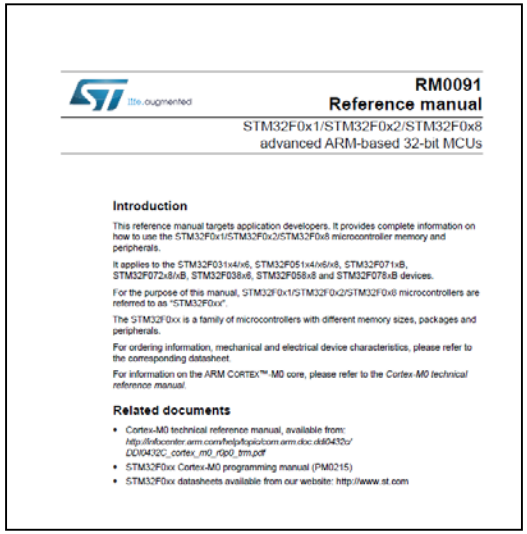
Embedded Communication

Serial Peripheral Interface (SPI)

Chapter 15 SPI
"Mastering STM32" by Carmine Noviello
https://leanpub.com/mastering-stm32

Chapter 27 Serial Peripheral Interface (SPI)
RM0091: STM32F0x1 Reference manual


RM0091
Reference manual
STM32F0x1/STM32F0x2/STM32F0x8
advanced ARM-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32F0x1/STM32F0x2/STM32F0x8 microcontroller memory and peripherals.

It applies to the STM32F031x4/x6, STM32F051x4/x6/x8, STM32F071xB, STM32F072x8/xB, STM32F038x6, STM32F058x8 and STM32F078xB devices.

For the purpose of this manual, STM32F0x1/STM32F0x2/STM32F0x8 microcontrollers are referred to as "STM32F0xx".

The STM32F0xx is a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics, please refer to the corresponding datasheet.

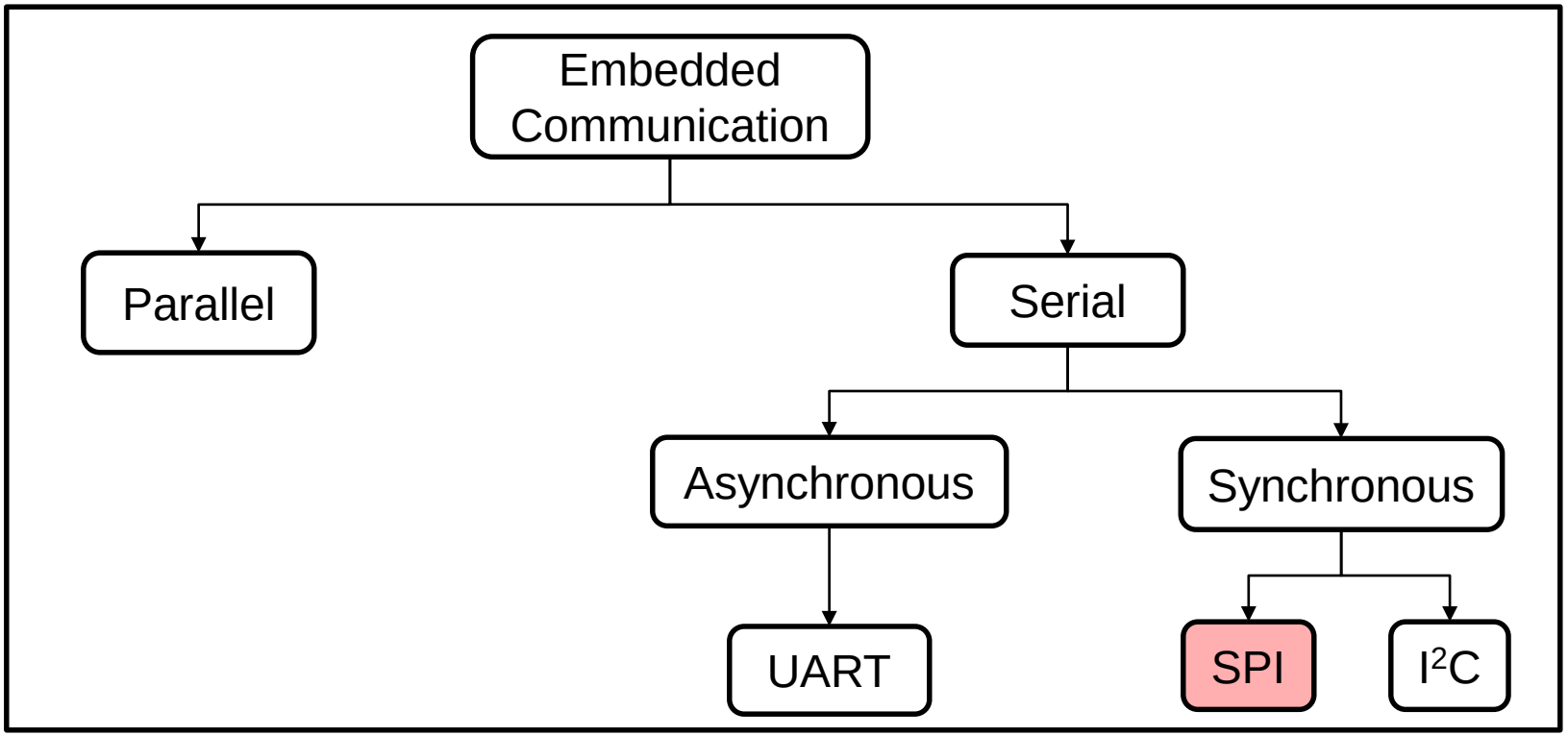
For information on the ARM CORTEX™-M0 core, please refer to the *Cortex-M0 technical reference manual*.

Related documents

- Cortex-M0 technical reference manual, available from: *http://infocenter.arm.com/help/topic/com.arm.doc.ddi0432c/DDI0432C_cortex_m0_r0p0_trm.pdf*
- STM32F0xx Cortex-M0 programming manual (PM0215)
- STM32F0xx datasheets available from our website: http://www.st.com

SPI and the big picture
Context

- SPI is a serial synchronous type of communication
  - Two devices share a common clock signal to synchronise the exchange of data



Overview of embedded communication

Serial Peripheral Interface

Introduction and basic concepts

Serial Peripheral Interface
Introduction and Basic Concepts

- History
  - SPI was developed by Motorola (now called Freescale) in 1979
  - SPI is a defacto standard (ie. standard that has gained acceptance)
  - Synchronous serial data transfer

Serial Peripheral Interface
Introduction and Basic Concepts

- History
- How is synchronous different to asynchronous data transfer?
  - **Asynchronous**: two devices need to agree on how long it takes to transmit one bit of information
  - **Synchronous**: two devices share a common clock that influences timing

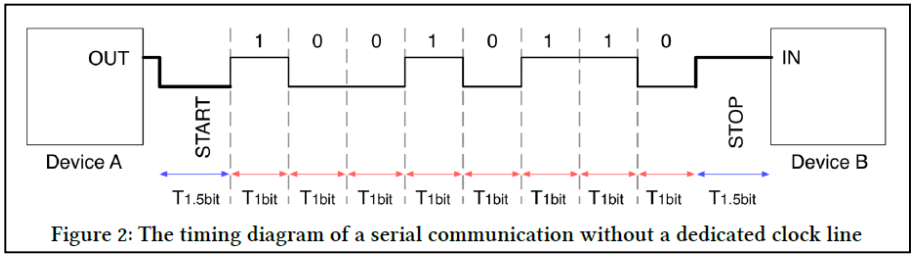


Figure 2: The timing diagram of a serial communication without a dedicated clock line

Asynchronous serial communication

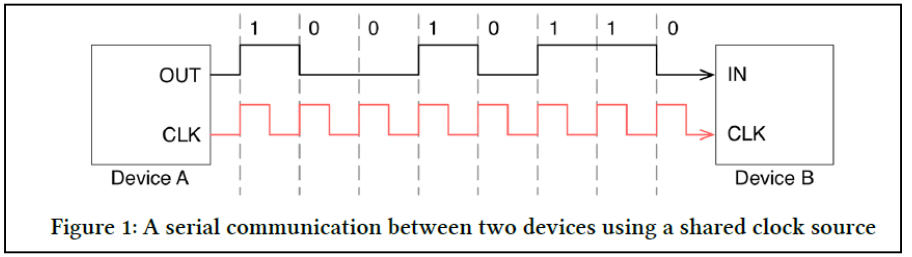


Figure 1: A serial communication between two devices using a shared clock source

Synchronous serial communication

Serial Peripheral Interface
Introduction and Basic Concepts

- History
- How is synchronous different to asynchronous data transfer?
  - **Asynchronous**: two devices need to agree on how long it takes to transmit one bit of information
  - **Synchronous**: two devices share a common clock that influences timing

