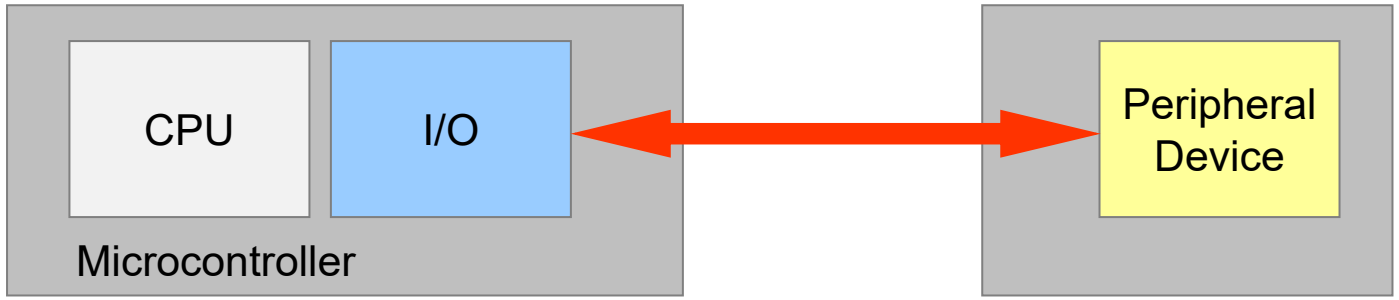
Serial Data Transfer – SPI

Computer Engineering 2

Motivation

- **Communication CPU – Peripheral Devices**

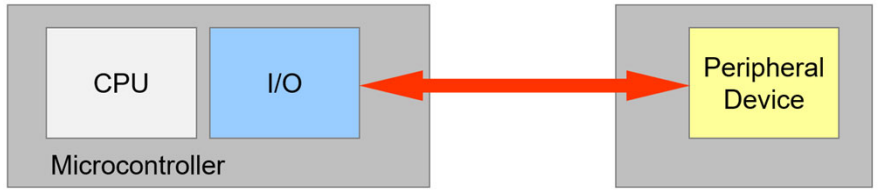


- **Parallel Bus**
  - n address lines
  - m data lines
  - Control lines (NWE, NOE)
  - Decode logic

- **Serial Bus**
  - 2 to 4 lines
    - (CLK)
    - Data: Din, Dout
    - (Select)

Motivation

- **Serial Connection → UART, SPI, I2C, etc.**
  - Provide simple, low-level physical connection
    - Simpler → save PCB area
    - Reduce number of switching lines → reduce power, and improve EMC [1)]



  - Requires "higher level" protocol
    - Usually in software
    - Error detection, reliability, quality-of-service (QoS)
    - Interpretation of commands

1) Electromagnetic compatibility:
Simultaneous switching of lines creates unwanted emission

Agenda

School of Engineering
InES Institute of Embedded Systems

- **Serial Communication**
- **SPI – Serial Peripheral Interface**
  - SPI Basics
  - SPI Modes
  - SPI – STM32F4xxx
  - SPI – Flash Devices

Learning Objectives

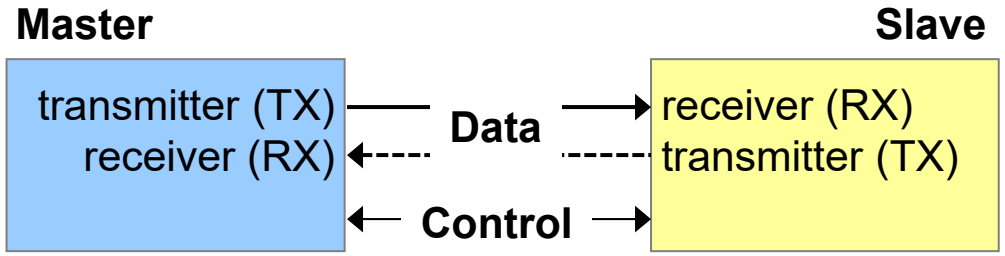
At the end of this lesson, you will be able

- to explain what SPI is and how it works
- to outline the differences between the four SPI modes of operation
- to draw and interpret SPI timing diagrams
- to interpret the SPI block diagram of the STM32F4xxx
- to outline how SPI data transmission and reception has to be handled by software on the STM32F4xxx

