

26 Serial peripheral interface/ inter-IC sound (SPI/I²S)

26.1 Introduction

The SPI/I²S interface can be used to communicate with external devices using the SPI protocol or the I²S audio protocol. SPI or I²S mode is selectable by software. SPI mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

The Inter-IC sound (I²S) protocol is also a synchronous serial communication interface. It can operate in slave or master mode with half-duplex communication. Full duplex operations are possible by combining two I²S blocks.

It can address four different audio standards including the Philips I²S standard, the MSB- and LSB-justified standards and the PCM standard.

26.1.1 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 8-bit to 16-bit transfer frame format selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to $f_{PCLK}/2$.
- Slave mode frequency up to $f_{PCLK}/2$.
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
 - CRC value can be transmitted as last byte in Tx mode
 - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- 1-byte/word transmission and reception buffer with DMA capability: Tx and Rx requests

26.1.2 SPI extended features

- SPI TI mode support

26.1.3 I2S features

- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode, overrun flag in reception mode (master and slave) and Frame Error Flag in reception and transmitter mode (slave only)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I²S protocols:
 - I²S Philips standard
 - MSB-Justified standard (Left-Justified)
 - LSB-Justified standard (Right-Justified)
 - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock can be output to drive an external audio component. Ratio is fixed at $256 \times F_S$ (where F_S is the audio sampling frequency)
- I²S (I2S1, I2S2 and I2S3) clock can be derived from an external clock mapped on the I2S_CKIN pin.

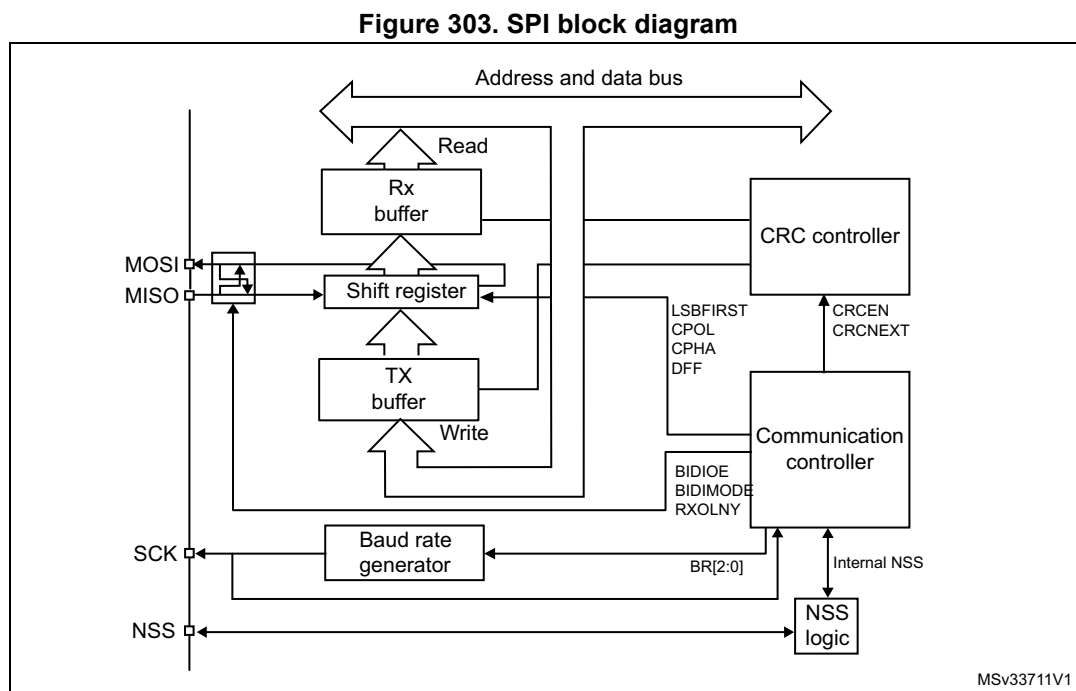
26.2 SPI/I2S implementation

This manual describes the full set of features implemented in SPI1 and SPI2.

26.3 SPI functional description

26.3.1 General description

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram [Figure 303](#).



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
 - select an individual slave device for communication
 - synchronize the data frame or
 - detect a conflict between multiple masters

See [Section 26.3.5: Slave select \(NSS\) pin management](#) for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

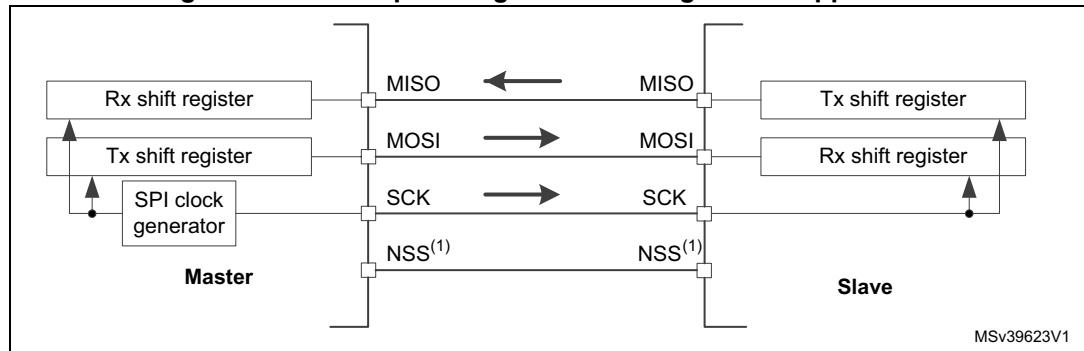
26.3.2 Communications between one master and one slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management). Communication is always initiated by the master.

Full-duplex communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 304. Full-duplex single master/ single slave application

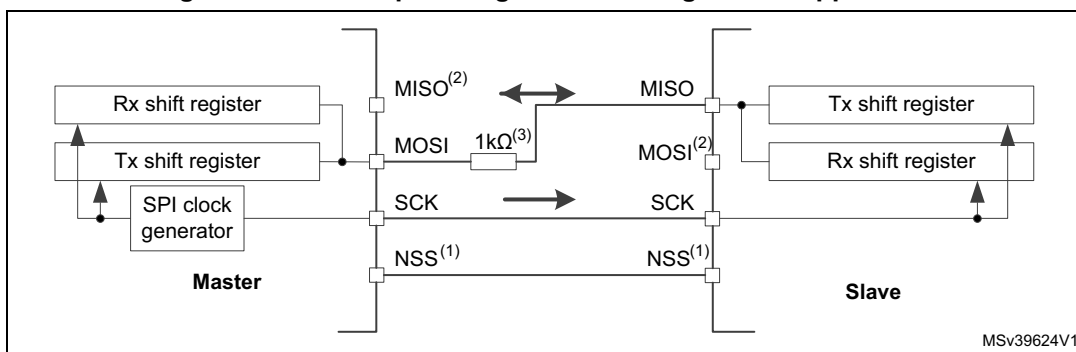


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 26.3.5: Slave select \(NSS\) pin management](#).

Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx_CR1 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

Figure 305. Half-duplex single master/ single slave application



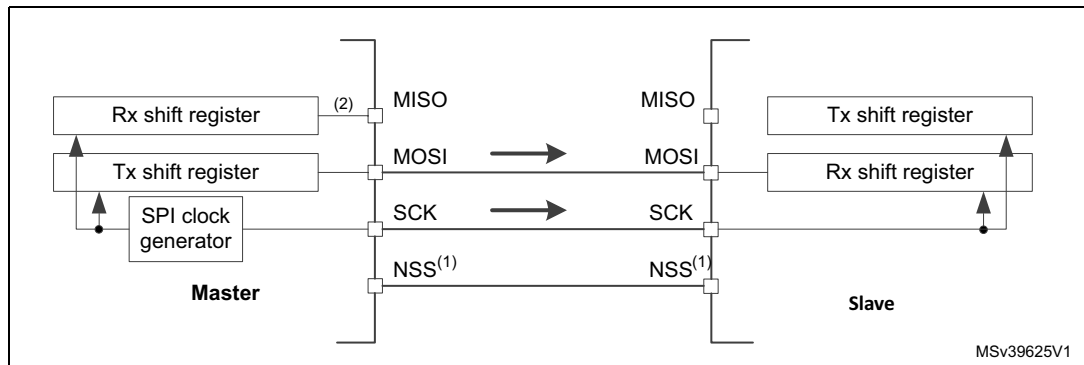
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 26.3.5: Slave select \(NSS\) pin management](#).
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.
3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectionnal mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communication data). Both nodes then fight while providing opposite output levels on the common line temporary till next node changes its direction settings correspondingly, too. It is suggested to insert a serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx_CR2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1):** The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see [26.3.5: Slave select \(NSS\) pin management](#)). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

Figure 306. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)



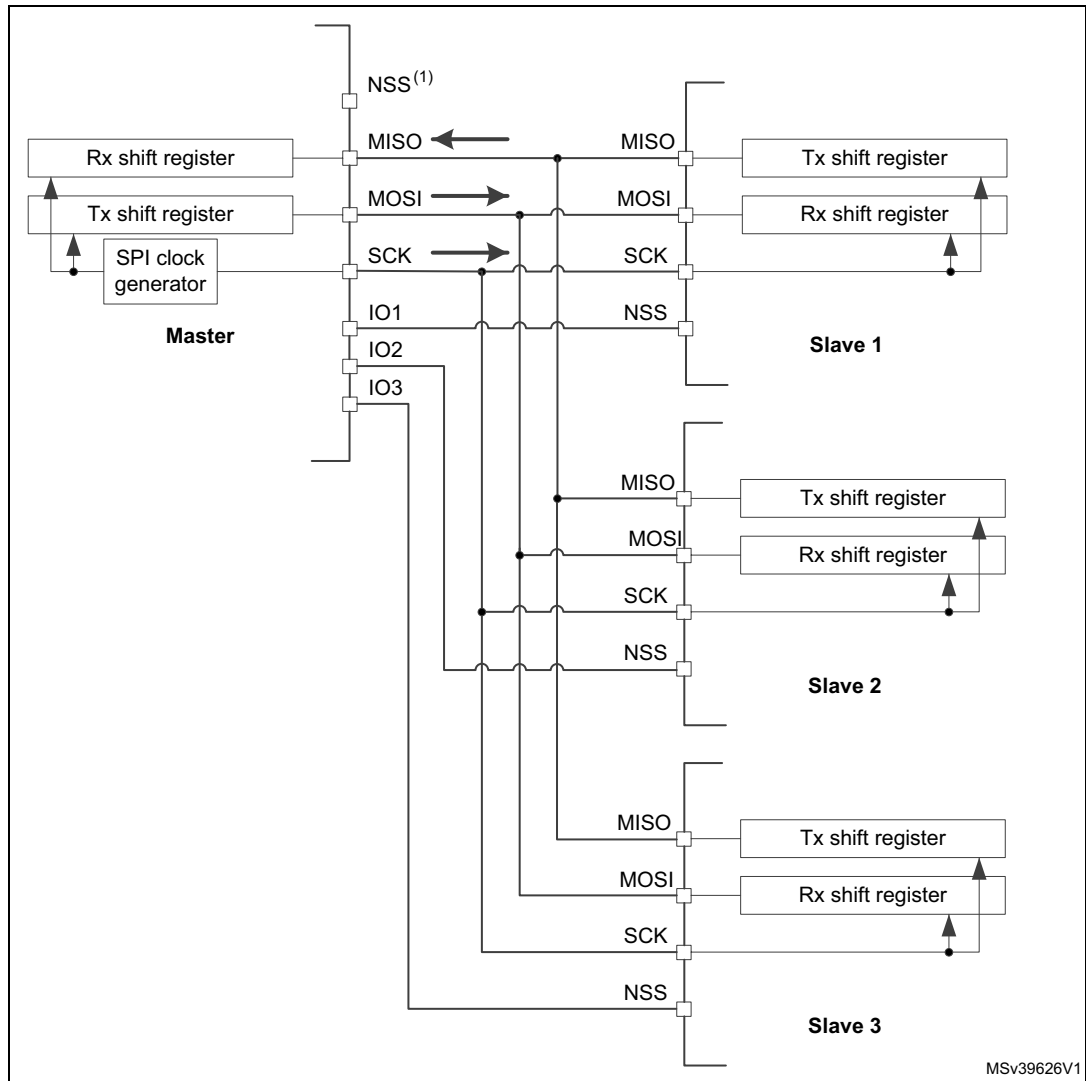
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 26.3.5: Slave select \(NSS\) pin management](#).
2. An accidental input information is captured at the input of transmitter Rx shift register. All the events associated with the transmitter receive flow must be ignored in standard transmit only mode (e.g. OVF flag).
3. In this configuration, both the MISO pins can be used as GPIOs.

Note: *Any simplex communication can be alternatively replaced by a variant of the half-duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).*

26.3.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see [Figure 307](#)). The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input. When this is done, a standard master and dedicated slave communication is established.

Figure 307. Master and three independent slaves



1. NSS pin is not used on master side at this configuration. It has to be managed internally (SSM=1, SSI=1) to prevent any MODF error.
2. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see [Section 7.3.7: I/O alternate function input/output on page 181](#)).