Serial Peripheral Interface
Introduction and Basic Concepts

- History
- How is synchronous different to asynchronous data transfer?
- Why SPI over UART/RS232?
  - **Faster data rates**
  - A single ‘SPI’ hardware peripheral connects to **multiple SPI peripherals**
  - Hardware is essentially a shift register and **less costly** than UART
  - **No “start” and “stop” bits**. So data can be streamed without these interruptions

Serial Peripheral Interface
Introduction and Basic Concepts

- History
- How is synchronous different to asynchronous data transfer?
- Why SPI over UART/RS232?
- Where is SPI used?
  - SPI is typically used between devices on the same PCB

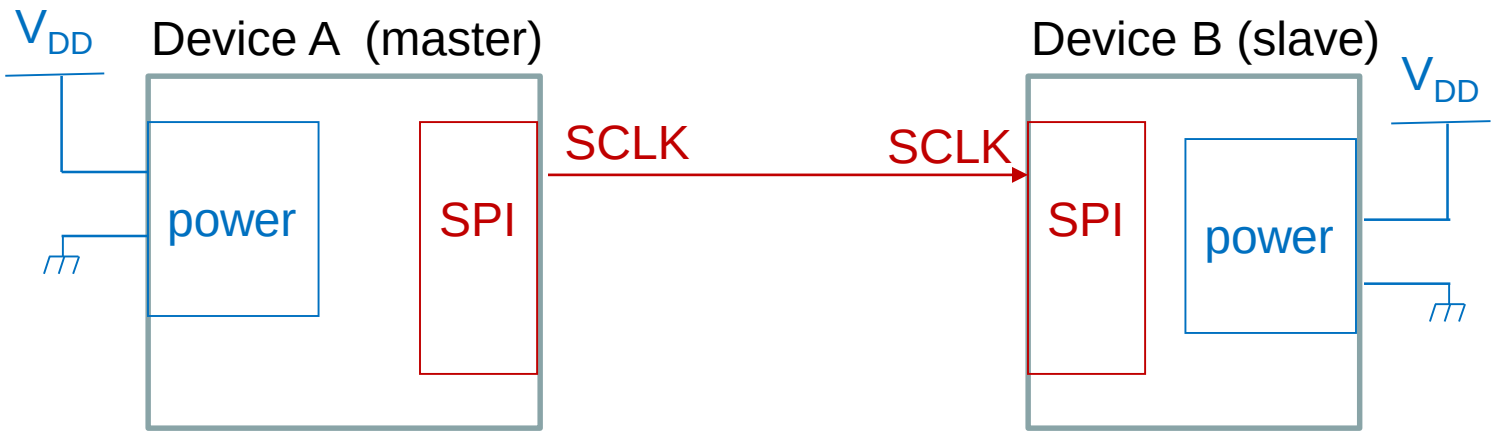
Serial Peripheral Interface
Introduction and Basic Concepts

- History
- How is synchronous different to asynchronous data transfer?
- Why SPI over UART/RS232?
- Where is SPI used?
- Devices that use SPI
  - Analogue to Digital Converters (ADC)
  - Temperature sensors
  - Real time clock
  - SD card

Serial Peripheral Interface
Connections

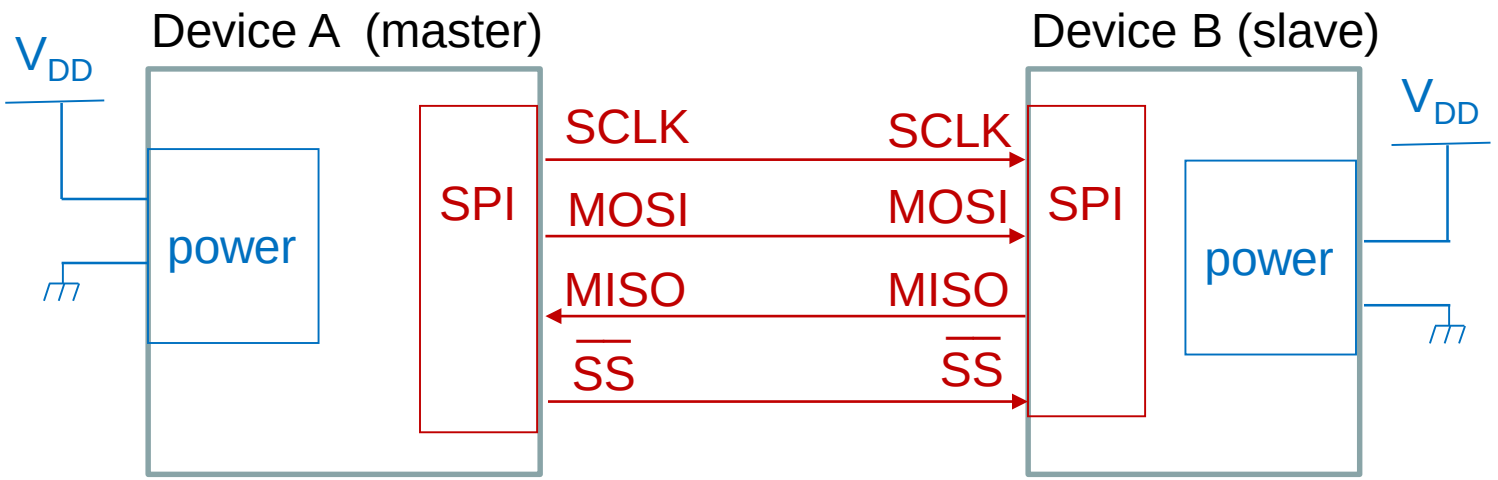
Serial Peripheral Interface
Connection: one to one

- Master and slave
  - The device that generates the clock signal (SCLK) is termed “master”



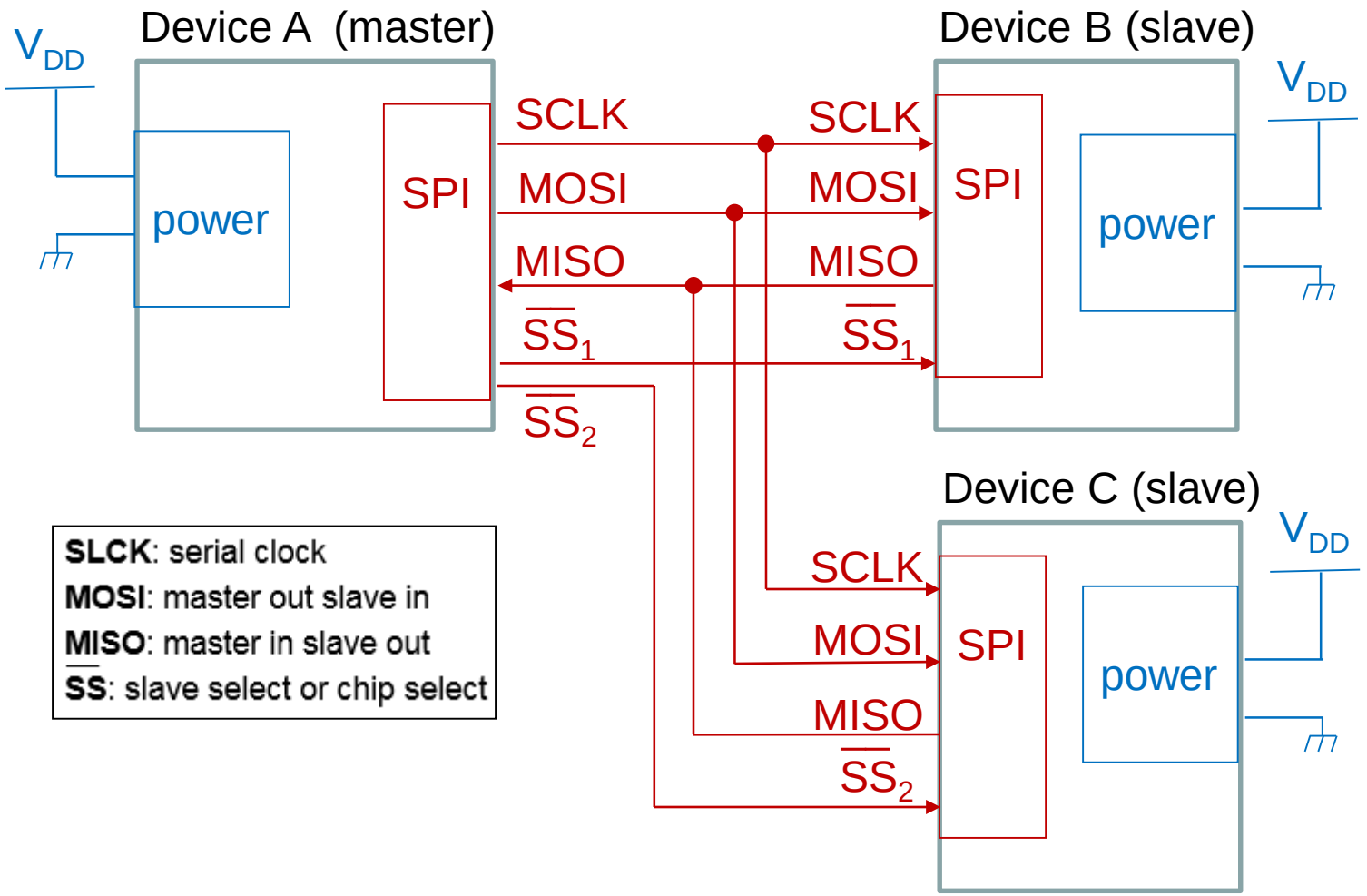
Serial Peripheral Interface
Connection: one to one