Serial Communication

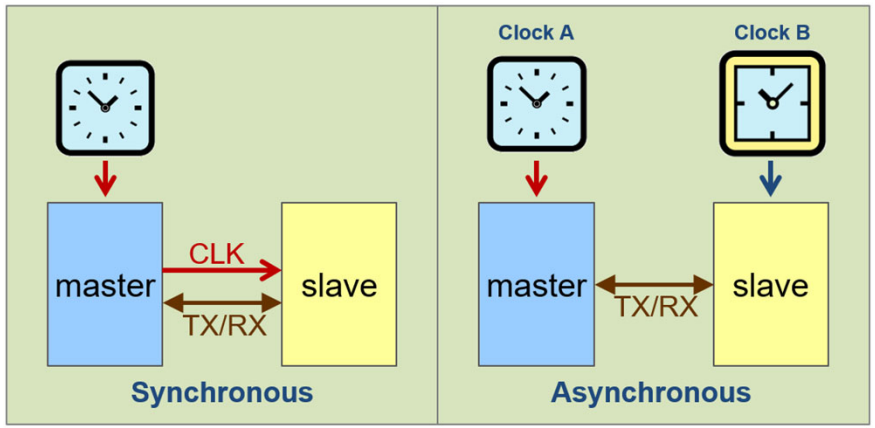
School of Engineering
InES Institute of Embedded Systems

- **Connections (wires)**
  - Serial data line(s)
  - Optional control lines



- **Communication Modes**
  - Simplex — Unidirectional, one-way only
  - Half-duplex — Bidirectional, only one direction at a time
  - Full-duplex — Bidirectional, both directions simultaneously

- **Timing**
  - Synchronous — Both nodes use the same clock
Clock often provided by master
  - Asynchronous — Each node uses an individual clock



SPI / I2C UART

SPI Basics

School of Engineering
InES Institute of Embedded Systems

- **SPI – Serial Peripheral Interface**
  - Serial bus for on-board connections
    - Used for short distance communication
  - Connects microcontroller and external devices
    - Sensors, A/D converters, displays, flash memories, codecs
    - IOs, Real Time Clocks (RTC), wireless transceivers...
  - Synchronous
    - Master distributes the clock to slaves
  - Compared to a parallel bus
    - Saves board area
    - Lowers pin count on both chips (TX and RX) → smaller, low-cost
    - Simplifies EMC (Electromagnetic compatibility)

SPI Basics

School of Engineering
InES Institute of Embedded Systems

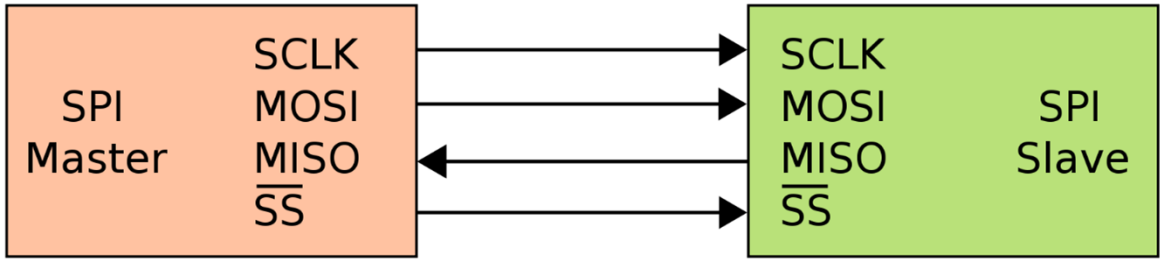
- **SPI → De facto Standard**
  - No official standards organization
  - No legally binding specification
    - Only chip datasheets and application notes
    - Many different variants exist
  - Introduced by Motorola (today NXP) around 1979
  - Also called 4-wire Bus

A de facto standard is a custom, convention, product, or system that has achieved a dominant position by public acceptance or market forces.

