**DATLEN** rw		**CHLEN** rw

Биты 15:12 зарезервированы и должны удерживаться в состоянии сброса (все нули).

I2SMOD (бит 11): I2S MODe selection, выбор режима работы I2S.

0: выбран режим SPI.
1: выбран режим I2S.

Примечание: этот бит должен быть сконфигурирован, когда SPI или I2S запрещен.

I2SE (бит 10): I2S Enable, разрешение интерфейса I2S.

0: периферийное устройство I2S запрещено.
1: периферийное устройство I2S разрешено.

Примечание: в режиме SPI этот бит не используется.

I2SCFG (биты 9:8): I2S ConFiGuration mode, конфигурация режима I2S.

00: Slave – передача.
01: Slave – прием.
10: Master – передача.
11: Master - прием.

Примечание: эти биты должны быть сконфигурированы, когда I2S запрещен. Они не используются в режиме SPI.

PCMSYNC (бит 7): синхронизация фрейма PCM.

0: синхронизация короткого фрейма.
1: синхронизация длинного фрейма.

Примечание: этот бит имеет значение только когда I2SSTD=11 (используется стандарт PCM). В режиме SPI бит PCMSYNC не используется.

Бит 6 зарезервирован, и аппаратно удерживается в значении лог. 0.

I2SSTD (биты 5:4): I2S STanDard selection, выбор стандарта I2S.

00: I2S Philips.
01: MSB justified (выравнивание данных влево, в сторону старших разрядов).
10: LSB justified (выравнивание данных вправо, в сторону младших разрядов).
11: PCM.

Более подробно про стандарты см. врезку "Функциональное описание I2S", раздел "Audio-стандарты I2S". Биты I2SSTD в режиме SPI не используются.

Примечание: для корректного функционирования биты I2SSTD должны конфигурироваться, когда интерфейс I2S запрещен.

CKPOL (бит 3): Steady state clock polarity, полярность тактов в режиме ожидания.

0: в состоянии ожидания I2S такты находятся в лог. 0.

1: в состоянии ожидания I2S такты находятся в лог. 1.

Примечание: для корректного функционирования этот бит должен быть сконфигурирован, когда I2S запрещен. Бит CKPOL не используется в режиме SPI.

DATLEN (биты 2:1): DATa LENgth, длина передаваемых данных.

00: длина данных 16 бит.
01: длина данных 24 бита.
10: длина данных 32 бита.
11: недопустимое значение.

Примечание: для корректного функционирования эти биты должны быть сконфигурированы, когда I2S запрещен. Биты DATLEN не используются в режиме SPI.

CHLEN (бит 0): CHannel LENgth, длина канала (количество бит, используемых для аудиоканала).

0: 16 бит.
1: 32 бита.

Запись в эти биты имеет смысл только если DATLEN=00, иначе длина канала фиксирована аппаратно на 32 бита, независимо от заполненного значения. CHLEN в режиме SPI не используется.

Примечание: для корректного функционирования этот бит должен быть сконфигурирован, когда I2S запрещен.

SPI/I2S prescaler register (SPIx_I2SPR)

Смещение адреса: 0x20

Значение сброса: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
зарезервировано						**MCKOE** rw	**ODD** rw	**I2SDIV** rw							

Биты 15:10 зарезервированы и должны удерживаться в состоянии сброса (все нули).

MCKOE (бит 9): Master ClocK Output Enable, разрешение вывода мастер-частоты тактов для внешнего аудиоустройства.

0: выход MCK запрещен.

1: выход MCK разрешен.

Примечание: этот бит должен быть сконфигурирован, когда I2S запрещен. Он используется только когда I2S настроен в режиме master. Этот бит не используется в режиме SPI.

ODD (бит 8): коэффициент нечетности (ODD factor) для прескалера.

*0: реальное значение делителя = I2SDIV*2*
*1: реальное значение делителя = (I2SDIV*2)+1*

См. раздел "Генератор тактов I2S" во врезке "Функциональное описание I2S". Бит ODD не используется в режиме SPI.

Примечание: этот бит должен быть сконфигурирован, когда I2S запрещен. Бит используется, когда I2S работает в режиме master.

I2SDIV (биты 7:0): линейный прескалер I2S.