- Master and slave
  - The device that generates the clock signal (SCLK) is termed “master”
- Connecting one to one
  - **SLCK**: serial clock
  - **MOSI**: master out slave in
  - **MISO**: master in slave out
  - **$\overline{SS}$**: slave select or chip select. [Note: $\overline{SS}$ = 0 to enable device]



Serial Peripheral Interface
Connection: one to many

- Connecting one to many

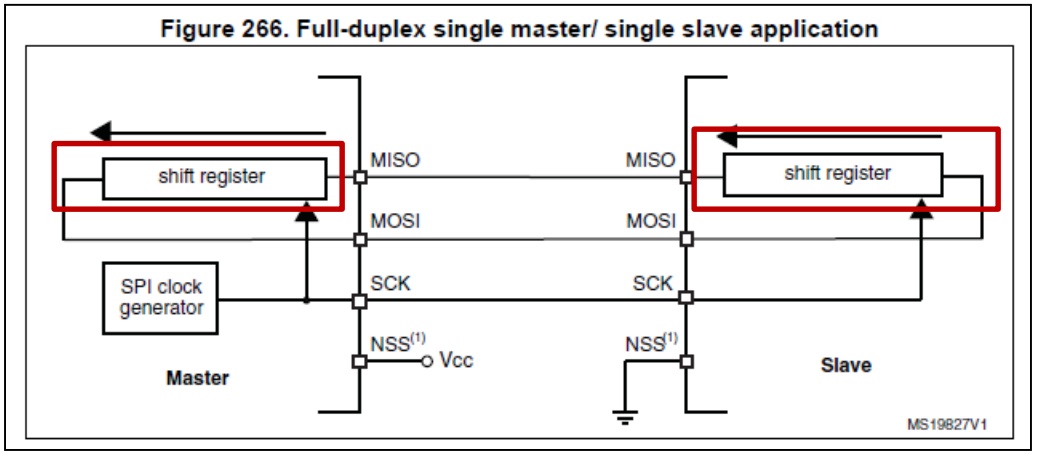


Serial Peripheral Interface

Principles of data transfer: full duplex

Serial Peripheral Interface
Principles of data transfer: full duplex

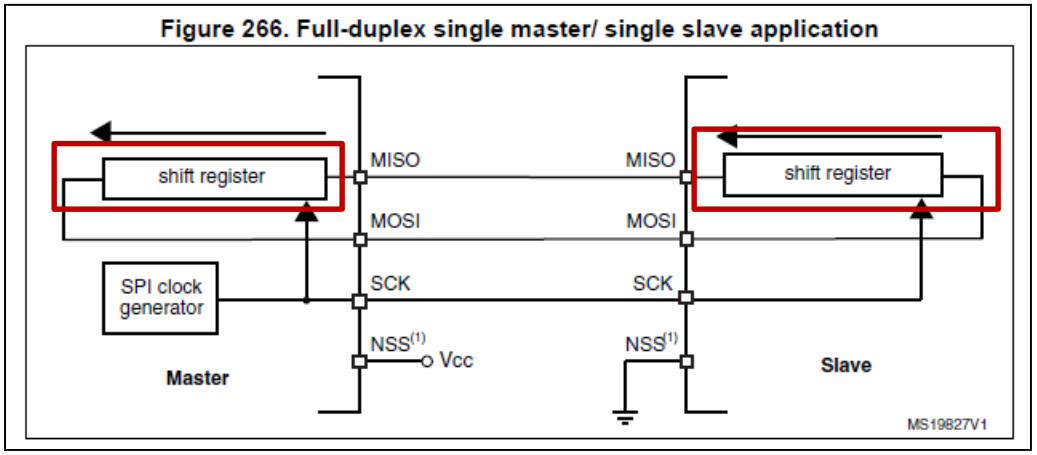
- Clock signal (SCK) **simultaneously** shifts data out of both shift registers



Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Principles of data transfer: full duplex

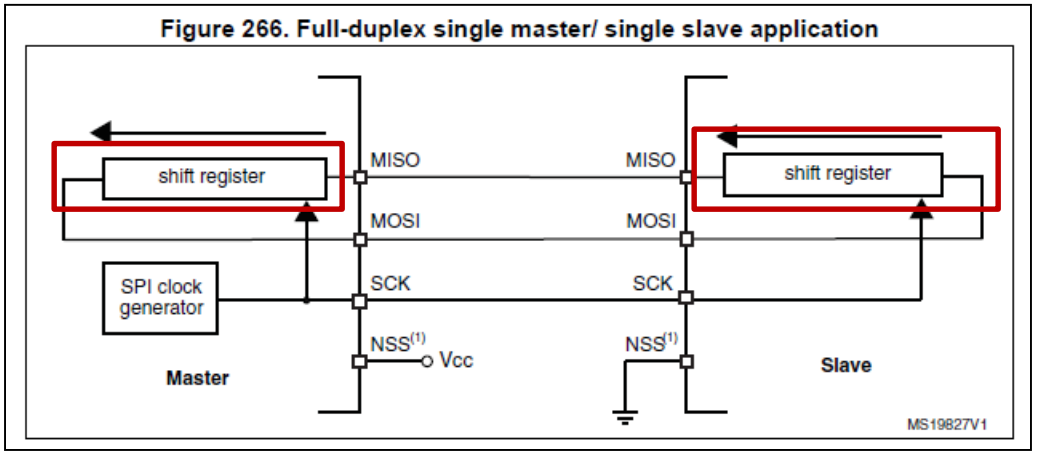
- Clock signal (SCK) **simultaneously** shifts data out of both shift registers
- Master sending data to slave
  - When master shifts the MSB of data from its shift register to the slave, the slave also shifts the MSB of data to the master. **Master receives 'dummy' data**.



Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Principles of data transfer: full duplex

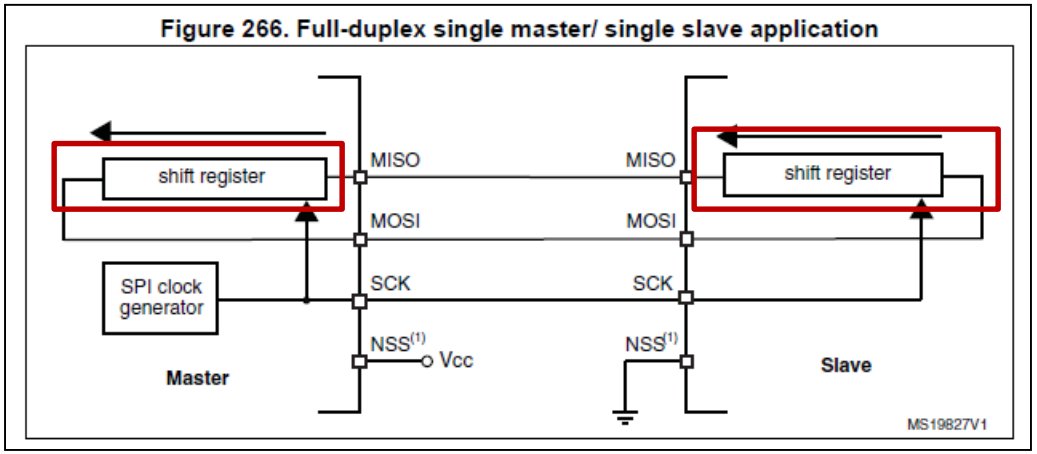
- Clock signal (SCK) **simultaneously** shifts data out of both shift registers
- Master sending data to slave
  - When master shifts the MSB of data from its shift register to the slave, the slave also shifts the MSB of data to the master. **Master receives 'dummy' data**.
- Master receiving data from the slave
  - When the master receives a bit of data from the slave, a bit of data is transferred to the slave as well. **Slave receives 'dummy' data.**



Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Principles of data transfer: full duplex

- Clock signal (SCK) **simultaneously** shifts data out of both shift registers
- Master sending data to slave
  - When master shifts the MSB of data from its shift register to the slave, the slave also shifts the MSB of data to the master. **Master receives 'dummy' data**.
- Master receiving data from the slave
  - When the master receives a bit of data from the slave, a bit of data is transferred to the slave as well. **Slave receives 'dummy' data**.
- When a device transmits data, it **simultaneously** receives data as well. The **'dummy' data** received must be read and ignored.