26.3.4 Multi-master communication

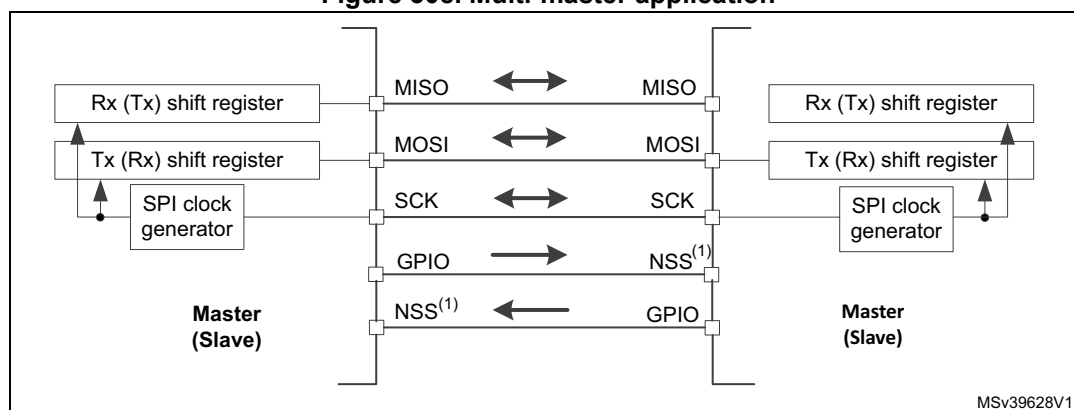
Unless SPI bus is not designed for a multi-master capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

Figure 308. Multi-master application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.

26.3.5 Slave select (NSS) pin management

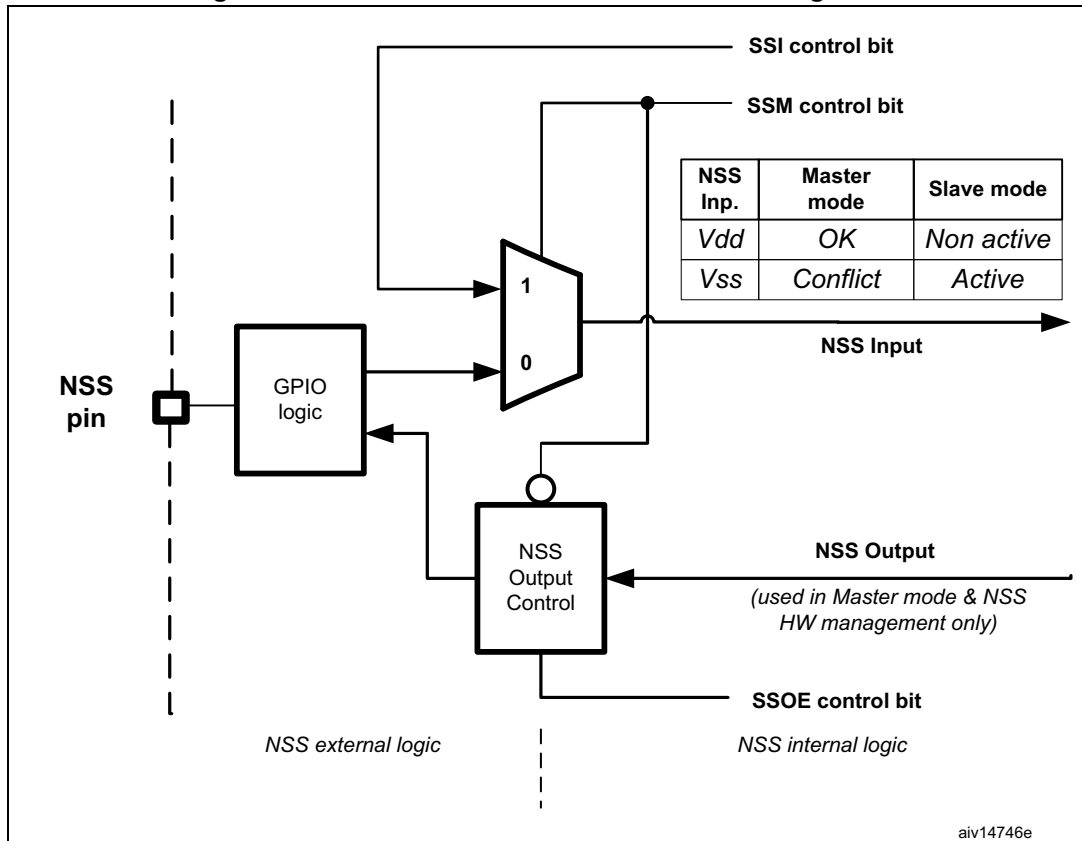
In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

- **Software NSS management (SSM = 1):** in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0):** in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx_CR1).

- **NSS output enable (SSM=0,SSOE = 1):** this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0).
- **NSS output disable (SSM=0, SSOE = 0):** if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 309. Hardware/software slave select management



26.3.6 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

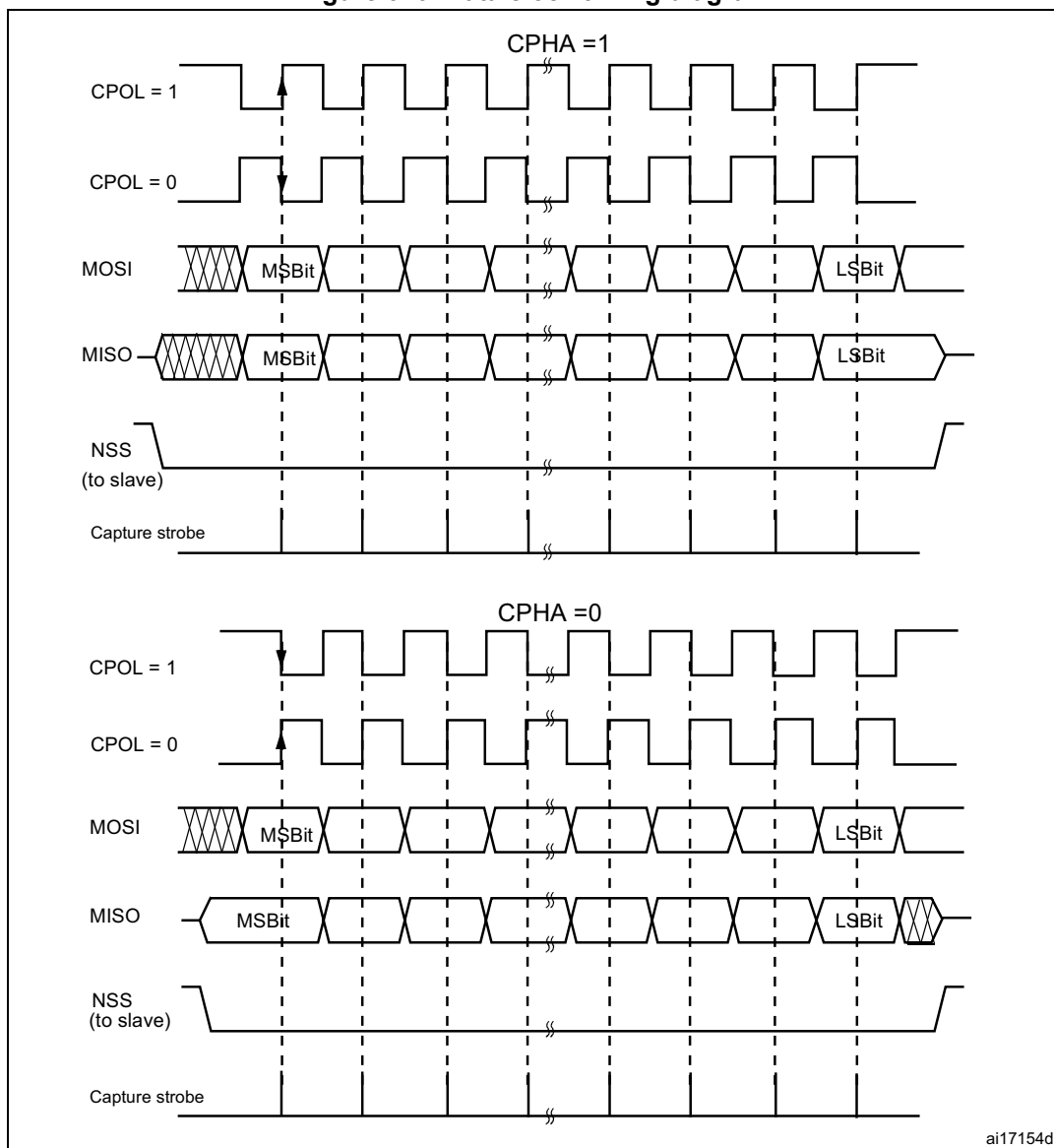
If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

[Figure 310](#), shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit. The idle state of SCK must correspond to the polarity selected in the SPIx_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

Figure 310. Data clock timing diagram



Note: The order of data bits depends on LSBFIRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit. Each data frame is 8 or 16 bit long depending on the size of the data programmed using the DFF bit in the SPI_CR1 register. The selected data frame format is applicable both for transmission and reception.

26.3.7 SPI configuration

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated chapters. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI_CR1 register:
 - a) Configure the serial clock baud rate using the BR[2:0] bits (*Note*: 3).
 - b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock. (*Note*: 2 - except the case when CRC is enabled at TI mode).
 - c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE can't be set at the same time).
 - d) Configure the LSBFIRST bit to define the frame format (*Note*: 2).
 - e) Configure the CRCEN and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
 - f) Configure SSM and SSI (*Note*: 2).
 - g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
 - h) Set the DFF bit to configure the data frame format (8 or 16 bits).
3. Write to SPI_CR2 register:
 - a) Configure SSOE (*Note*: 1 & 2).
 - b) Set the FRF bit if the TI protocol is required.
4. Write to SPI_CR3 register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

Note:

- (1) Step is not required in slave mode.
- (2) Step is not required in TI mode.
- (3) The step is not required in slave mode except slave working at TI mode.

26.3.8 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. Otherwise, undesired data transmission might occur. The slave data register must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

At full-duplex (or in any transmit-only mode), the master starts communicating when the SPI is enabled and data to be sent is written in the Tx Buffer.

In any master receive-only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), the master starts communicating and the clock starts running immediately after the SPI is enabled.

The slave starts communicating when it receives a correct clock signal from the master. The slave software must write the data to be sent before the SPI master initiates the transfer.

Refer to [Section 26.3.11: Communication using DMA \(direct memory addressing\)](#) for details on how to handle DMA.

26.3.9 Data transmission and reception procedures

Rx and Tx buffers

In reception, data are received and then stored into an internal Rx buffer while in transmission, data are first stored into an internal Tx buffer before being transmitted. A read access to the SPI_DR register returns the Rx buffered value whereas a write access to the SPI_DR stores the written data into the Tx buffer.

Tx buffer handling

The data frame is loaded from the Tx buffer into the shift register during the first bit transmission. Bits are then shifted out serially from the shift register to a dedicated output pin depending on LSBFIRST bit setting. The TXE flag (Tx buffer empty) is set when the data are transferred from the Tx buffer to the shift register. It indicates that the internal Tx buffer is ready to be loaded with the next data. An interrupt can be generated if the TXEIE bit of the SPI_CR2 register is set. Clearing the TXE bit is performed by writing to the SPI_DR register.

A continuous transmit stream can be achieved if the next data to be transmitted are stored in the Tx buffer while previous frame transmission is still ongoing. When the software writes to Tx buffer while the TXE flag is not set, the data waiting for transaction is overwritten.

Rx buffer handling

The RXNE flag (Rx buffer not empty) is set on the last sampling clock edge, when the data are transferred from the shift register to the Rx buffer. It indicates that data are ready to be read from the SPI_DR register. An interrupt can be generated if the RXNEIE bit in the SPI_CR2 register is set. Clearing the RXNE bit is performed by reading the SPI_DR register.

If a device has not cleared the RXNE bit resulting from the previous data byte transmitted, an overrun condition occurs when the next value is buffered. The OVR bit is set and an interrupt is generated if the ERRIE bit is set.

Another way to manage the data exchange is to use DMA (see [Section 9.2: DMA main features](#)).

Sequence handling

The BSY bit is set when a current data frame transaction is ongoing. When the clock signal runs continuously, the BSY flag remains set between data frames on the master side. However, on the slave side, it becomes low for a minimum duration of one SPI clock cycle between each data frame transfer.

For some configurations, the BSY flag can be used during the last data transfer to wait until the completion of the transfer.

When a receive-only mode is configured on the master side, either in half-duplex (BIDIMODE=1, BIDIOE=0) or simplex configuration (BIDIMODE=0, RXONLY=1), the master starts the receive sequence as soon as the SPI is enabled. Then the clock signal is provided by the master and it does not stop until either the SPI or the receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous), it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no

underflow error signal for slave operating in SPI mode, and that data from the slave are always transacted and processed by the master even if the slave cannot not prepare them correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In single slave systems, using NSS to control the slave is not necessary. However, the NSS pulse can be used to synchronize the slave with the beginning of each data transfer sequence. NSS can be managed either by software or by hardware (see [Section 26.3.4: Multi-master communication](#)).

Refer to [Figure 311](#) and [Figure 312](#) for a description of continuous transfers in master / full-duplex and slave full-duplex mode.