source: Wikipedia

SPI Basics

- **Synchronous Serial Data Connection**
  - Full duplex

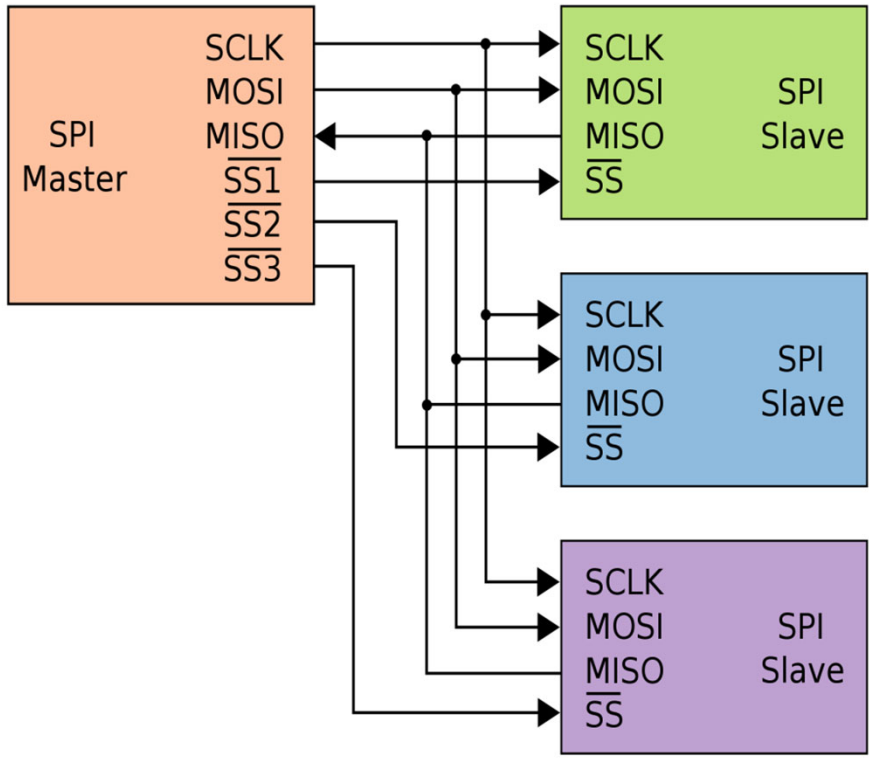


Source: Wikipedia

  - Master (single master)
    - Generates clock (SCLK)
    - Initiates data transfer by setting $\overline{SS}$ = 0 (Slave Select)
  - MOSI – Master Out Slave In
    - Data from master to slave
  - MISO – Master In Slave Out
    - Data from slave to master