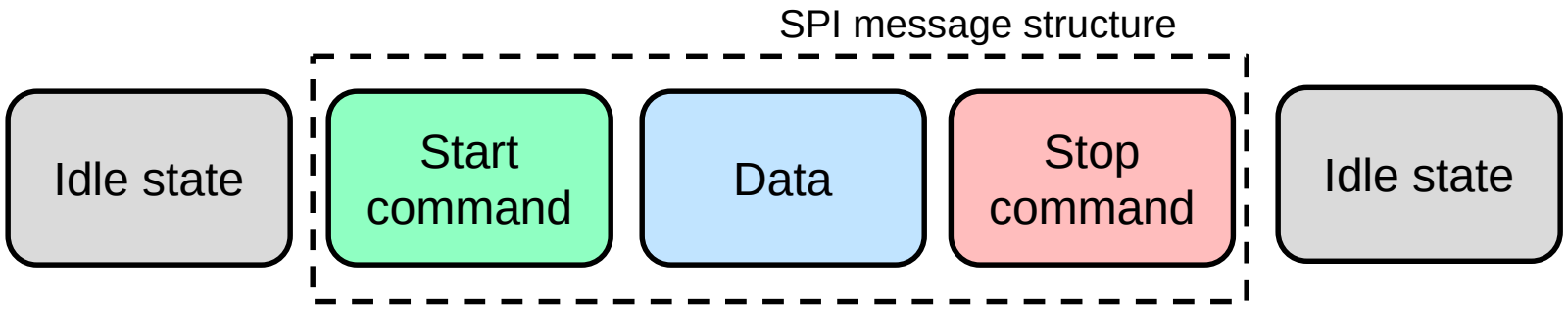
Extracted from the STM32F051 reference manual