Figure 311. TXE/RXNE/BSY behavior in master / full-duplex mode (BIDIMODE=0, RXONLY=0) in the case of continuous transfers

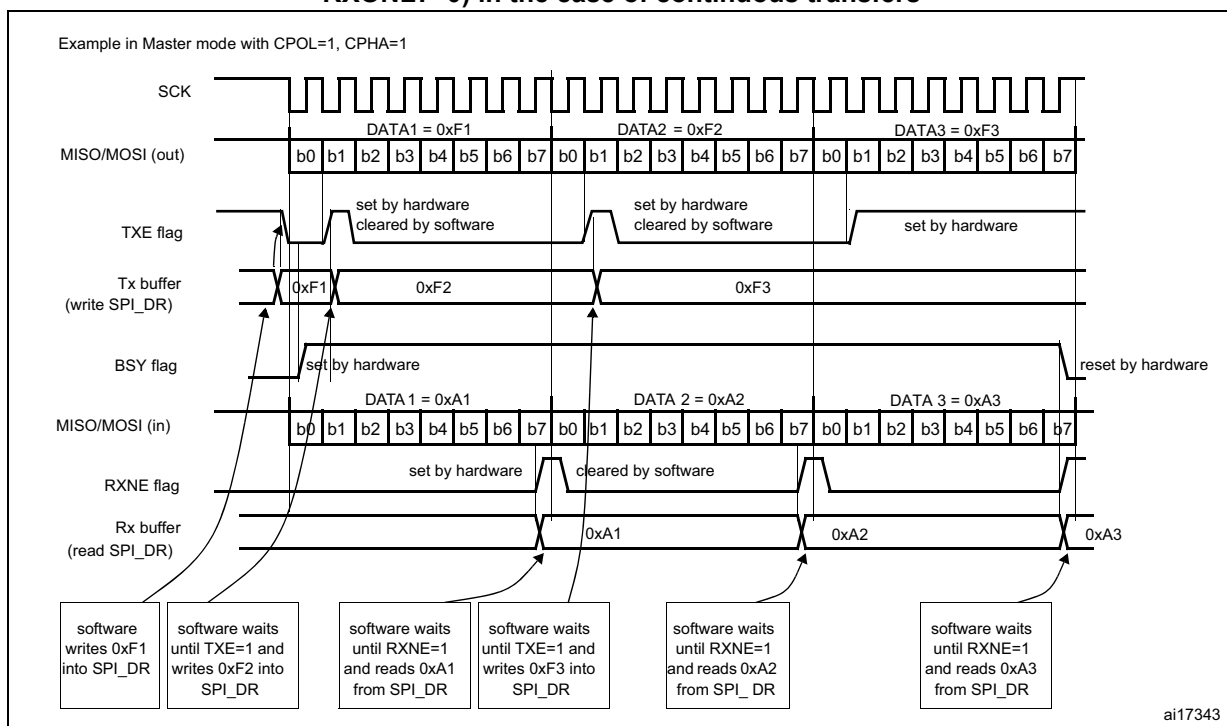
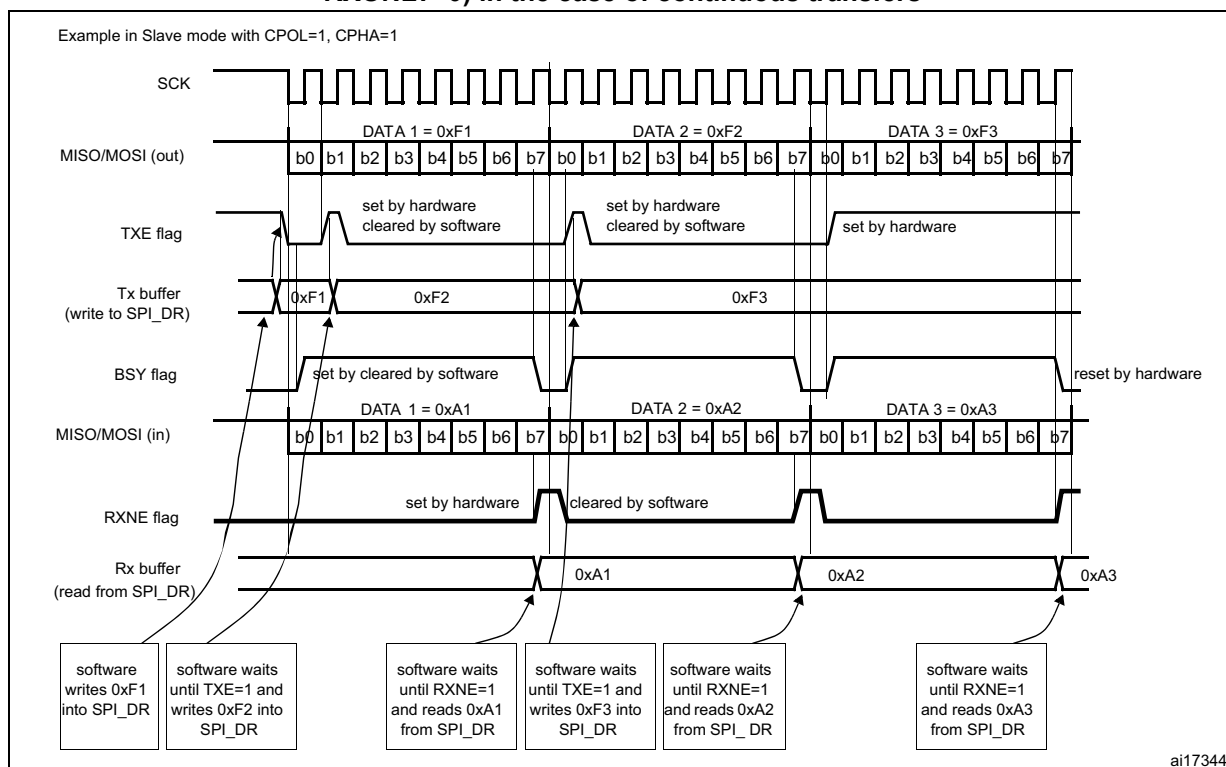


Figure 312. TXE/RXNE/BSY behavior in slave / full-duplex mode (BIDIMODE=0, RXONLY=0) in the case of continuous transfers



26.3.10 Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full-duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction.

Standard disable procedure is based on pulling BSY status together with TXE flag to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by an arbitrary GPIO toggle and the master has to provide proper end of NSS pulse for slave, or
- When transactions' streams from DMA are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive-only mode is used):

1. Wait until RXNE=1 to receive the last data.
2. Wait until TXE=1 and then wait until BSY=0 before disabling the SPI.
3. Read received data.

Note: During discontinuous communications, there is a 2 APB clock period delay between the write operation to the SPI_DR register and BSY bit setting. As a consequence it is mandatory to wait first until TXE is set and then until BSY is cleared after writing the last data.

The correct disable procedure for certain receive-only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read received data.

Note: To stop a continuous receive sequence, a specific time window must be respected during the reception of the last data frame. It starts when the first bit is sampled and ends before the last bit transfer starts.

26.3.11 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXE or RXNE enable bit in the SPIx_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx_DR register.

Refer to [Figure 313](#) and [Figure 314](#) for a description of the DMA transmission and reception waveforms.

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received is not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until TXE = 1 and then until BSY = 0.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CR2 register, if DMA Rx is used.
2. Enable DMA streams for Tx and Rx in DMA registers, if the streams are used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CR2 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA streams for Tx and Rx in the DMA registers, if the streams are used.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CR2 register, if DMA Tx and/or DMA Rx are used.

Figure 313. Transmission using DMA

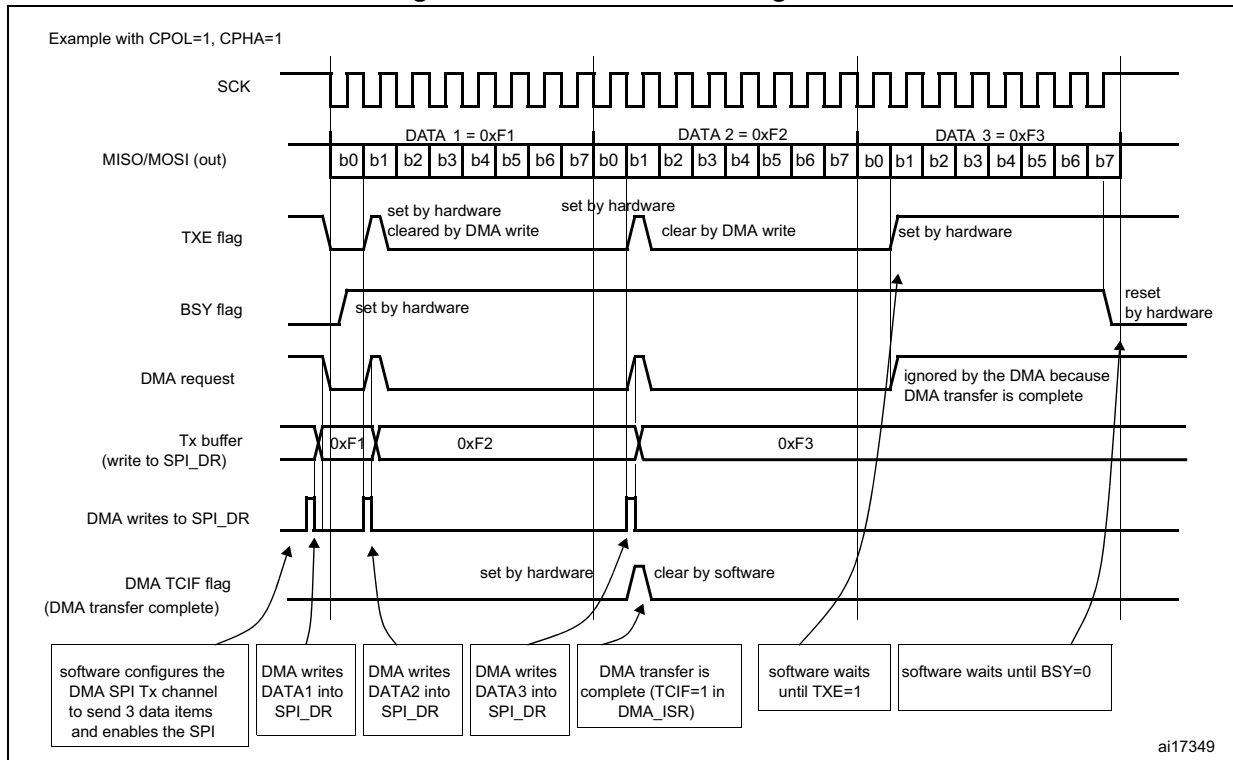
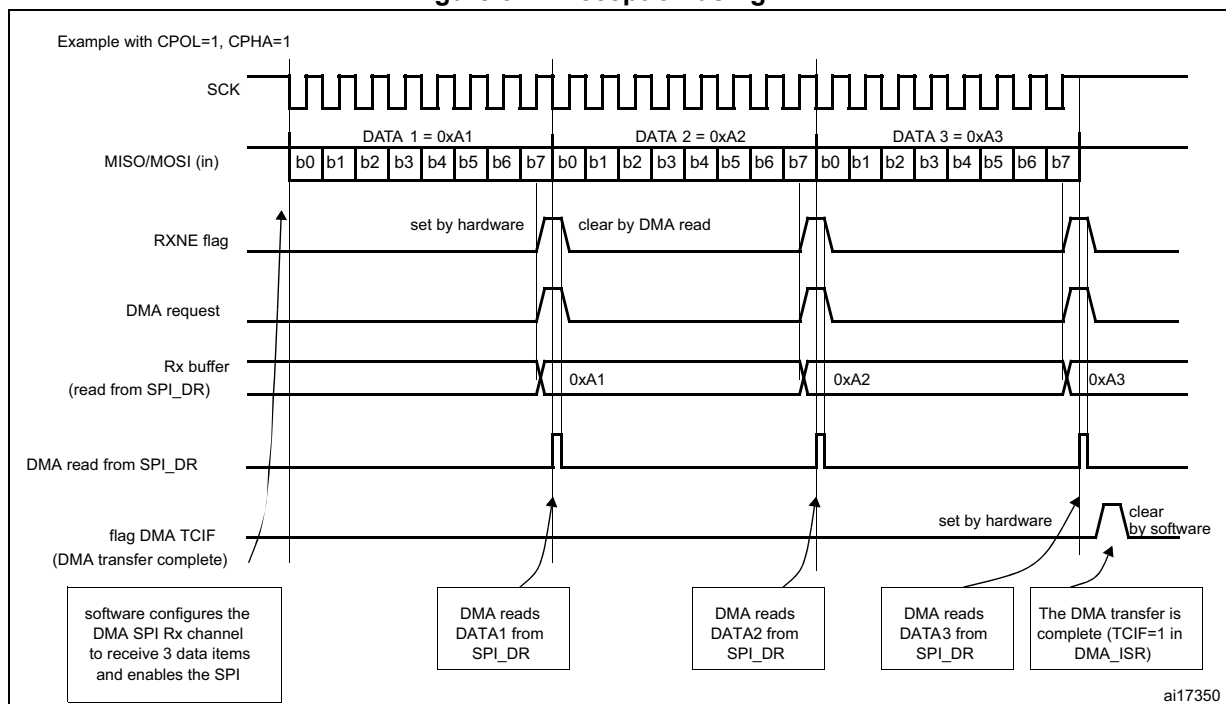


Figure 314. Reception using DMA



26.3.12 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

When it is set, the TXE flag indicates that the Tx buffer is empty and that the next data to be transmitted can be loaded into the buffer. The TXE flag is cleared by writing to the SPI_DR register.