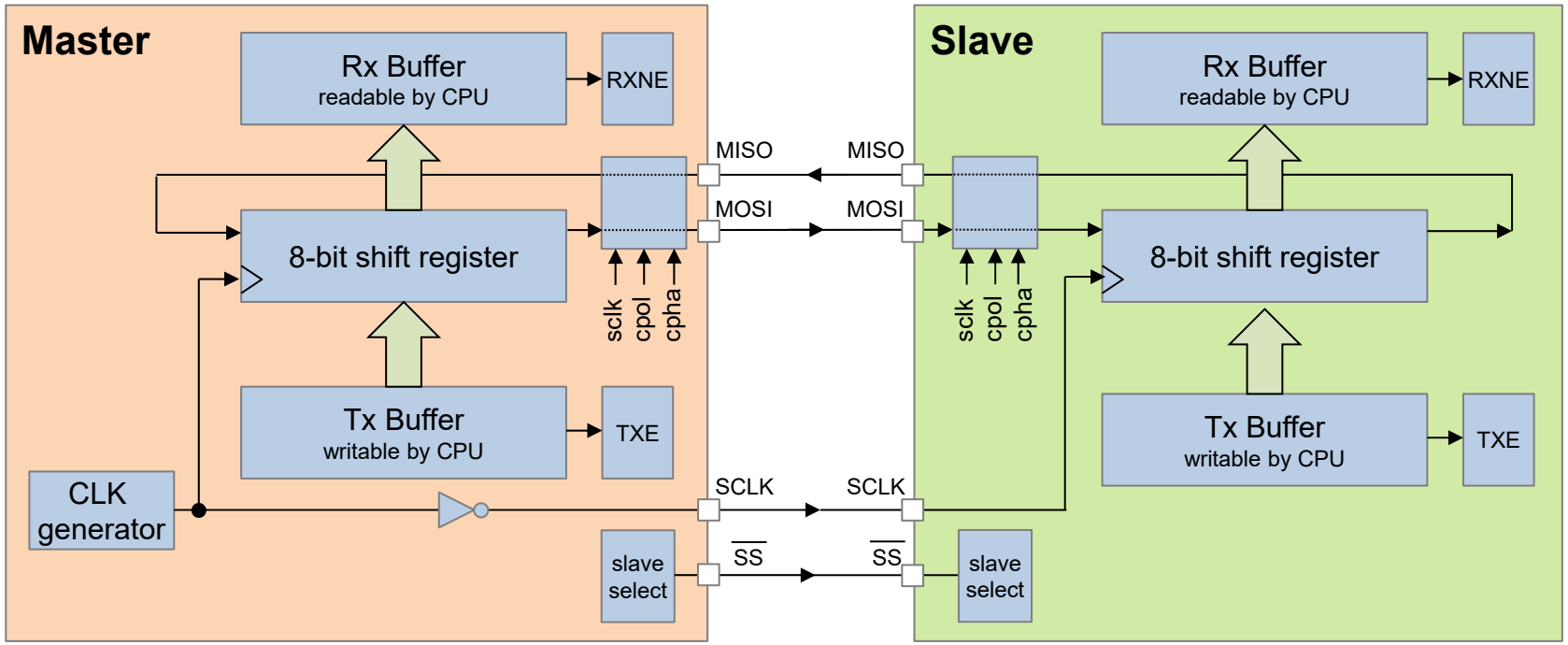
SPI Basics

- **Single Master – Multiple Slaves**
  - Master generates a common clock signal for all slaves
  - MOSI
    - From master output to all slave inputs
  - MISO
    - All slave outputs connected to single master input
  - Slaves
    - Individual select $\overline{SS1}$, $\overline{SS2}$, $\overline{SS3}$