Serial Peripheral Interface

Message Structure

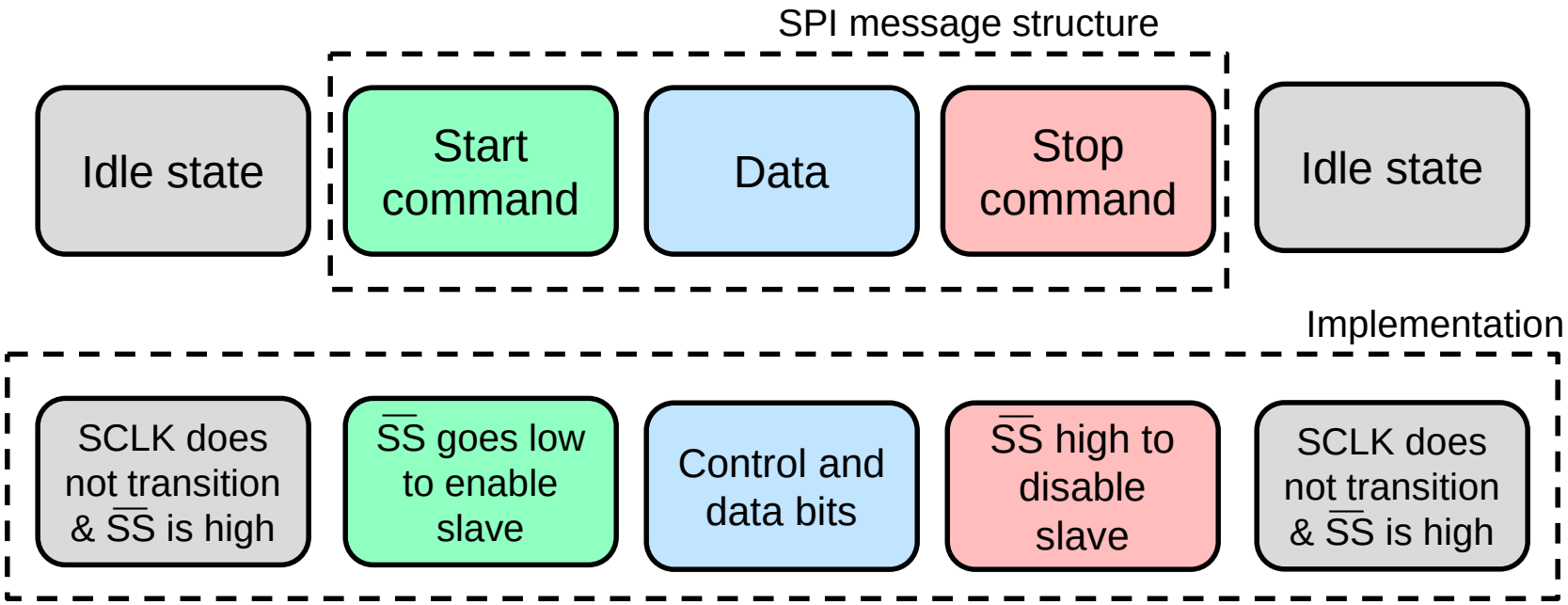
Serial Peripheral Interface

Message Structure



Serial Peripheral Interface

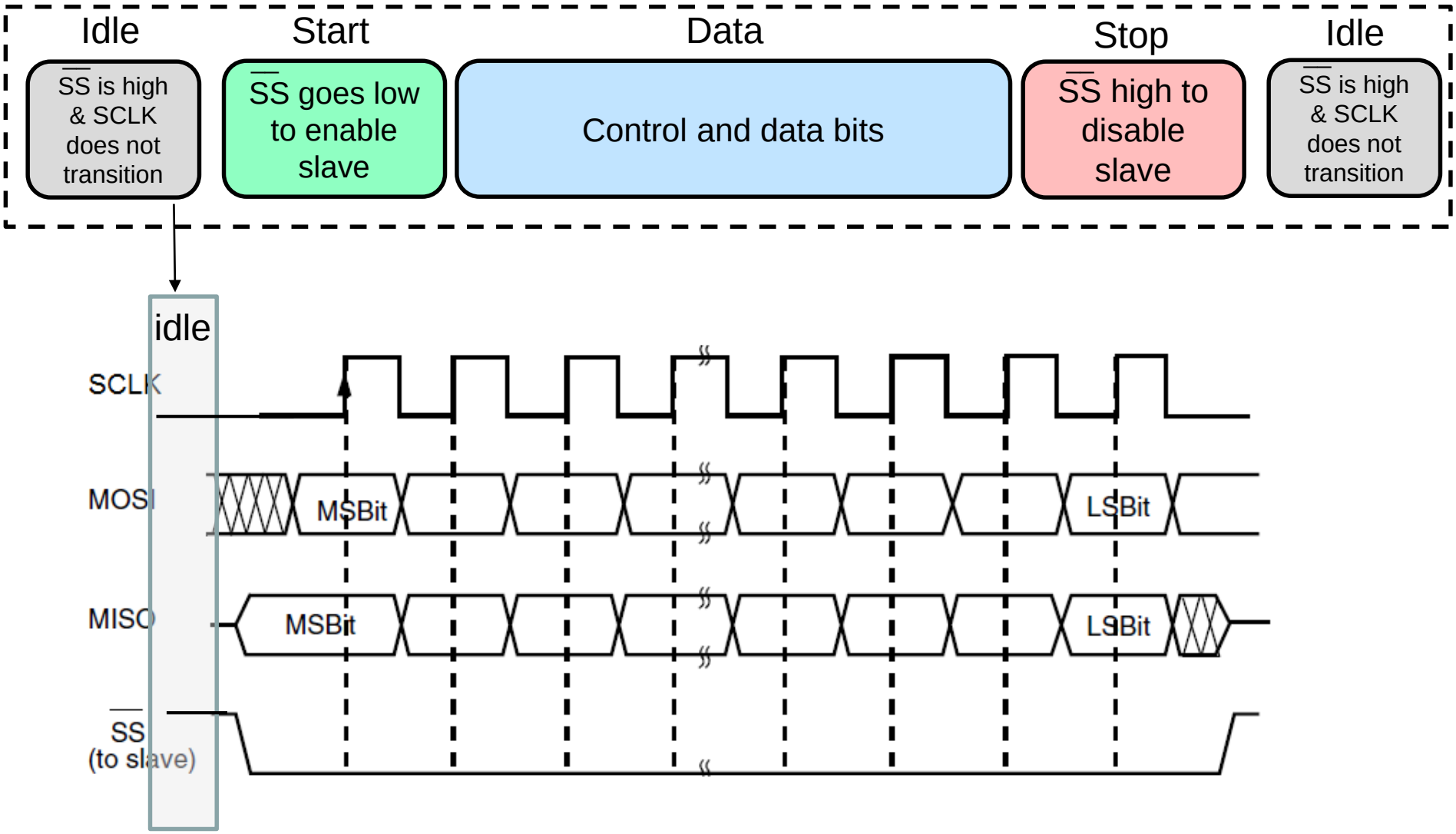
Message Structure



Extracted from the STM32F051 reference manual

Serial Peripheral Interface

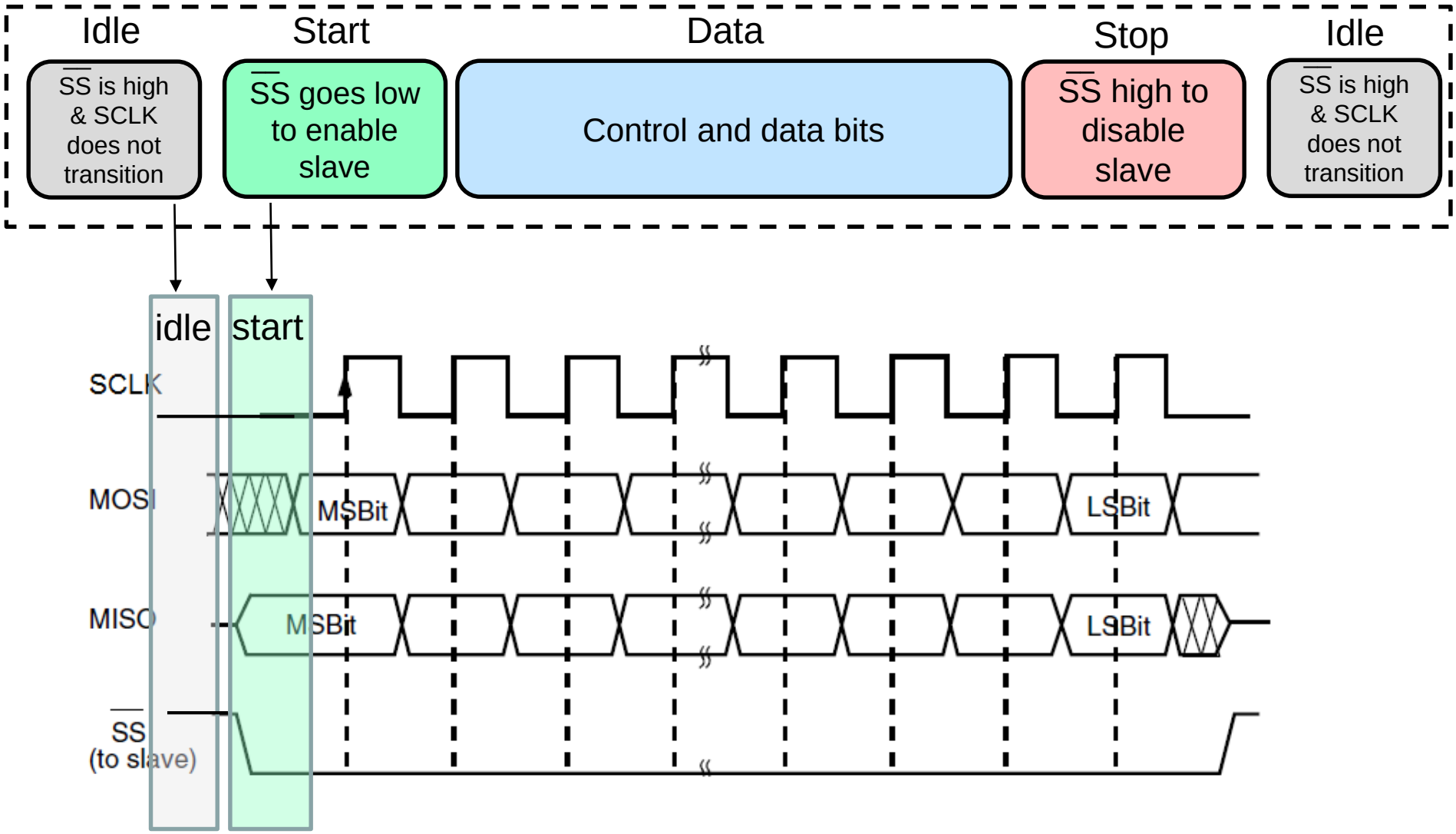
Message Structure



Extracted from the STM32F051 reference manual

Serial Peripheral Interface

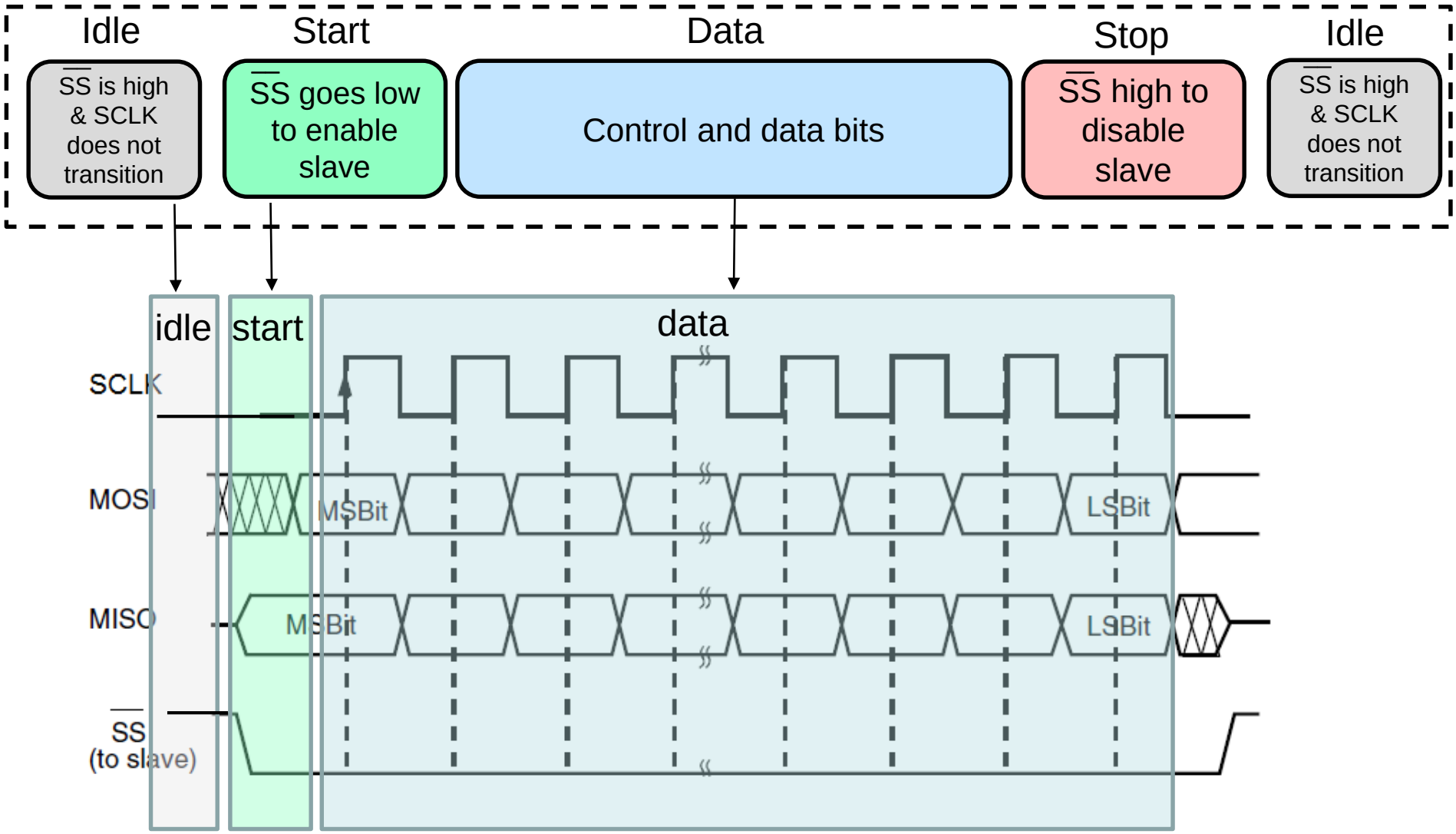
Message Structure



Extracted from the STM32F051 reference manual

Serial Peripheral Interface

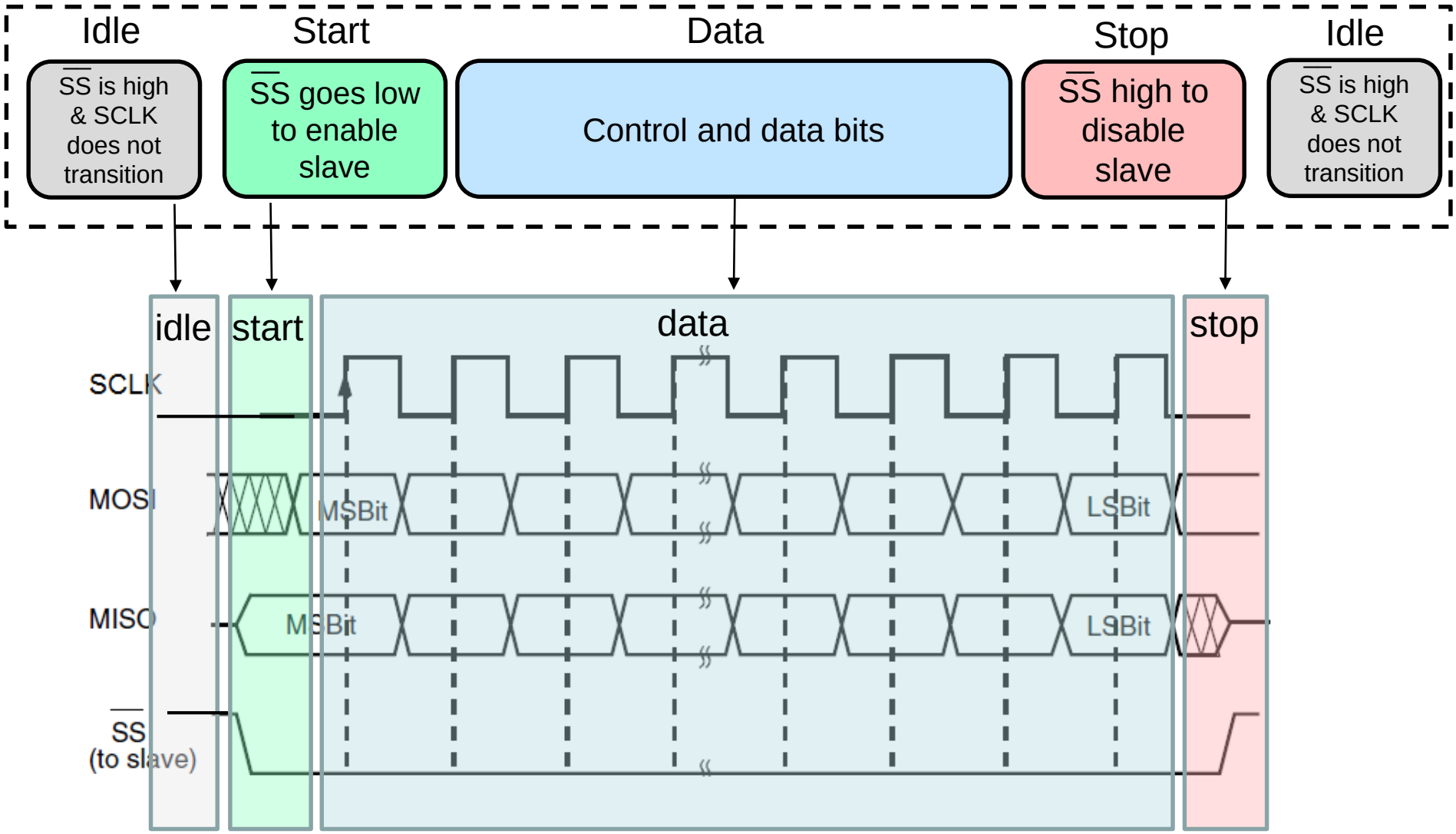
Message Structure



Extracted from the STM32F051 reference manual