Rx buffer not empty (RXNE)

When set, the RXNE flag indicates that there are valid received data in the Rx buffer. It is cleared by reading from the SPI_DR register.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy). There is one exception in master bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0) where the BSY flag is kept low during reception.

The BSY flag can be used in certain modes to detect the end of a transfer, thus preventing corruption of the last transfer when the SPI peripheral clock is disabled before entering a low-power mode or an NSS pulse end is handled by software.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note: It is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.

26.3.13 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

Overrun flag (OVR)

An overrun condition occurs when the master or the slave completes the reception of the next data frame while the read operation of the previous frame from the Rx buffer has not completed (case RXNE flag is set).

In this case, the content of the Rx buffer is not updated with the new data received. A read operation from the SPI_DR register returns the frame previously received. All other subsequently transmitted data are lost.

Clearing the OVR bit is done by a read access to the SPI_DR register followed by a read access to the SPI_SR register.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx_SR register while the MODF bit is set.
2. Then write to the SPIx_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx_CR1 register is set. The CRCERR flag in the SPIx_SR register is set if the value received in the shift register does not match the receiver SPIx_RXCRC value. The flag is cleared by the software.

TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

The FRE flag is cleared when SPIx_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be re-initiated by the master when the slave SPI is enabled again.

26.4 SPI special features**26.4.1 TI mode****TI protocol in master mode**

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx_CR1 and SPIx_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see [Figure 315](#)). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ (t_{release}) depends on internal resynchronization and on the baud rate value set in through the BR[2:0] bits in the SPIx_CR1 register. It is given by the formula:

$$\frac{t_{\text{baud_rate}}}{2} + 4 \times t_{\text{pclk}} < t_{\text{release}} < \frac{t_{\text{baud_rate}}}{2} + 6 \times t_{\text{pclk}}$$

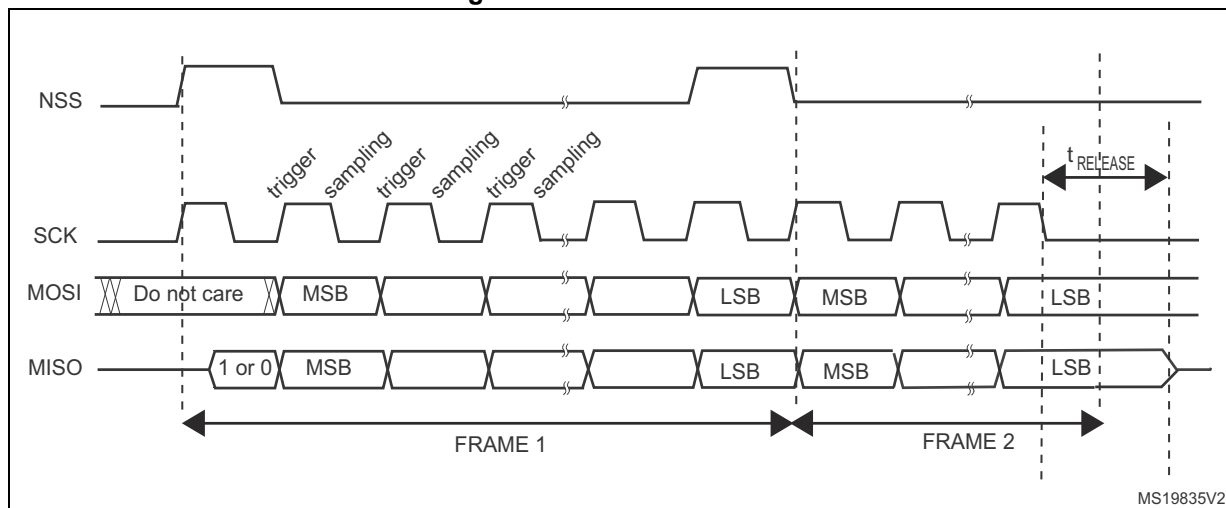
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

Note: To detect TI frame errors in slave transmitter only mode by using the Error interrupt (ERRIE=1), the SPI must be configured in 2-line unidirectional mode by setting BIDIMODE and BIDIOE to 1 in the SPI_CR1 register. When BIDIMODE is set to 0, OVR is set to 1 because the data register is never read and error interrupts are always generated, while when BIDIMODE is set to 1, data are not received and OVR is never set.

Figure 315 shows the SPI communication waveforms when TI mode is selected.

Figure 315. TI mode transfer



26.4.2 CRC calculation

Two separate CRC calculators (on transmission and reception data flows) are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation depending on the data format selected through the DFF bit. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register.

CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

Note: The polynomial value should only be odd. No even values are supported.

CRC transfer managed by CPU

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx_DR register. Then CRCNEXT bit has to be set in the SPIx_CR1 register to indicate that the CRC frame transaction will follow after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the Rx buffer like any other data frame.

A CRC-format transaction takes one more data frame to communicate at the end of data sequence.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx_RXCRC register. Software has to check the CRCERR flag in the SPIx_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the Rx buffer and must be read in the SPIx_DR register in order to clear the RXNE flag.

CRC transfer managed by DMA

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive-only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out the CRC frame received from SPI_DR as it is always loaded into it.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx_SR register is set if corruption occurred during the transfer.

Resetting the SPIx_TXCRC and SPIx_RXCRC values

The SPIx_TXCRC and SPIx_RXCRC values are cleared automatically when CRC calculation is enabled.

When the SPI is configured in slave mode with the CRC feature enabled, a CRC calculation is performed even if a high level is applied on the NSS pin. This may happen for example in case of a multislave environment where the communication master addresses slaves alternately.

Between a slave disabling (high level on NSS) and a new slave enabling (low level on NSS), the CRC value should be cleared on both master and slave sides to resynchronize the master and slave respective CRC calculation.

To clear the CRC, follow the below sequence:

1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

Note: When the SPI is in slave mode, the CRC calculator is sensitive to the SCK slave input clock as soon as the CRCEN bit is set, and this is the case whatever the value of the SPE bit. In order to avoid any wrong CRC calculation, the software must enable the CRC calculation only when the clock is stable (in steady state). When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low between the data phase and the CRC phase once the CRCNEXT signal is released.

At TI mode, despite the fact that the clock phase and clock polarity setting is fixed and independent on the SPIx_CR1 register, the corresponding setting CPOL=0 CPHA=1 has to be kept at the SPIx_CR1 register anyway if CRC is applied. In addition, the CRC calculation

has to be reset between sessions by the SPI disable sequence by re-enabling the CRCEN bit described above at both master and slave sides, else the CRC calculation can be corrupted at this specific mode.

26.5 SPI interrupts

During SPI communication an interrupts can be generated by the following events:

- Transmit Tx buffer ready to be loaded
- Data received in Rx buffer
- Master mode fault
- Overrun error
- TI frame format error

Interrupts can be enabled and disabled separately.

Table 160. SPI interrupt requests

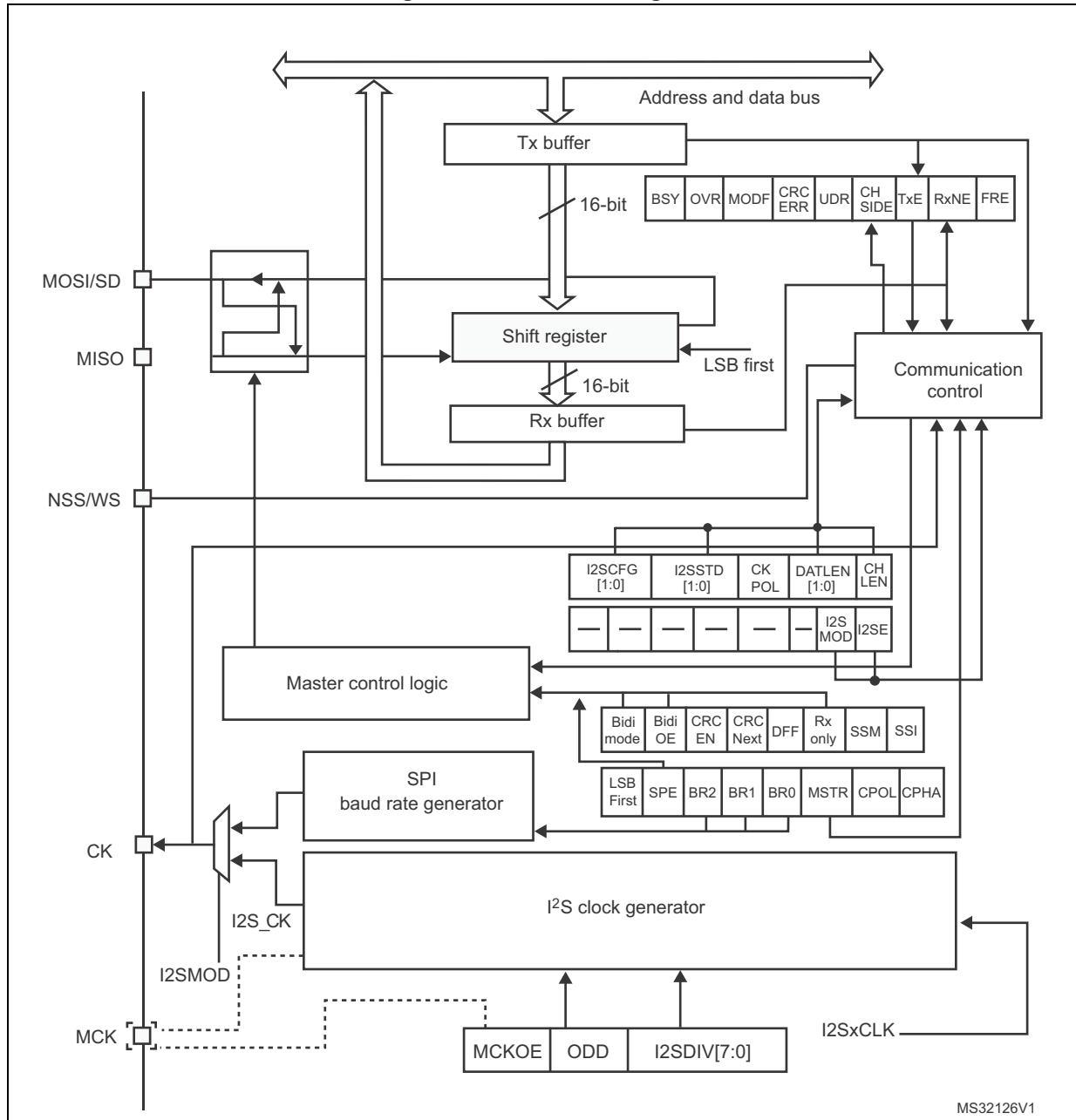
Interrupt event	Event flag	Enable Control bit
Transmit Tx buffer ready to be loaded	TXE	TXEIE
Data received in Rx buffer	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
CRC error	CRCERR	
TI frame format error	FRE	

26.6 I²S functional description

26.6.1 I²S general description

The block diagram of the I²S is shown in [Figure 316](#).

Figure 316. I²S block diagram



1. MCK is mapped on the MISO pin.

The SPI can function as an audio I²S interface when the I²S capability is enabled (by setting the I2SMOD bit in the SPIx_I2SCFGR register). This interface mainly uses the same pins, flags and interrupts as the SPI.

The I²S shares three common pins with the SPI:

- SD: Serial Data (mapped on the MOSI pin) to transmit or receive the two time-multiplexed data channels (in half-duplex mode only).
- WS: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I²S is configured in master mode (and when the MCKOE bit in the SPIx_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to $256 \times f_s$, where f_s is the audio sampling frequency.

The I²S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I²S mode. One is linked to the clock generator configuration SPIx_I2SPR and the other one is a generic I²S configuration register SPIx_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPIx_CR1 register and all CRC registers are not used in the I²S mode. Likewise, the SSOE bit in the SPIx_CR2 register and the MODF and CRCERR bits in the SPIx_SR are not used.

The I²S uses the same SPI register for data transfer (SPIx_DR) in 16-bit wide mode.

26.6.2 I2S full-duplex

Figure 317 shows how to perform full-duplex communications using two SPI2S instances. In this case, the WS and CK IOs of both SPI2S must be connected together.