    - $\overline{SSx}$ = '1' → Slave output MISOx is tri-state



Source: Wikipedia

Implementation Using Shift Registers



'LSB first' vs. *'MSB first'* is configurable in most microcontrollers.
Slaves are often hard-wired.

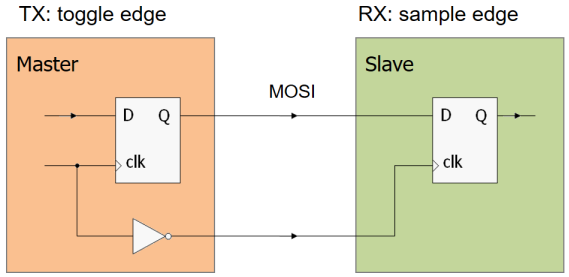
Status bits with Interrupt

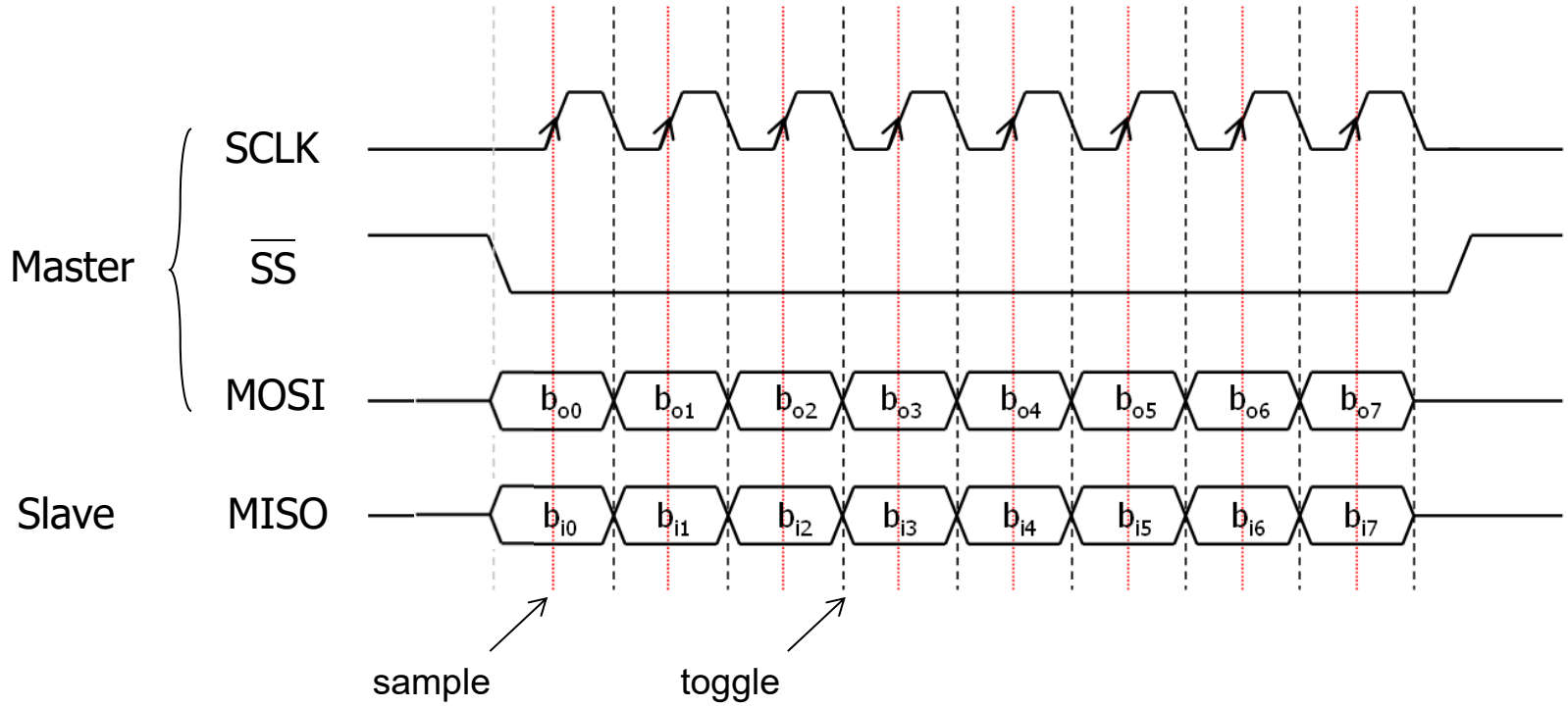
TXE	Tx Buffer Empty
RXNE	Rx Buffer Not Empty