Serial Peripheral Interface

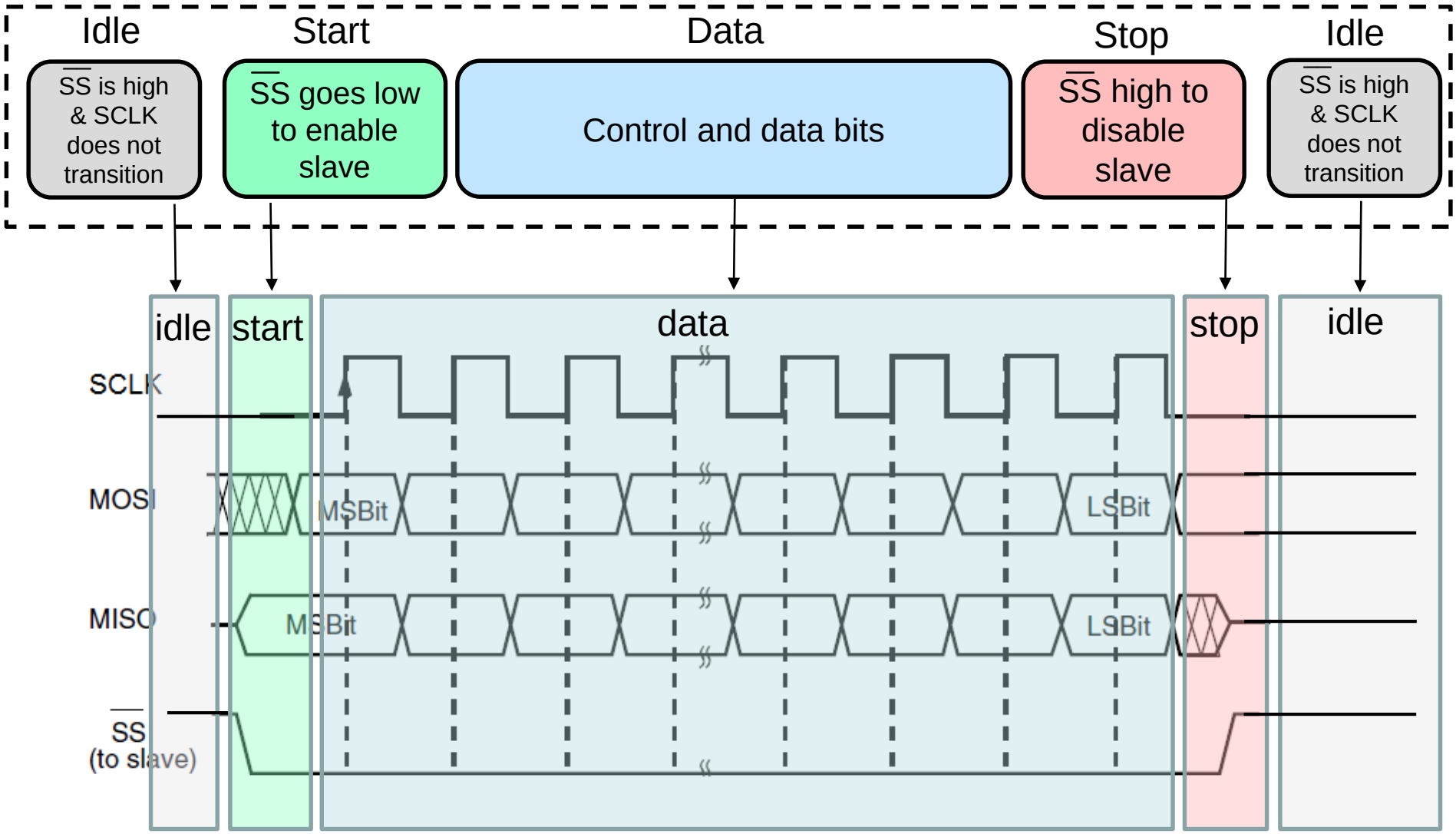
Message Structure



Extracted from the STM32F051 reference manual

Serial Peripheral Interface

Message Structure



Extracted from the STM32F051 reference manual

Serial Peripheral Interface

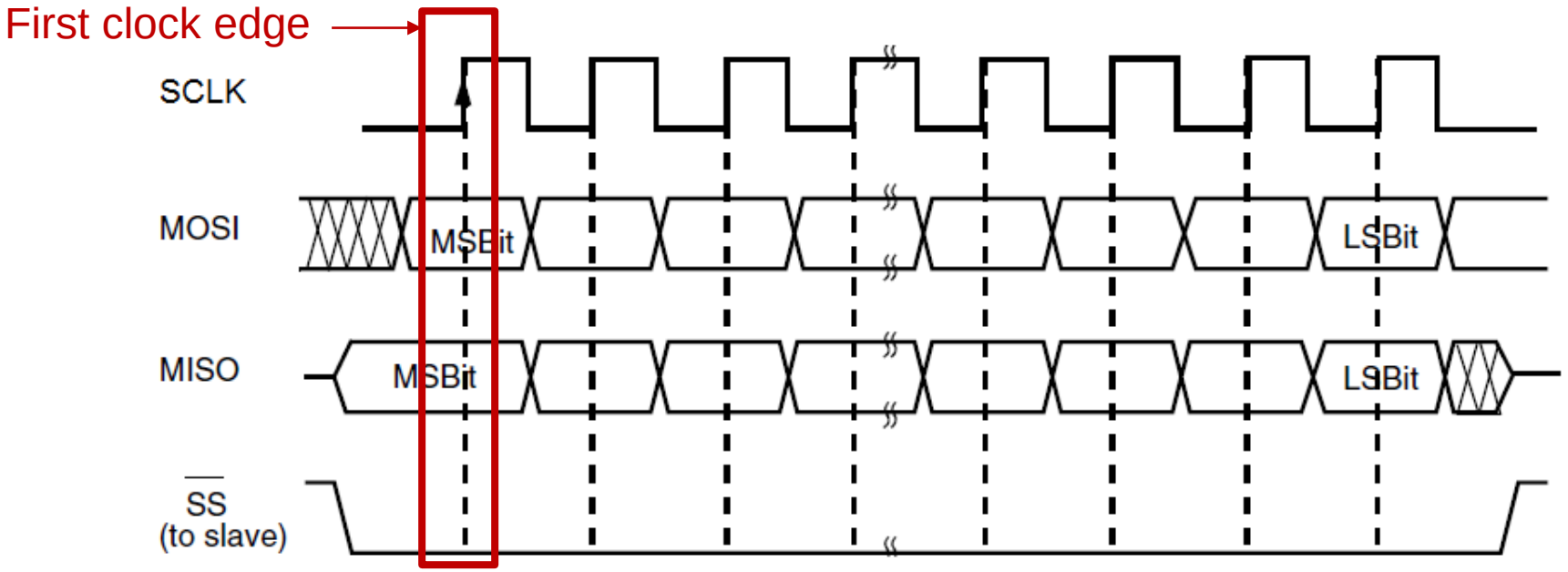
Timing and SPI modes

Serial Peripheral Interface
Timing and SPI modes

- When is the received signal sampled?
  - The received signal can be sampled on the **first clock edge or the second clock edge**. An edge is either a rising edge or a falling edge.

Serial Peripheral Interface
Timing and SPI modes

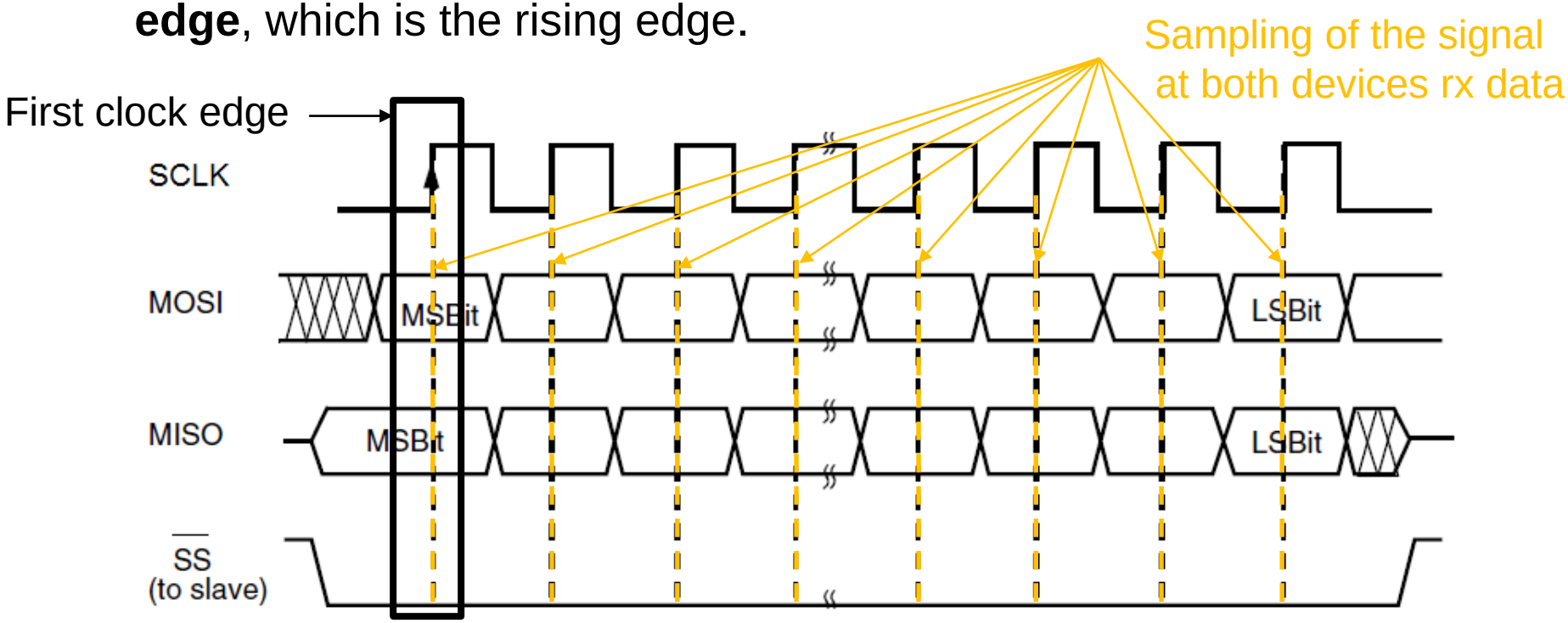
- When is the received signal sampled?