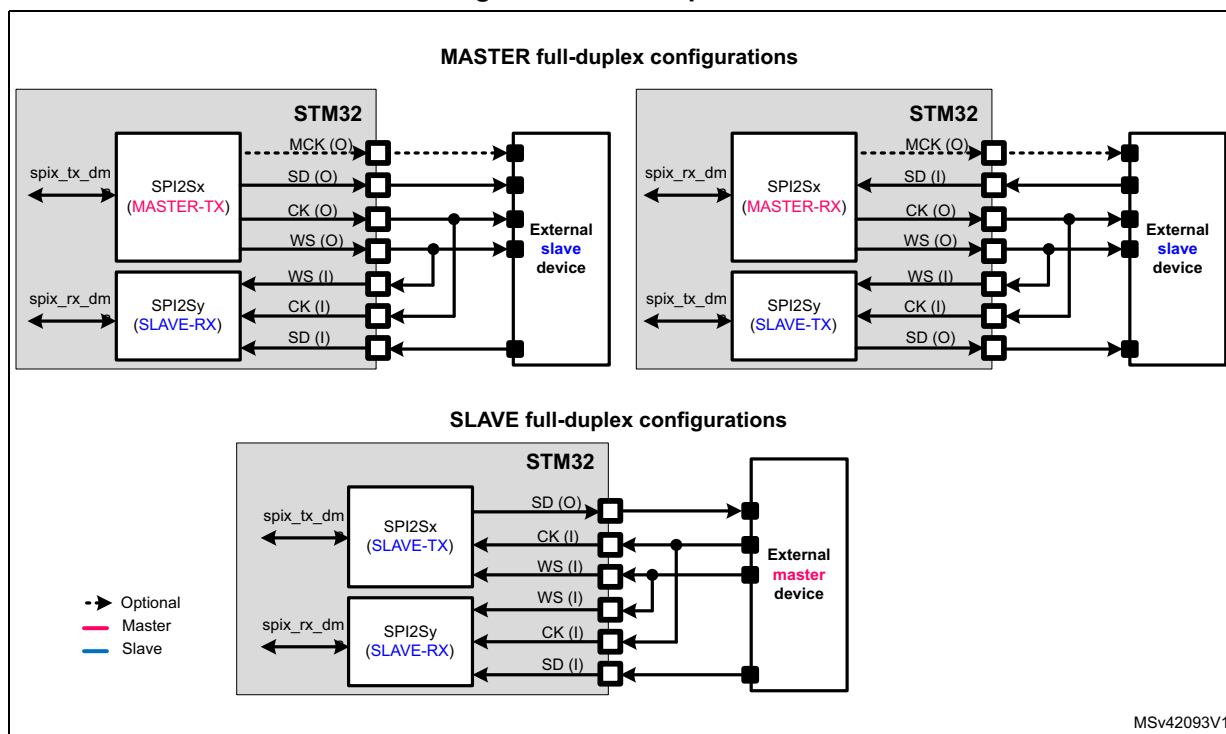
For the master full-duplex mode, one of the SPI2S block must be programmed in master (I2SCFG = '10' or '11'), and the other SPI2S block must be programmed in slave (I2SCFG = '00' or '01'). The MCK can be generated or not, depending on the application needs.

For the slave full-duplex mode, both SPI2S blocks must be programmed in slave. One of them in the slave receiver (I2SCFG = '01'), and the other in the slave transmitter (I2SCFG = '00'). The master external device then provides the bit clock (CK) and the frame synchronization (WS).

Note that the full-duplex mode can be used for all the supported standards: I2S Philips, MSB justified, LSB justified and PCM.

For the full-duplex mode, both SPI2S instances must use the same standard, with the same parameters: I2SMOD, I2SSTD, CKPOL, PCMSYNC, DATLEN and CHLEN must contain the same value on both instances.

Figure 317. Full-duplex communication



26.6.3 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for transmission or reception. So, it is up to the software to write into the data register the appropriate value corresponding to each channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPIx_SR register. Channel left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol).

Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in a 16-bit frame
- 16-bit data packed in a 32-bit frame
- 24-bit data packed in a 32-bit frame
- 32-bit data packed in a 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPIx_DR register or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 non significant bits are extended to 32 bits with 0-bits (by hardware).

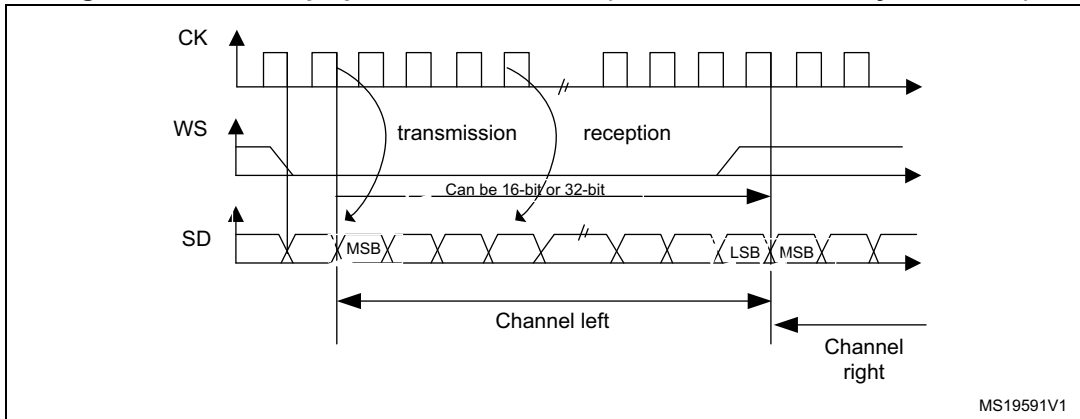
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I²S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx_I2SCFGR register.

I²S Philips standard

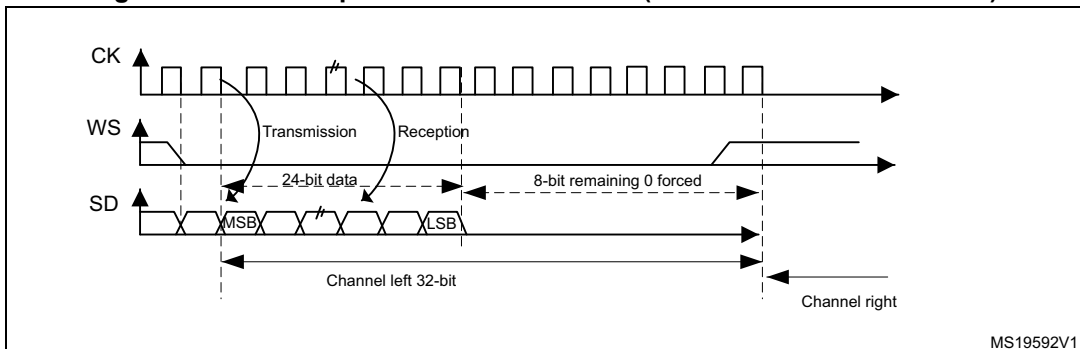
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

Figure 318. I²S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0)



Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

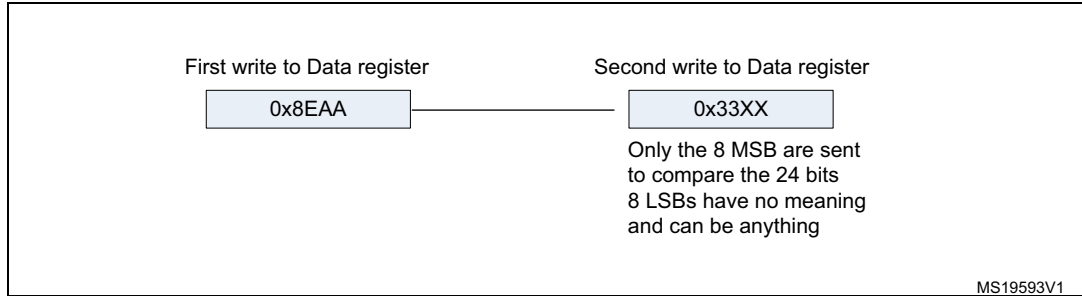
Figure 319. I²S Philips standard waveforms (24-bit frame with CPOL = 0)



This mode needs two write or read operations to/from the SPIx_DR register.

- In transmission mode:
If 0x8EAA33 has to be sent (24-bit):

Figure 320. Transmitting 0x8EAA33



- In reception mode:
If data 0x8EAA33 is received:

Figure 321. Receiving 0x8EAA33

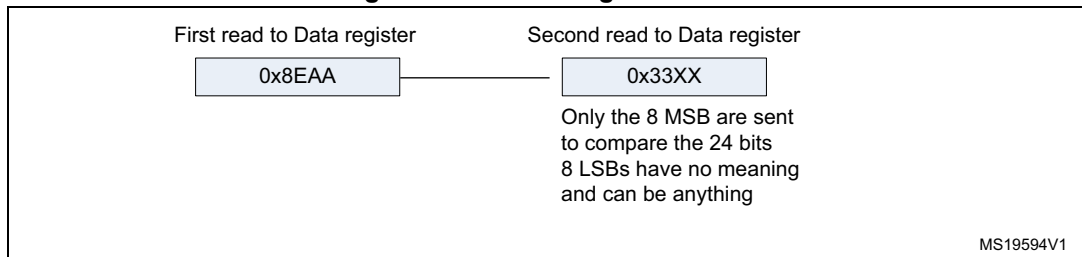
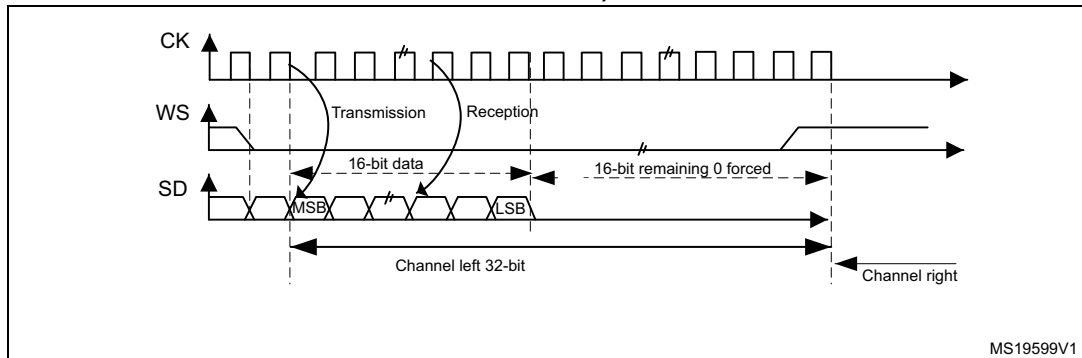


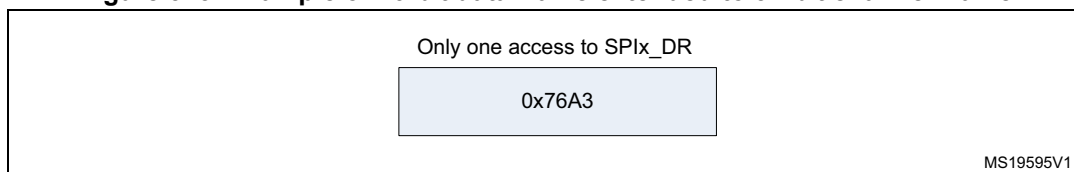
Figure 322. I²S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0)



When 16-bit data frame extended to 32-bit channel frame is selected during the I²S configuration phase, only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in [Figure 323](#) is required.

Figure 323. Example of 16-bit data frame extended to 32-bit channel frame



For transmission, each time an MSB is written to SPIx_DR, the TXE flag is set and its interrupt, if allowed, is generated to load the SPIx_DR register with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

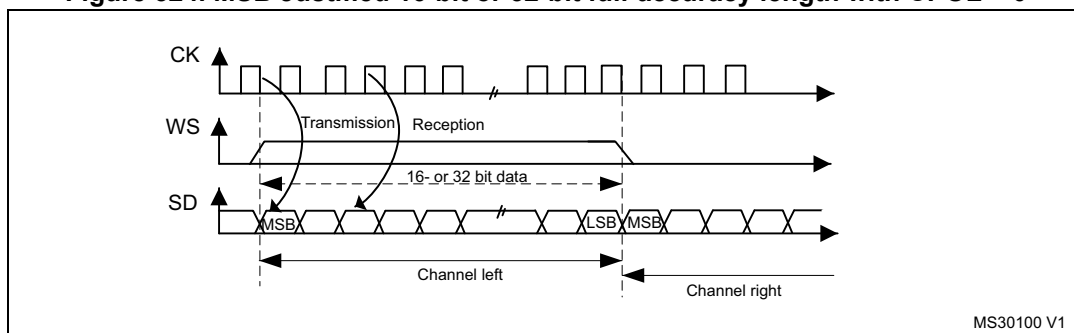
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