SPI Basics

School of Engineering
InES Institute of Embedded Systems

- **Timing**
  - Toggle output on one clock edge
  - Sample on other clock edge

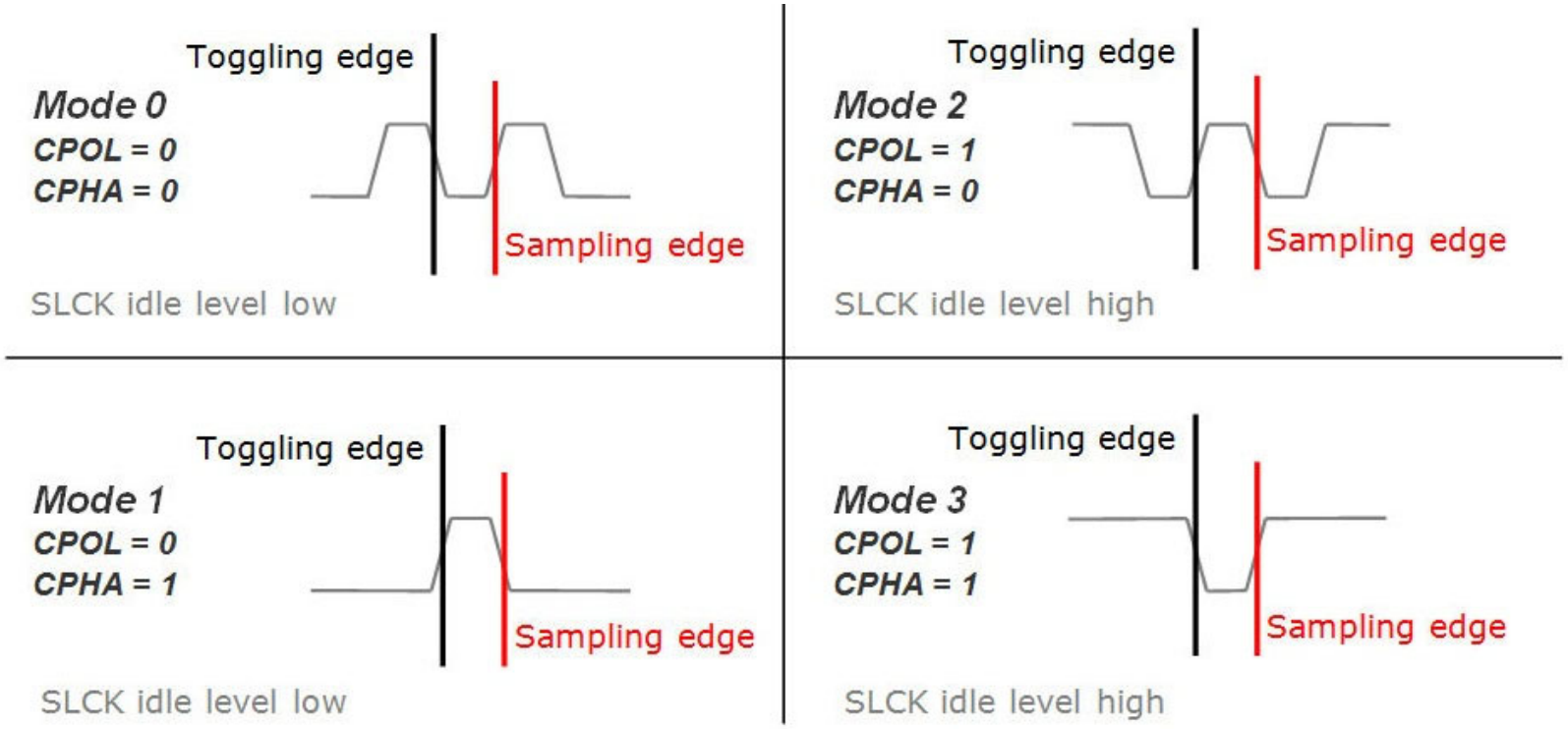




SPI Modes

- **Clock Polarity and Clock Phase**
  - TX provides data on 'Toggling Edge'
  - RX takes over data with 'Sampling Edge'