  - The received signal can be sampled on the **first clock edge or the second clock edge**. An edge is either a rising edge or a falling edge.
  - In diagram below, the received signal is sampled on the **first clock edge**, which is the rising edge.



Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Timing and SPI modes

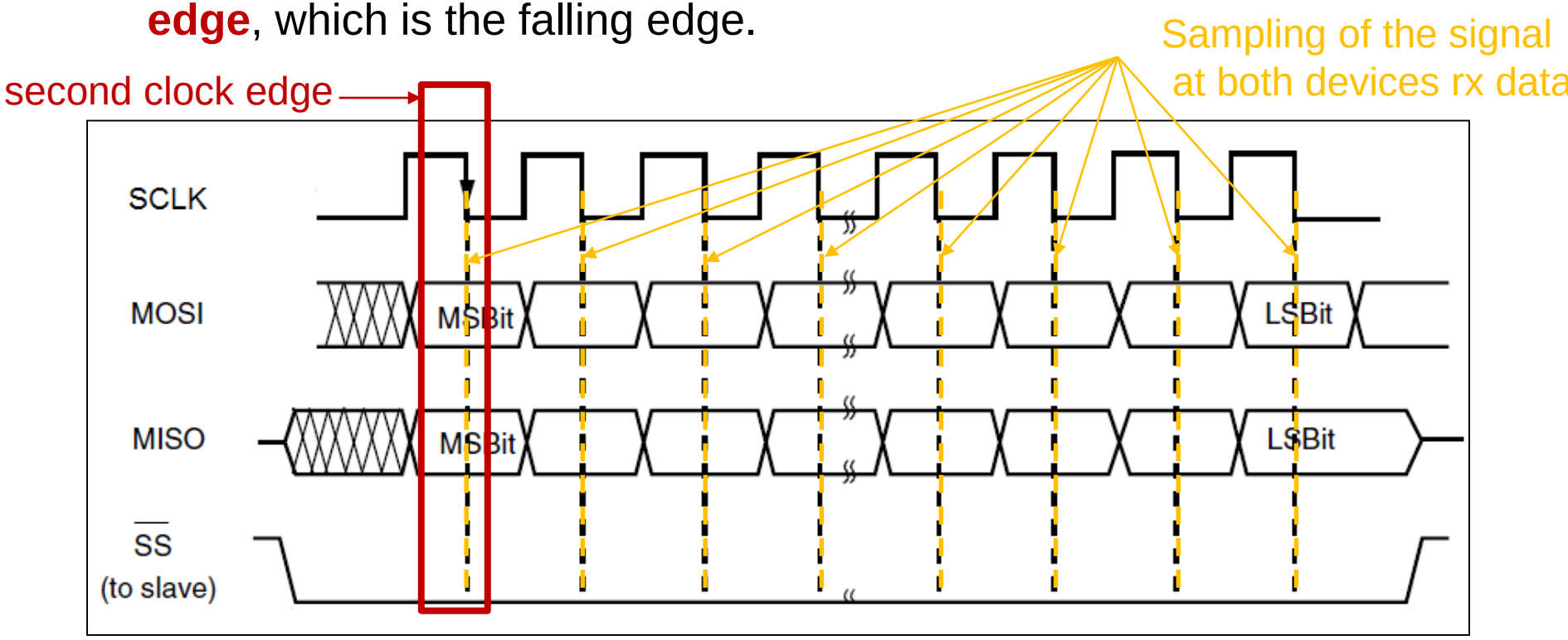
- When is the received signal sampled?
  - The received signal can be sampled on the **first clock edge or the second clock edge**. An edge is either a rising edge or a falling edge.
  - In diagram below, the received signal is sampled on the **first clock edge**, which is the rising edge.



Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Timing and SPI modes

- When is the received signal sampled?
  - The received signal can be sampled on the **first clock edge or the second clock edge**. An edge is either a rising edge or a falling edge.
  - In diagram below, the received signal is sampled on the **second clock edge**, which is the falling edge.



Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Timing and SPI modes

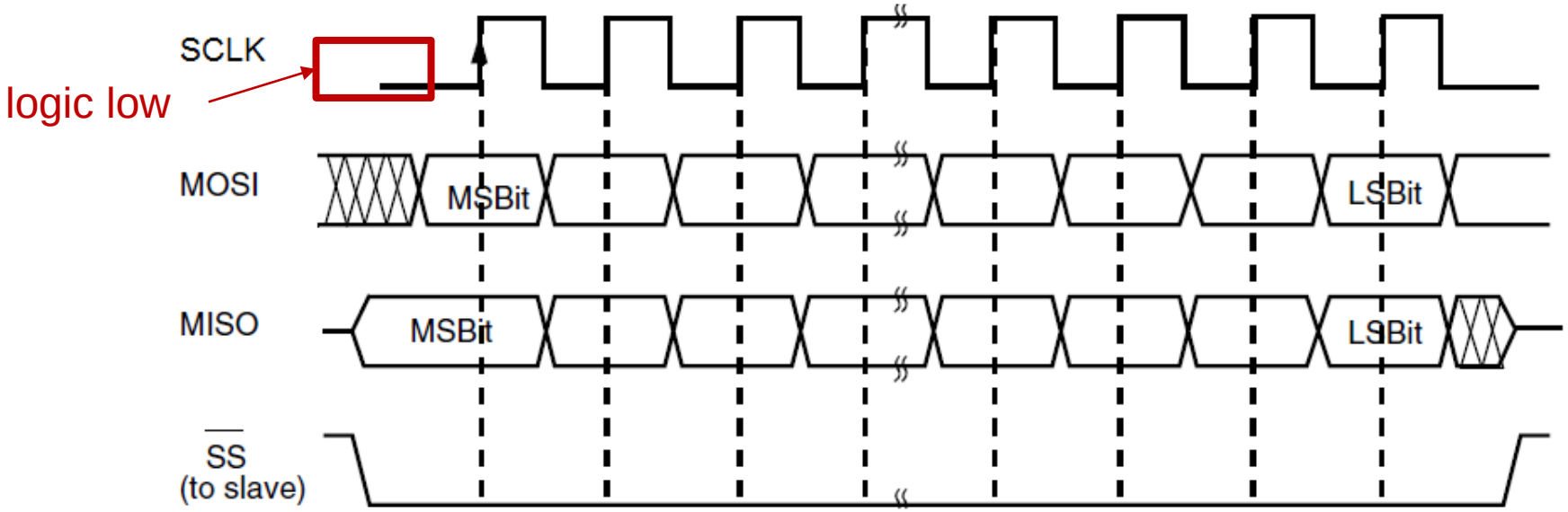
- When is the received signal sampled?
  - The received signal can be sampled on the **first clock edge or the second clock edge**. An edge is either a rising edge or a falling edge.
  - When the received signal is sampled on first clock edge, then clock phase (CPHA) = 0
  - When the received signal is sampled on the second clock edge, then the clock phase (CPHA) = 1

Table summarising SPI modes (incomplete)

Sampling of the received signal		
Clock phase		
CPHA = 0		
CPHA = 1		
CPHA = 0		
CPHA = 1		

Serial Peripheral Interface
Timing and SPI modes

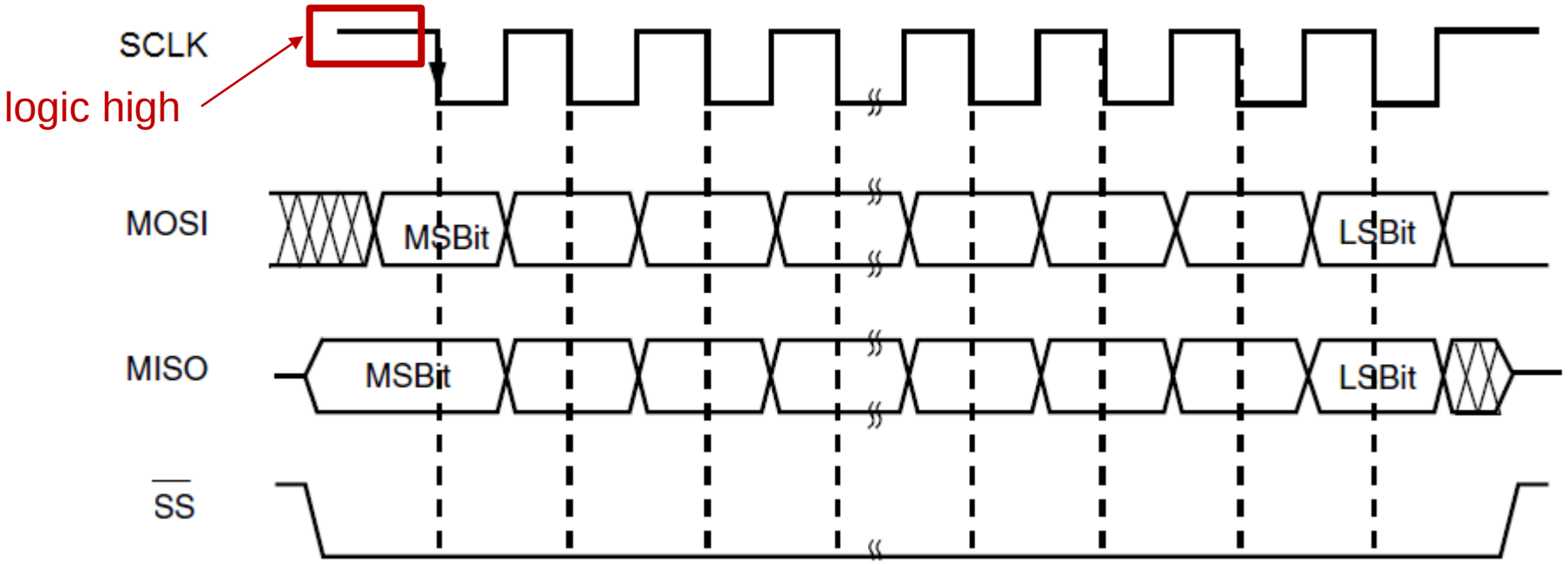
- During idle, what is the clock's logic state?
  - During idle, the clock's logic state may be **logic high** or **logic low**.
- In the diagram below, the **clock's state is low**, during idle



Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Timing and SPI modes

- During idle, what is the clock's logic state?
  - During idle, the clock's logic state may be **logic high** or **logic low**.
- In the diagram below, the **clock's state is high**, during idle



Extracted from the STM32F051 reference manual

Serial Peripheral Interface
Timing and SPI modes

- During idle, what is the clock's logic state?
  - During idle, the clock's logic state may be **logic high** or **logic low**.
- Clock Polarity (CPOL)
  - When the clock's logic state is low, then clock polarity (CPOL) = 0
  - When the clock's logic state is high, then clock polarity (CPOL) = 1

Table summarising SPI modes (incomplete)

Sampling of the received signal	Logic state of the clock during idle	
Clock phase	Clock Polarity	
CPHA = 0	CPOL = 0	
CPHA = 1	CPOL = 0	
CPHA = 0	CPOL = 1	
CPHA = 1	CPOL = 1	

Serial Peripheral Interface
Timing and SPI modes

- During idle, what is the clock's logic state?
  - During idle, the clock's logic state may be **logic high** or **logic low**.
- Clock Polarity (CPOL)
  - When the clock's logic state is low, then clock polarity (CPOL) = 0
  - When the clock's logic state is high, then clock polarity (CPOL) = 1
- SPI modes: 0 to 3. It depends on both CPHA and CPOL

Table summarising SPI modes (complete)

Sampling of the received signal	Logic state of the clock during idle	SPI modes
Clock phase	Clock Polarity	
CPHA = 0	CPOL = 0	0
CPHA = 1	CPOL = 0	1
CPHA = 0	CPOL = 1	2
CPHA = 1	CPOL = 1	3