MSB justified standard

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

Figure 324. MSB Justified 16-bit or 32-bit full-accuracy length with CPOL = 0



Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

Figure 325. MSB justified 24-bit frame length with CPOL = 0

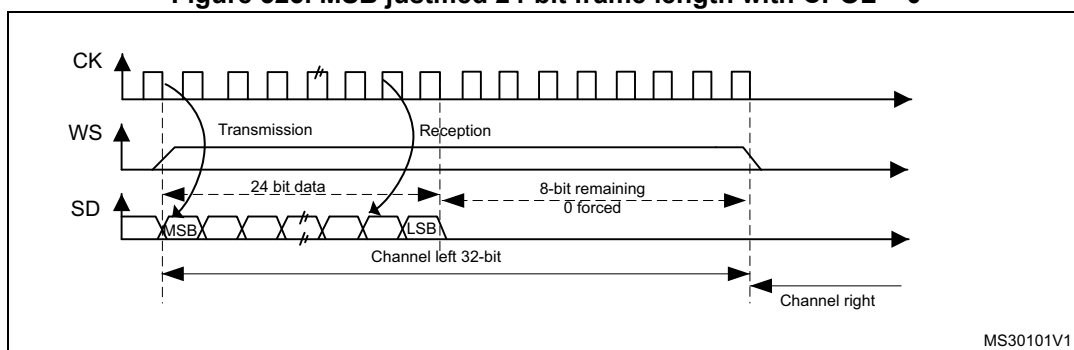
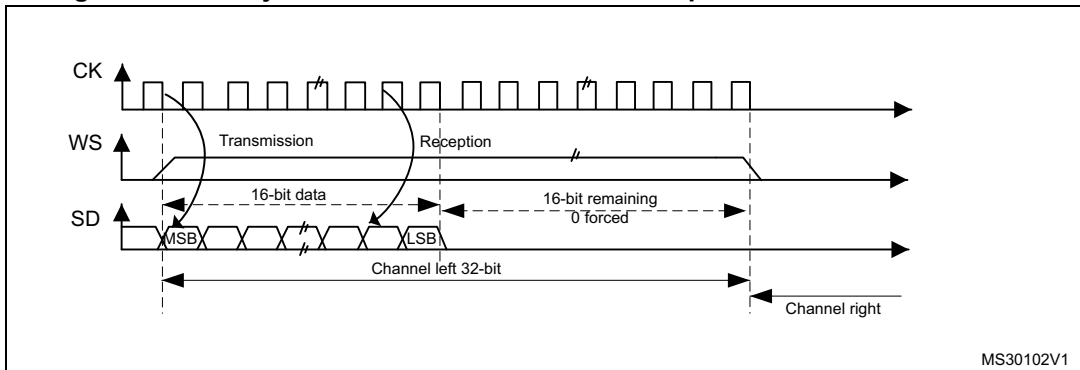


Figure 326. MSB justified 16-bit extended to 32-bit packet frame with CPOL = 0



LSB justified standard

This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

Figure 327. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0

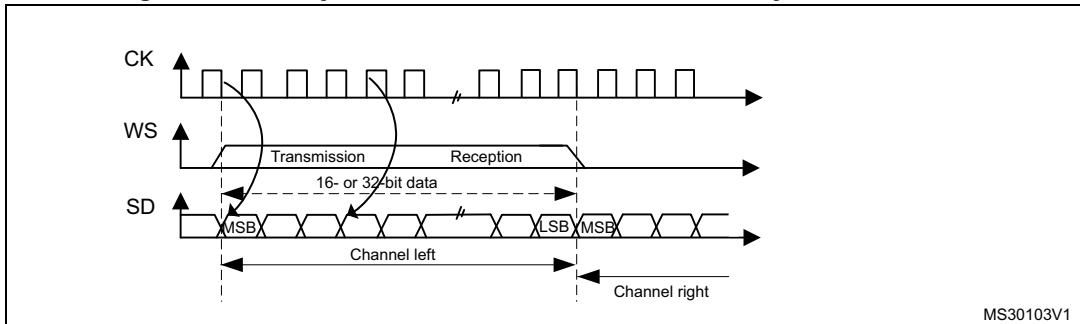
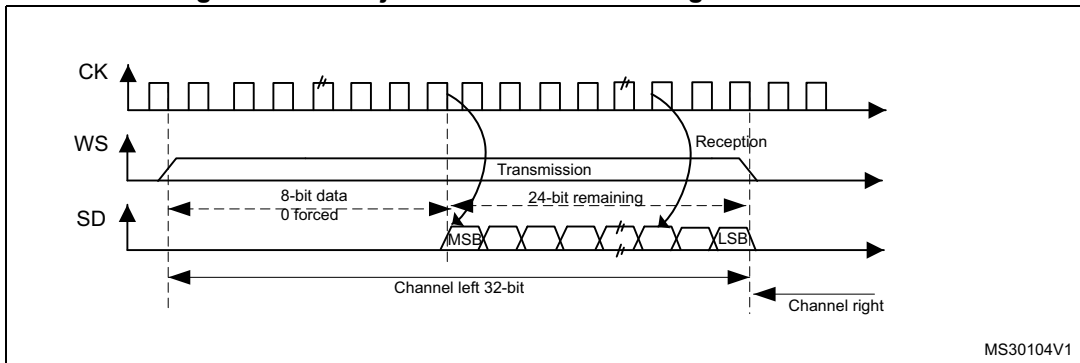
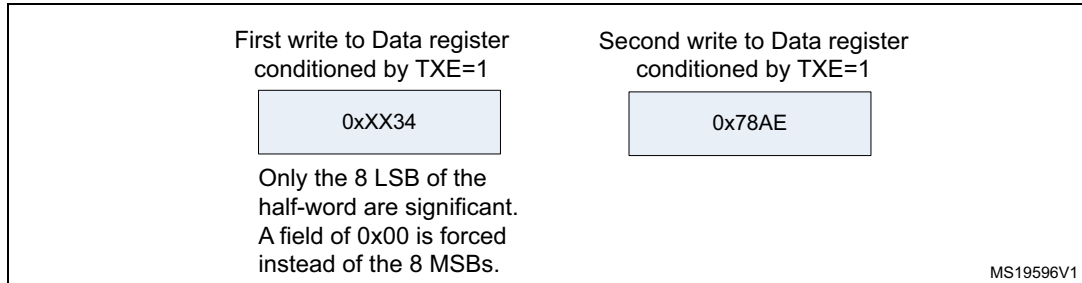


Figure 328. LSB justified 24-bit frame length with CPOL = 0



- In transmission mode:
If data 0x3478AE have to be transmitted, two write operations to the SPIx_DR register are required by software or by DMA. The operations are shown below.

Figure 329. Operations required to transmit 0x3478AE



- In reception mode:
If data 0x3478AE are received, two successive read operations from the SPIx_DR register are required on each RXNE event.

Figure 330. Operations required to receive 0x3478AE

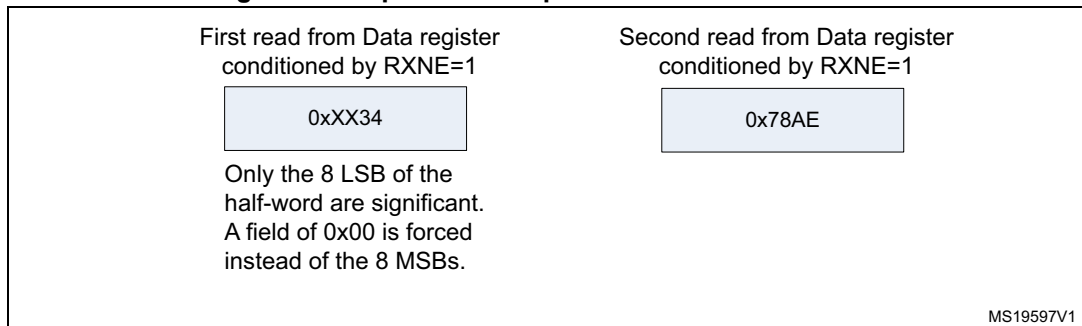
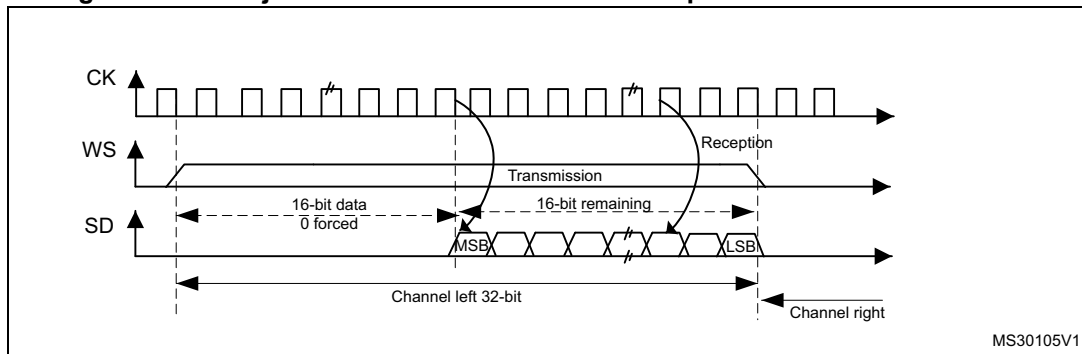


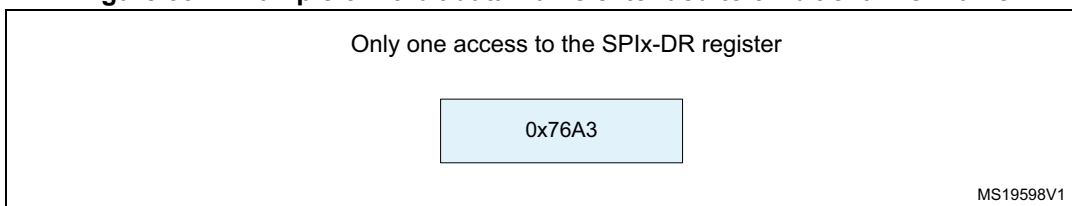
Figure 331. LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0



When 16-bit data frame extended to 32-bit channel frame is selected during the I²S configuration phase, Only one access to the SPIx_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in [Figure 332](#) is required.

Figure 332. Example of 16-bit data frame extended to 32-bit channel frame



In transmission mode, when a TXE event occurs, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). The TXE flag is set again as soon as the effective data (0x76A3) is sent on SD.

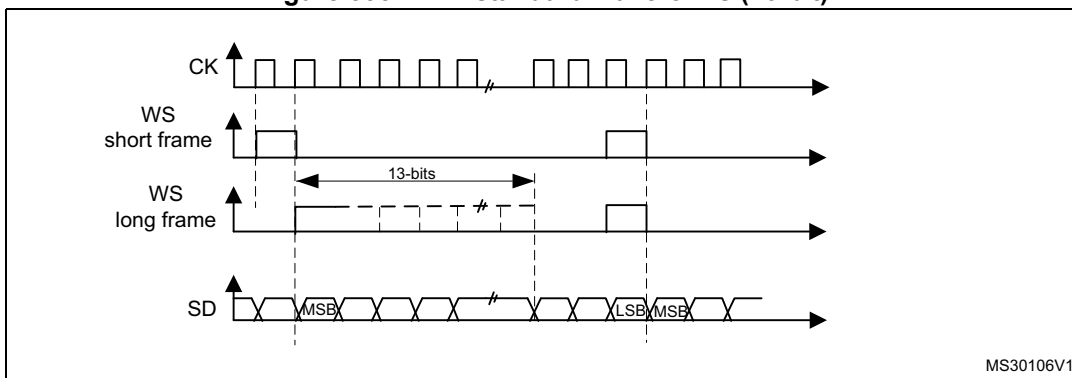
In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPIx_I2SCFGR register.

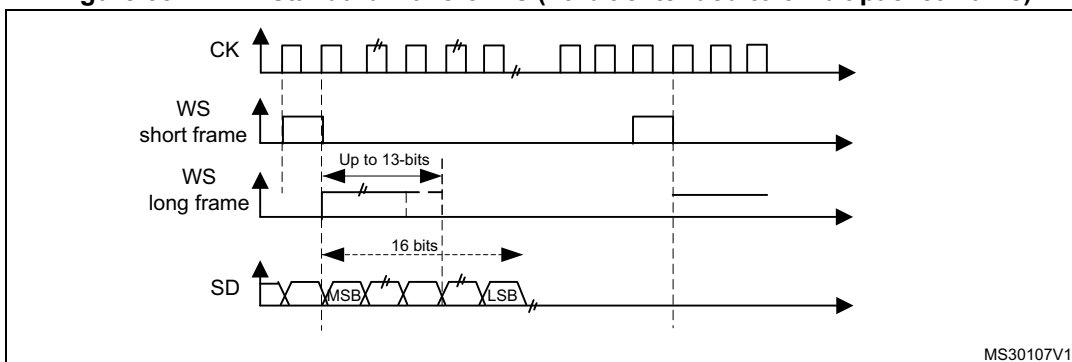
Figure 333. PCM standard waveforms (16-bit)



For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

For short frame synchronization, the WS synchronization signal is only one cycle long.

Figure 334. PCM standard waveforms (16-bit extended to 32-bit packet frame)



Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPIx_I2SCFGR register) even in slave mode.

26.6.4 Clock generator

The I²S bitrate determines the data flow on the I²S data line and the I²S clock signal frequency.

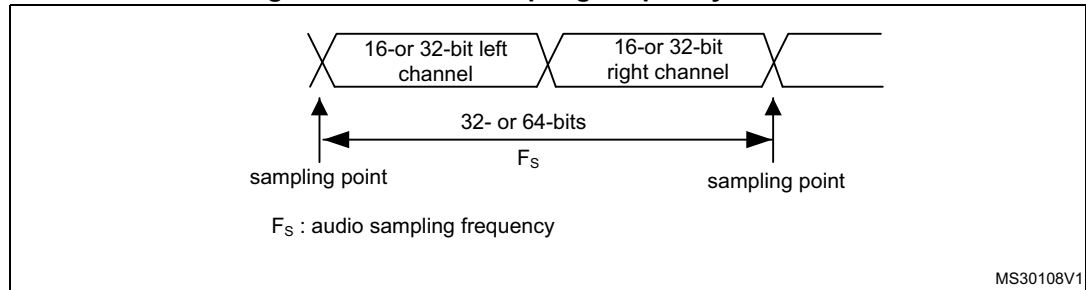
I²S bitrate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I²S bitrate is calculated as follows:

$$I^2S \text{ bitrate} = 16 \times 2 \times f_s$$

It will be: I²S bitrate = 32 × 2 × f_s if the packet length is 32-bit wide.