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1



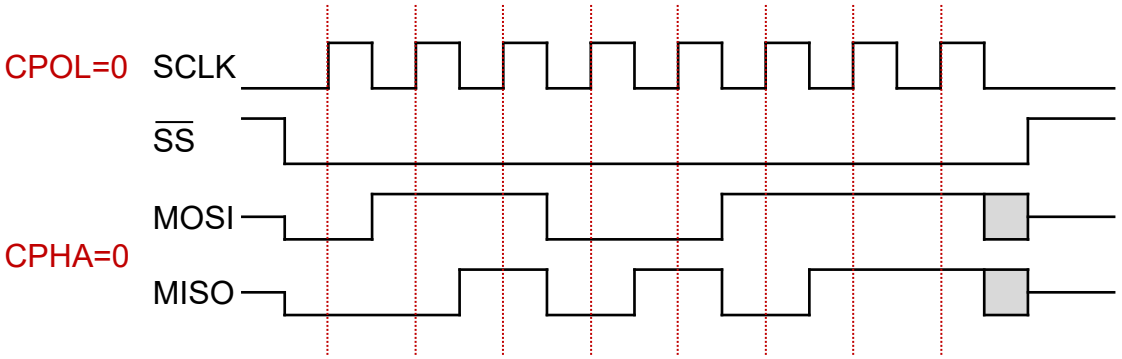
Source: http://www.byteparadigm.com

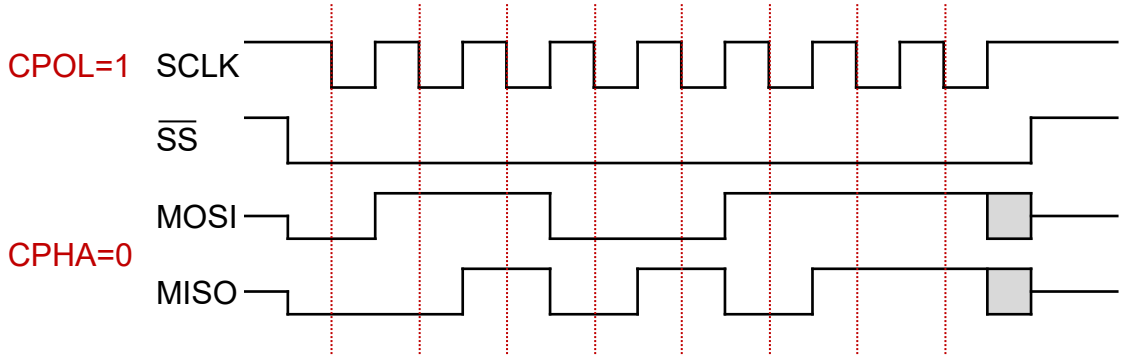
SPI Modes

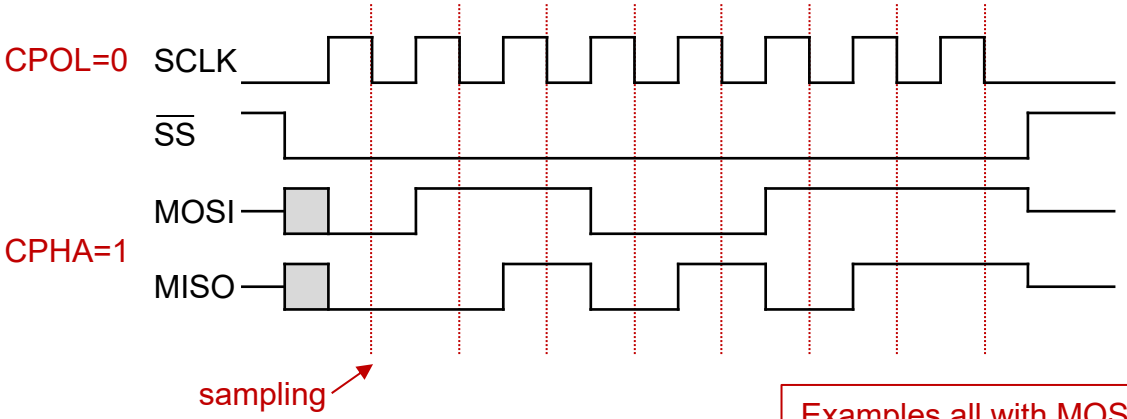
Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

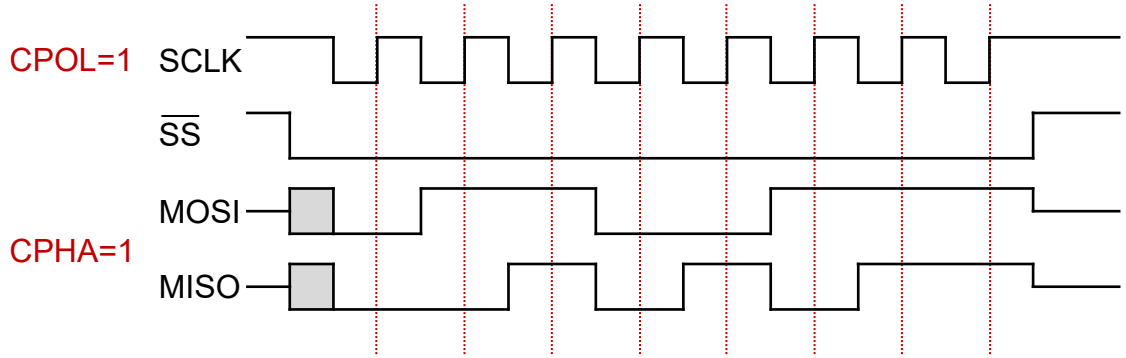
School of Engineering
InES Institute of Embedded Systems

- **Clock Polarity and Clock Phase**
  - TX provides data on 'Toggling Edge‘ RX takes over data with 'Sampling Edge'