Значения 0 и 1 являются запрещенными. См. раздел "Генератор тактов I2S" во врезке "Функциональное описание I2S". Биты I2SDIV не используется в режиме SPI.

Примечание: биты I2SDIV должны конфигурироваться, когда I2S запрещен. Они используются только когда I2S работает в режиме master.

Альтернативные функции ножек портов STM32F429, относящиеся к SPI

Здесь в качестве примера приведено соответствие ножек портов аппаратным сигналам SPI, которые могут быть присвоены ножкам портов как альтернативная функция (AF). Альтернативные функции SPI других микроконтроллеров STM32 см. в их даташите.

[SPI1]

Сигнал	GPIO	AF	Другие AF
NSS	PA4	AF5	SPI3_NSS/I2S3_WS, USART2_CK, OTG_HS_SOF, DCMI_HSYNC, LCD_VSYNC, EVENTOUT
SCK	PA5		TIM2_CH1/TIM2_ETR, TIM8_CH1N, OTG_HS_ULPI_CK, EVENTOUT
MISO	PA6		TIM1_BKIN, TIM3_CH1, TIM8_BKIN, TIM13_CH1, DCMI_PIXCLK, LCD_G2, EVENTOUT
MOSI	PA7		TIM1_CH1N, TIM3_CH2, TIM8_CH1N, TIM14_CH1, ETH_MII_RX_DV/ETH_RMII_CRS_DV, EVENTOUT
NSS	PA15		JTDI, TIM2_CH1/TIM2_ETR, SPI3_NSS/I2S3_WS, EVENTOUT
SCK	PB3		JTDO/TRACESWO, TIM2_CH2, SPI3_SCK/I2S3_CK, EVENTOUT
MISO	PB4		NJTRST, TIM3_CH1, SPI3_MISO, I2S3ext_SD, EVENTOUT
MOSI	PB5		TIM3_CH2, I2C1_SMBA, SPI3_MOSI/I2S3_SD, CAN2_RX, OTG_HS_ULPI_D7, ETH_PPS_OUT, FMC_SDCKE1, DCMI_D10, EVENTOUT

[SPI2]

Сигнал	GPIO	AF	Другие AF
NSS	PB9	AF5	TIM4_CH4, TIM11_CH1, I2C1_SDA, CAN1_TX, SDIO_D5, DCMI_D7, LCD_B7, EVENTOUT
SCK	PB10		TIM2_CH3, I2C2_SCL, USART3_TX, OTG_HS_ULPI_D3, ETH_MII_RX_ER, LCD_G4, EVENTOUT
NSS	PB12		TIM1_BKIN, I2C2_SMBA, USART3_CK, CAN2_RX, OTG_HS_ULPI_D5, ETH_MII_TXD0/ETH_RMII_TXD0, OTG_HS_ID, EVENTOUT
SCK	PB13		TIM1_CH1N, USART3_CTS, CAN2_TX, OTG_HS_ULPI_D6, ETH_MII_TXD1/ETH_RMII_TXD1, EVENTOUT
MISO	PB14		TIM1_CH2N, TIM8_CH2N, I2S2ext_SD, USART3_RTS, TIM12_CH1, OTG_HS_DM, EVENTOUT
MOSI	PB15		RTC_REFIN, TIM1_CH3N, TIM8_CH3N, TIM12_CH2, OTG_HS_DP, EVENTOUT
MISO	PC2		I2S2ext_SD, OTG_HS_ULPI_DIR, ETH_MII_TXD2, FMC_SDNE0, EVENTOUT
MOSI	PC3		OTG_HS_ULPI_NXT, ETH_MII_TX_CLK, FMC_SDCKE0, EVENTOUT
SCK	PD3		USART2_CTS, FMC_CLK, DCMI_D5, LCD_G7, EVENTOUT

NSS	PI0		TIM5_CH4, FMC_D24, DCMI_D13, LCD_G5, EVENTOUT
SCK	PI1		FMC_D25, DCMI_D8, LCD_G6, EVENTOUT
MISO	PI2		TIM8_CH4, I2S2ext_SD, FMC_D26, DCMI_D9, LCD_G7, EVENTOUT
MOSI	PI3		TIM8_ETR, FMC_D27, DCMI_D10, EVENTOUT

[SPI3]