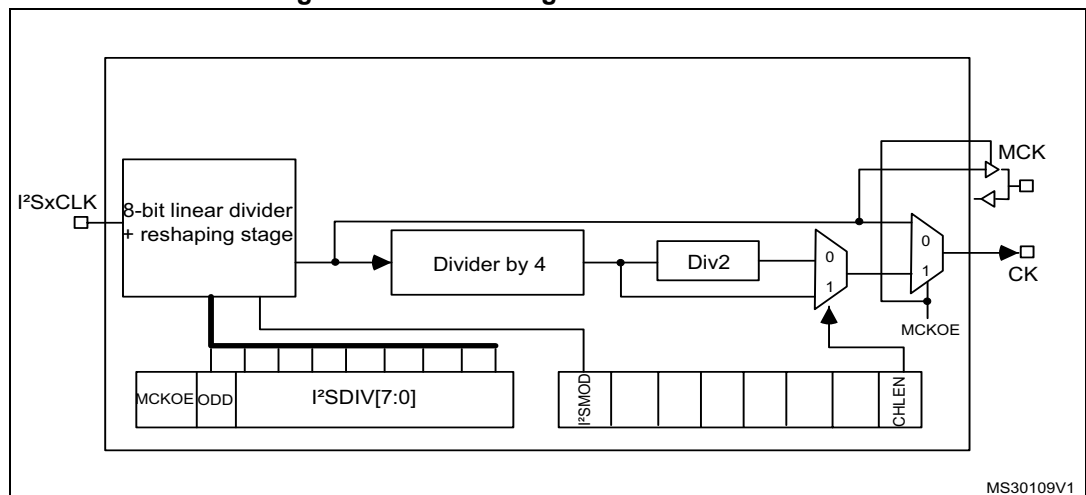
Figure 335. Audio sampling frequency definition



When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency.

Figure 336 presents the communication clock architecture. The I2Sx clock is always the system clock.

Figure 336. I²S clock generator architecture



1. Where x = 2.

The audio sampling frequency may be 192 KHz, 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range). In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

When the master clock is generated (MCKOE in the SPIx_I2SPR register is set):

$$f_S = I2SxCLK / [(16 \cdot 2)^{\cdot} ((2 \cdot I2SDIV) + ODD) \cdot 8]$$
 when the channel frame is 16-bit wide

$$f_S = I2SxCLK / [(32 \cdot 2)^{\cdot} ((2 \cdot I2SDIV) + ODD) \cdot 4]$$
 when the channel frame is 32-bit wide

When the master clock is disabled (MCKOE bit cleared):

$$f_S = I2SxCLK / [(16 \cdot 2)^{\cdot} ((2 \cdot I2SDIV) + ODD)]$$
 when the channel frame is 16-bit wide

$$f_S = I2SxCLK / [(32 \cdot 2)^{\cdot} ((2 \cdot I2SDIV) + ODD)]$$
 when the channel frame is 32-bit wide

Table 161 provides example precision values for different clock configurations.

Note: Other configurations are possible that allow optimum clock precision.

Table 161. Audio-frequency precision using standard 8 MHz HSE⁽¹⁾

SYSCCLK (MHz)	Data length	I2SDIV	I2SODD	MCLK	Target f _S (Hz)	Real f _S (KHz)	Error
48	16	8	0	No	96000	93750	2.3438%
48	32	4	0	No	96000	93750	2.3438%
48	16	15	1	No	48000	48387.0968	0.8065%
48	32	8	0	No	48000	46875	2.3438%
48	16	17	0	No	44100	44117.647	0.0400%
48	32	8	1	No	44100	44117.647	0.0400%
48	16	23	1	No	32000	31914.8936	0.2660%
48	32	11	1	No	32000	32608.696	1.9022%
48	16	34	0	No	22050	22058.8235	0.0400%
48	32	17	0	No	22050	22058.8235	0.0400%
48	16	47	0	No	16000	15957.4468	0.2660%
48	32	23	1	No	16000	15957.447	0.2660%
48	16	68	0	No	11025	11029.4118	0.0400%
48	32	34	0	No	11025	11029.412	0.0400%
48	16	94	0	No	8000	7978.7234	0.2660%
48	32	47	0	No	8000	7978.7234	0.2660%
48	16	2	0	Yes	48000	46875	2.3430%
48	32	2	0	Yes	48000	46875	2.3430%
48	16	2	0	Yes	44100	46875	6.2925%
48	32	2	0	Yes	44100	46875	6.2925%
48	16	3	0	Yes	32000	31250	2.3438%
48	32	3	0	Yes	32000	31250	2.3438%
48	16	4	1	Yes	22050	20833.333	5.5178%

Table 161. Audio-frequency precision using standard 8 MHz HSE⁽¹⁾ (continued)

SYSCLK (MHz)	Data length	I2SDIV	I2SODD	MCLK	Target f _s (Hz)	Real f _s (KHz)	Error
48	32	4	1	Yes	22050	20833.333	5.5178%
48	16	6	0	Yes	16000	15625	2.3438%
48	32	6	0	Yes	16000	15625	2.3438%
48	16	8	1	Yes	11025	11029.4118	0.0400%
48	32	8	1	Yes	11025	11029.4118	0.0400%
48	16	11	1	Yes	8000	8152.17391	1.9022%
48	32	11	1	Yes	8000	8152.17391	1.9022%

1. This table gives only example values for different clock configurations. Other configurations allowing optimum clock precision are possible.

26.6.5 I²S master mode

The I²S can be configured in master mode. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, controlled by the MCKOE bit in the SPIx_I2SPR register.

Procedure

1. Select the I2SDIV[7:0] bits in the SPIx_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPIx_I2SPR register also has to be defined.
2. Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPIx_I2SPR register if the master clock MCK needs to be provided to the external ADC audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to [Section 26.6.4: Clock generator](#)).
3. Set the I2SMOD bit in the SPIx_I2SCFGR register to activate the I²S functions and choose the I²S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the CHLEN bit. Select also the I²S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.
4. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
5. The I2SE bit in SPIx_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPIx_I2SPR is set.

Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Lets assume the first data written into the Tx buffer corresponds to the left channel data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the right channel have to be written into the Tx buffer. The CHSIDE flag

indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high.

A full frame has to be considered as a left channel data transmission followed by a right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

For more details about the write operations depending on the I²S Standard-mode selected, refer to [Section 26.6.3: Supported audio protocols](#).

To ensure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission.

To switch off the I²S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for transmission mode except for the point 3 (refer to the procedure described in [Section 26.6.5: I2S master mode](#)), where the configuration should set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPIx_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I²S cell.

For more details about the read operations depending on the I²S Standard-mode selected, refer to [Section 26.6.3: Supported audio protocols](#).

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I²S, specific actions are required to ensure that the I²S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB justified mode (I2SSTD = 10)
 - a) Wait for the second to last RXNE = 1 (n – 1)
 - b) Then wait 17 I²S clock cycles (using a software loop)
 - c) Disable the I²S (I2SE = 0)
- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, I²S or PCM modes (I2SSTD = 00, I2SSTD = 01 or I2SSTD = 11, respectively)
 - a) Wait for the last RXNE

- b) Then wait 1 I²S clock cycle (using a software loop)
- c) Disable the I²S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I²S:
 - a) Wait for the second to last RXNE = 1 (n – 1)
 - b) Then wait one I²S clock cycle (using a software loop)
 - c) Disable the I²S (I2SE = 0)

Note: The BSY flag is kept low during transfers.

26.6.6 I²S slave mode

For the slave configuration, the I²S can be configured in transmission or reception mode.

The operating mode is following mainly the same rules as described for the I²S master configuration. In slave mode, there is no clock to be generated by the I²S interface. The clock and WS signals are input from the external master connected to the I²S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPIx_I2SCFGR register to select I²S mode and choose the I²S standard through the I2SSTD[1:0] bits, the data length through the DATLEN[1:0] bits and the number of bits per channel for the frame configuring the CHLEN bit. Select also the mode (transmission or reception) for the slave through the I2SCFG[1:0] bits in SPIx_I2SCFGR register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CR2 register.
3. The I2SE bit in SPIx_I2SCFGR register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I²S data register has to be loaded before the master initiates the communication.

For the I²S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I²S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

Note: The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx_CR2 register is set.

Note that the TXE flag should be checked to be at 1 before attempting to write the Tx buffer. For more details about the write operations depending on the I²S Standard-mode selected, refer to [Section 26.6.3: Supported audio protocols](#).

To secure a continuous audio data transmission, it is mandatory to write the SPIx_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPIx_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPIx_CR2 register, an interrupt is generated when the UDR flag in the SPIx_SR register goes high. In this case, it is mandatory to switch off the I²S and to restart a data transfer starting from the left channel.

To switch off the I²S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in [Section 26.6.6: I2S slave mode](#)), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPIx_I2SCFGR register.

Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPIx_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPIx_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer.

The CHSIDE flag is updated each time data are received to be read from the SPIx_DR register. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPIx_DR register.

For more details about the read operations depending the I²S Standard-mode selected, refer to [Section 26.6.3: Supported audio protocols](#).

If data are received while the preceding received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPIx_CR2 register, an interrupt is generated to indicate the error.

To switch off the I²S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

Note: The external master components should have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

26.6.7 I²S status flags

Three status flags are provided for the application to fully monitor the state of the I²S bus.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). It indicates the state of the communication layer of the I²S.

When BSY is set, it indicates that the I²S is busy communicating. There is one exception in master receive mode (I2SCFG = 11) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software needs to disable the I²S. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is set when a transfer starts, except when the I²S is in master receiver mode.

The BSY flag is cleared:

- When a transfer completes (except in master transmit mode, in which the communication is supposed to be continuous)
- When the I²S is disabled

When communication is continuous:

- In master transmit mode, the BSY flag is kept high during all the transfers
- In slave mode, the BSY flag goes low for one I²S clock cycle between each transfer

Note: Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

Tx buffer empty flag (TXE)

When set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can then be loaded into it. The TXE flag is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I²S is disabled (I2SE bit is reset).

RX buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when SPIx_DR register is read.

Channel Side flag (CHSIDE)

In transmission mode, this flag is refreshed when TXE goes high. It indicates the channel side to which the data to transfer on SD has to belong. In case of an underrun error event in slave transmission mode, this flag is not reliable and I²S needs to be switched off and switched on before resuming the communication.

In reception mode, this flag is refreshed when data are received into SPIx_DR. It indicates from which channel side data have been received. Note that in case of error (like OVR) this flag becomes meaningless and the I²S should be reset by disabling and then enabling it (with configuration if it needs changing).

This flag has no meaning in the PCM standard (for both Short and Long frame modes).

When the OVR or UDR flag in the SPIx_SR is set and the ERRIE bit in SPIx_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPIx_SR status register (once the interrupt source has been cleared).

26.6.8 I²S error flags

There are three error flags for the I²S cell.

Underrun flag (UDR)

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPIx_DR. It is available when the I2SMOD bit in the SPIx_I2SCFGR register is set. An interrupt may be generated if the

ERRIE bit in the SPIx_CR2 register is set.
The UDR bit is cleared by a read operation on the SPIx_SR register.

Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from the SPIx_DR register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx_CR2 register.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPIx_DR register followed by a read access to the SPIx_SR register.

Frame error flag (FRE)

This flag can be set by hardware only if the I²S is configured in Slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the synchronization is lost, the following steps are required to recover from this state and resynchronize the external master device with the I²S slave device:

1. Disable the I²S.
2. Enable it again when the correct level is detected on the WS line (WS line is high in I²S mode or low for MSB- or LSB-justified or PCM modes).

Desynchronization between master and slave devices may be due to noisy environment on the SCK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by software when the status register is read.

26.6.9 I²S interrupts

[Table 162](#) provides the list of I²S interrupts.

Table 162. I²S interrupt requests

Interrupt event	Event flag	Enable control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Overrun error	OVR	ERRIE
Underrun error	UDR	
Frame error flag	FRE	

26.6.10 DMA features

In I²S mode, the DMA works in exactly the same way as it does in SPI mode. There is no difference except that the CRC feature is not available in I²S mode since there is no data transfer protection system.

26.7 SPI and I²S registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI_DR in addition by can be accessed by 8-bit access.

Refer to [Section 1.1](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16 bits) or words (32 bits).

26.7.1 SPI control register 1 (SPI_CR1) (not used in I²S mode)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFE	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE**: Bidirectional data mode enable

This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

Note: *This bit is not used in I²S mode*

Bit 14 **BIDIOE**: Output enable in bidirectional mode

This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

Note: *This bit is not used in I²S mode.*

In master mode, the MOSI pin is used while the MISO pin is used in slave mode.

Bit 13 **CRCEN**: Hardware CRC calculation enable

0: CRC calculation disabled

1: CRC calculation enabled

Note: *This bit should be written only when SPI is disabled (SPE = '0') for correct operation.*

It is not used in I²S mode.