Сигнал	GPIO	AF	Другие AF
NSS	PA4	AF6	SPI1_NSS, USART2_CK, OTG_HS_SOF, DCMI_HSYNC, LCD_VSYNC, EVENTOUT
NSS	PA15		JTDI, TIM2_CH1/TIM2_ETR, SPI1_NSS, EVENTOUT
SCK	PB3		JTDO/TRACESWO, TIM2_CH2, SPI1_SCK, EVENTOUT
MISO	PB4		NJTRST, TIM3_CH1, SPI1_MISO, I2S3ext_SD, EVENTOUT
MOSI	PB5		TIM3_CH2, I2C1_SMBA, SPI1_MOSI, CAN2_RX, OTG_HS_ULPI_D7, ETH_PPS_OUT, FMC_SDCKE1, DCMI_D10, EVENTOUT
SCK	PC10		USART3_TX, UART4_TX, SDIO_D2, DCMI_D8, LCD_R2, EVENTOUT
MISO	PC11		USART3_RX, UART4_RX, SDIO_D3, DCMI_D4, EVENTOUT
MOSI	PC12		USART3_CK, UART5_TX, SDIO_CK, DCMI_D9, EVENTOUT
MOSI	PD6	AF5	SAI1_SD_A, USART2_RX, FMC_NWAIT, DCMI_D10, LCD_B2, EVENTOUT

[SPI4]

Сигнал	GPIO	AF	Другие AF
SCK	PE2	AF5	TRACECLK, SAI1_MCLK_A, ETH_MII_TXD3, FMC_A23, EVENTOUT
NSS	PE4		TRACED1, SAI1_FS_A, FMC_A20, DCMI_D4, LCD_B0, EVENTOUT
MISO	PE5		TRACED2, TIM9_CH1, SAI1_SCK_A, FMC_A21, DCMI_D6, LCD_G0, EVENTOUT
MOSI	PE6		TRACED3, TIM9_CH2, SAI1_SD_A, FMC_A22, DCMI_D7, LCD_G1, EVENTOUT
NSS	PE11		TIM1_CH2, FMC_D8, LCD_G3, EVENTOUT
SCK	PE12		TIM1_CH3N, FMC_D9, LCD_B4, EVENTOUT
MISO	PE13		TIM1_CH3, FMC_D10, LCD_DE, EVENTOUT
MOSI	PE14		TIM1_CH4, FMC_D11, LCD_CLK, EVENTOUT

[SPI5]

Сигнал	GPIO	AF	Другие AF
NSS	PF6	AF5	TIM10_CH1, SAI1_SD_B, UART7_Rx, FMC_NIORD, EVENTOUT
SCK	PF7		TIM11_CH1, SAI1_MCLK_B, UART7_Tx, FMC_NREG, EVENTOUT
MISO	PF8		SAI1_SCK_B, TIM13_CH1, FMC_NIOWR, EVENTOUT
MOSI	PF9		SAI1_FS_B, TIM14_CH1, FMC_CD, EVENTOUT
MOSI	PF11		FMC_SDNRAS, DCMI_D12, EVENTOUT
NSS	PH5		I2C2_SDA, FMC_SDNWE, EVENTOUT
SCK	PH6		I2C2_SMBA, TIM12_CH1, FMC_SDNE1, DCMI_D8
MISO	PH7		I2C3_SCL, ETH_MII_RXD3, FMC_SDCKE1, DCMI_D9

[SPI6]

Сигнал	GPIO	AF	Другие AF
NSS	PG8	AF5	USART6_RTS, ETH_PPS_OUT, FMC_SDCLK, EVENTOUT
MISO	PG12		USART6_RTS, LCD_B4, FMC_NE4, LCD_B1, EVENTOUT
SCK	PG13		USART6_CTS, ETH_MII_TXD0/ETH_RMII_TXD0, FMC_A24, EVENTOUT
MOSI	PG14		USART6_TX, ETH_MII_TXD1/ETH_RMII_TXD1, FMC_A25, EVENTOUT

[**Ссылки**]

1. RM0090 Reference manual STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs site:st.com.

2. Интерфейс SPI.

3. STM32F407, назначение альтернативных функций выводам GPIO.

4. STM32F429: GPIO и альтернативные функции.