Bit 12 **CRCNEXT**: CRC transfer next

0: Data phase (no CRC phase)

1: Next transfer is CRC (CRC phase)

Note: *When the SPI is configured in full-duplex or transmitter only modes, CRCNEXT must be written as soon as the last data is written to the SPI_DR register.*

When the SPI is configured in receiver only mode, CRCNEXT must be set after the second last data reception.

This bit should be kept cleared when the transfers are managed by DMA.

It is not used in I²S mode.

Bit 11 **DFF**: Data frame format

- 0: 8-bit data frame format is selected for transmission/reception
- 1: 16-bit data frame format is selected for transmission/reception

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.

It is not used in I²S mode.

Bit 10 **RXONLY**: Receive only mode enable

This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receive only mode is active.

This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

- 0: full-duplex (Transmit and receive)
- 1: Output disabled (Receive-only mode)

Note: This bit is not used in I²S mode

Bit 9 **SSM**: Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

- 0: Software slave management disabled
- 1: Software slave management enabled

Note: This bit is not used in I²S mode and SPI TI mode

Bit 8 **SSI**: Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the IO value of the NSS pin is ignored.

Note: This bit is not used in I²S mode and SPI TI mode

Bit 7 **LSBFIRST**: Frame format

- 0: MSB transmitted first
- 1: LSB transmitted first

Note: This bit should not be changed when communication is ongoing.

It is not used in I²S mode and SPI TI mode

Bit 6 **SPE**: SPI enable

- 0: Peripheral disabled
- 1: Peripheral enabled

Note: This bit is not used in I²S mode.

When disabling the SPI, follow the procedure described in [Section 26.3.10: Procedure for disabling the SPI](#).

Bits 5:3 **BR[2:0]**: Baud rate control

- 000: $f_{PCLK}/2$
- 001: $f_{PCLK}/4$
- 010: $f_{PCLK}/8$
- 011: $f_{PCLK}/16$
- 100: $f_{PCLK}/32$
- 101: $f_{PCLK}/64$
- 110: $f_{PCLK}/128$
- 111: $f_{PCLK}/256$

Note: These bits should not be changed when communication is ongoing.

They are not used in I²S mode.

Bit 2 **MSTR**: Master selection

- 0: Slave configuration
- 1: Master configuration

Note: This bit should not be changed when communication is ongoing.

It is not used in I²S mode.

Bit1 **CPOL**: Clock polarity

- 0: CK to 0 when idle
- 1: CK to 1 when idle

Note: This bit should not be changed when communication is ongoing.

It is not used in I²S mode and SPI TI mode except the case when CRC is applied at TI mode.

Bit 0 **CPHA**: Clock phase

- 0: The first clock transition is the first data capture edge
- 1: The second clock transition is the first data capture edge

Note: This bit should not be changed when communication is ongoing.

It is not used in I²S mode and SPI TI mode except the case when CRC is applied at TI mode.

26.7.2 SPI control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEIE	RXNEIE	ERRIE	FRF	Res.	SSOE	TXDMAEN	RXDMAEN
								r/w	r/w	r/w	r/w		r/w	r/w	r/w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TXEIE**: Tx buffer empty interrupt enable

- 0: TXE interrupt masked
- 1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE**: RX buffer not empty interrupt enable

- 0: RXNE interrupt masked
- 1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE**: Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (OVR, CRCERR, MODF, FRE in SPI mode, and UDR, OVR, FRE in I²S mode).

- 0: Error interrupt is masked
- 1: Error interrupt is enabled

Bit 4 **FRF**: Frame format

- 0: SPI Motorola mode
- 1 SPI TI mode

Note: This bit is not used in I²S mode.

Bit 3 Reserved. Forced to 0 by hardware.

Bit 2 **SSOE**: SS output enable

- 0: SS output is disabled in master mode and the cell can work in multimaster configuration
- 1: SS output is enabled in master mode and when the cell is enabled. The cell cannot work in a multimaster environment.

Note: This bit is not used in I²S mode and SPI TI mode.

Bit 1 **TXDMAEN**: Tx buffer DMA enable

When this bit is set, the DMA request is made whenever the TXE flag is set.

- 0: Tx buffer DMA disabled
- 1: Tx buffer DMA enabled

Bit 0 **RXDMAEN**: Rx buffer DMA enable

When this bit is set, the DMA request is made whenever the RXNE flag is set.

- 0: Rx buffer DMA disabled
- 1: Rx buffer DMA enabled

26.7.3 SPI status register (SPI_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
							r	r	r	r	rc_w0	r	r	r	r

Bits 15:9 Reserved. Forced to 0 by hardware.

Bit 8 **FRE**: Frame Error

- 0: No frame error
- 1: Frame error occurred.

This bit is set by hardware and cleared by software when the SPI_SR register is read.

This bit is used in SPI TI mode or in I2S mode whatever the audio protocol selected. It detects a change on NSS or WS line which takes place in slave mode at a non expected time, informing about a desynchronization between the external master device and the slave.

Bit 7 **BSY**: Busy flag

- 0: SPI (or I2S) not busy
 - 1: SPI (or I2S) is busy in communication or Tx buffer is not empty
- This flag is set and cleared by hardware.

Note: BSY flag must be used with caution: refer to [Section 26.3.12: SPI status flags](#) and [Section 26.3.10: Procedure for disabling the SPI](#).

Bit 6 **OVR**: Overrun flag

- 0: No overrun occurred
- 1: Overrun occurred

This flag is set by hardware and reset by a software sequence. Refer to [Section 26.3.13: SPI error flags](#) for the software sequence.

Bit 5 **MODF**: Mode fault

- 0: No mode fault occurred
- 1: Mode fault occurred

This flag is set by hardware and reset by a software sequence. Refer to [Section 26.4 on page 865](#) for the software sequence.

Note: This bit is not used in I²S mode

Bit 4 **CRCERR**: CRC error flag

- 0: CRC value received matches the SPI_RXCRCR value
 - 1: CRC value received does not match the SPI_RXCRCR value
- This flag is set by hardware and cleared by software writing 0.

Note: This bit is not used in I²S mode.

Bit 3 **UDR**: Underrun flag

- 0: No underrun occurred
- 1: Underrun occurred

This flag is set by hardware and reset by a software sequence. Refer to [Section 26.6.8: I2S error flags](#) for the software sequence.

Note: This bit is not used in SPI mode.

Bit 2 **CHSIDE**: Channel side

- 0: Channel Left has to be transmitted or has been received
- 1: Channel Right has to be transmitted or has been received

Note: This bit is not used for SPI mode and is meaningless in PCM mode.

Bit 1 **TXE**: Transmit buffer empty

- 0: Tx buffer not empty
- 1: Tx buffer empty

Bit 0 **RXNE**: Receive buffer not empty

- 0: Rx buffer empty
- 1: Rx buffer not empty

26.7.4 SPI data register (SPI_DR)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:0 **DR[15:0]**: Data register

Data received or to be transmitted.

The data register is split into 2 buffers - one for writing (Transmit Buffer) and another one for reading (Receive buffer). A write to the data register will write into the Tx buffer and a read from the data register will return the value held in the Rx buffer.

Note: These notes apply to SPI mode:

Depending on the data frame format selection bit (DFF in SPI_CR1 register), the data sent or received is either 8-bit or 16-bit. This selection has to be made before enabling the SPI to ensure correct operation.

For an 8-bit data frame, the buffers are 8-bit and only the LSB of the register (SPI_DR[7:0]) is used for transmission/reception. When in reception mode, the MSB of the register (SPI_DR[15:8]) is forced to 0.

For a 16-bit data frame, the buffers are 16-bit and the entire register, SPI_DR[15:0] is used for transmission/reception.

26.7.5 SPI CRC polynomial register (SPI_CRCPR) (not used in I²S mode)

Address offset: 0x10

Reset value: 0x0007

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

Bits 15:0 **CRCPOLY[15:0]**: CRC polynomial register

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0007h) is the reset value of this register. Another polynomial can be configured as required.

Note: These bits are not used for the I²S mode.

26.7.6 SPI RX CRC register (SPI_RXCRCR) (not used in I²S mode)

Address offset: 0x14

Reset value: 0x0000

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

Bits 15:0 **RXCRC[15:0]**: Rx CRC register

When CRC calculation is enabled, the RxCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPI_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY Flag is set could return an incorrect value. These bits are not used for I²S mode.

26.7.7 SPI TX CRC register (SPI_TXCRCR) (not used in I²S mode)

Address offset: 0x18

Reset value: 0x0000

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

Bits 15:0 **TXCRC[15:0]**: Tx CRC register

When CRC calculation is enabled, the TxCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPI_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY flag is set could return an incorrect value. These bits are not used for I²S mode.

26.7.8 SPI_I2S configuration register (SPI_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ASTREN**: Asynchronous start enable.

0: The Asynchronous start is disabled. When the I2S is enabled in slave mode, the I2S slave starts the transfer when the I2S clock is received and an appropriate transition (depending on the protocol selected) is detected on the WS signal.

1: The Asynchronous start is enabled. When the I2S is enabled in slave mode, the I2S slave starts immediately the transfer when the I2S clock is received from the master without checking the expected transition of WS signal.

Note: Note: The appropriate transition is a falling edge on WS signal when I2S Philips Standard is used, or a rising edge for other standards.

Bit 11 **I2SMOD**: I2S mode selection

0: SPI mode is selected

1: I2S mode is selected

Note: This bit should be configured when the SPI or I2S is disabled

Bit 10 **I2SE**: I2S Enable

0: I2S peripheral is disabled

1: I2S peripheral is enabled

Note: This bit is not used in SPI mode.

Bits 9:8 **I2SCFG**: I2S configuration mode

00: Slave - transmit

01: Slave - receive

10: Master - transmit

11: Master - receive

Note: This bit should be configured when the I2S is disabled.

It is not used in SPI mode.

Bit 7 **PCMSYNC**: PCM frame synchronization

0: Short frame synchronization

1: Long frame synchronization

Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used)

It is not used in SPI mode.

Bit 6 Reserved: forced at 0 by hardware

Bits 5:4 **I2SSTD**: I2S standard selection

- 00: I²S Philips standard.
- 01: MSB justified standard (left justified)
- 10: LSB justified standard (right justified)
- 11: PCM standard

For more details on I²S standards, refer to [Section 26.6.3 on page 871](#). *Not used in SPI mode.*

Note: For correct operation, these bits should be configured when the I²S is disabled.

Bit 3 **CKPOL**: Steady state clock polarity

- 0: I²S clock steady state is low level
- 1: I²S clock steady state is high level

Note: For correct operation, this bit should be configured when the I²S is disabled.

This bit is not used in SPI mode

Bits 2:1 **DATLEN**: Data length to be transferred

- 00: 16-bit data length
- 01: 24-bit data length
- 10: 32-bit data length
- 11: Not allowed

Note: For correct operation, these bits should be configured when the I²S is disabled.

This bit is not used in SPI mode.

Bit 0 **CHLEN**: Channel length (number of bits per audio channel)

- 0: 16-bit wide
- 1: 32-bit wide

The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in. *Not used in SPI mode.*

Note: For correct operation, this bit should be configured when the I²S is disabled.

26.7.9 SPI_I²S prescaler register (SPI_I2SPR)

Address offset: 0x20

Reset value: 0000 0010 (0x0002)

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

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **MCKOE**: Master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Note: This bit should be configured when the I²S is disabled. It is used only when the I²S is in master mode.

This bit is not used in SPI mode.

Bit 8 **ODD**: Odd factor for the prescaler

0: real divider value is = I2SDIV *2

1: real divider value is = (I2SDIV * 2)+1

Refer to [Section 26.6.4 on page 878](#). *Not used in SPI mode.*

Note: This bit should be configured when the I²S is disabled. It is used only when the I²S is in master mode.

Bits 7:0 **I2SDIV**: I2S Linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to [Section 26.6.4 on page 878](#). *Not used in SPI mode.*

Note: These bits should be configured when the I²S is disabled. It is used only when the I²S is in master mode.

26.7.10 SPI register map

The table provides shows the SPI register map and reset values.

Table 163. SPI register map and reset values

Offset	Register	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BIDMODE	BIDIOE	CRCEN	CRCNEXT	DFE	RXONLY	SSM	SSI	LSBFIRST	SPE	BR	BR	MSTR	CPOL	CPHA				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x04	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEIE	RXNEIE	ERRIE	FRF	Res.	SSOE	TXDMAEN	RXDMAEN			
	Reset value																										0	0	0	0	0	0	0	0		
0x08	SPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE			
	Reset value																									0	0	0	0	0	0	0	1	0		
0x0C	SPI_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x10	SPI_CRCPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCPOLY[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1			
0x14	SPI_RXCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RxCRC[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x18	SPI_TXCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TxCRC[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	SPI_I2SCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASTREN	I2SMOD	I2SE	I2SCFG	PCMSYNC	Res.	I2SSTD	CKPOL	DATLEN	CHLEN						
	Reset value																				0	0	0	0	0		0	0	0	0	0					
0x20	SPI_I2SPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV									
	Reset value																									0	0	0	0	0	0	0	1	0		

Refer to Section 2.2.2 on page 56 for the register boundary addresses.