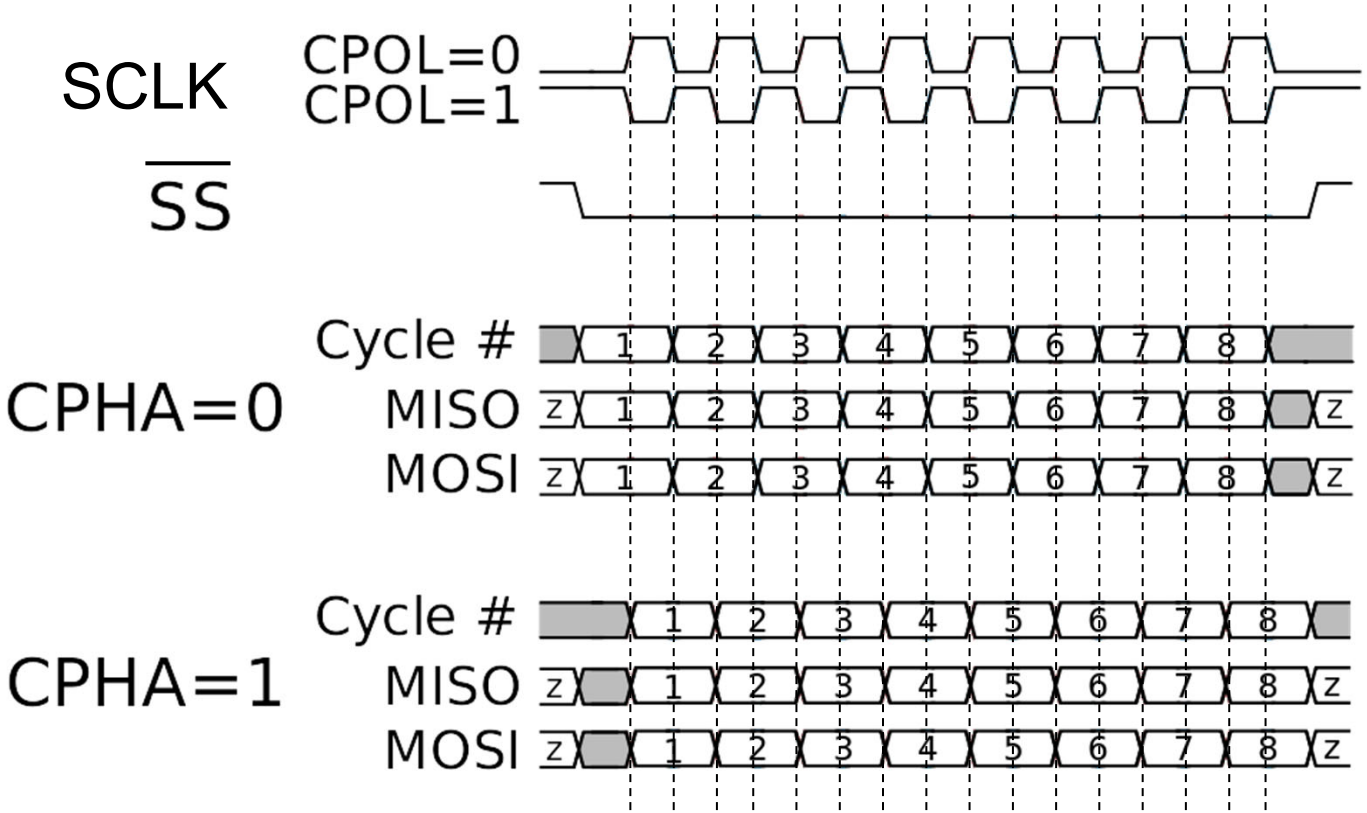


Examples all with MOSI = 0x67 and MISO = 0x2B, MSB first

SPI Modes

- **Data and Clock**
  - Summary of all possible combinations

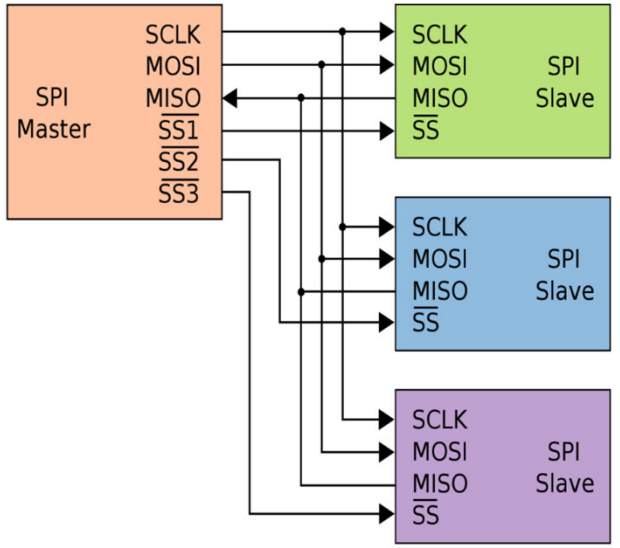
Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1



Source: Wikipedia

SPI

- **Properties**
  - No defined addressing scheme
    - Use of $\overline{SS}$ instead → KISS
  - Transmission without receive acknowledge and error detection
    - Has to be implemented in higher level protocols
  - Originally used only for transmission of single bytes
    - $\overline{SS}$ deactivated after each byte
    - Today also used for streams (endless transfers)
  - Data rate
    - Highly flexible as clock signal is transmitted
  - No flow-control available
    - Master can delay the next clock edge
    - Slave can not influence the data rate
  - Susceptible to spikes on clock line

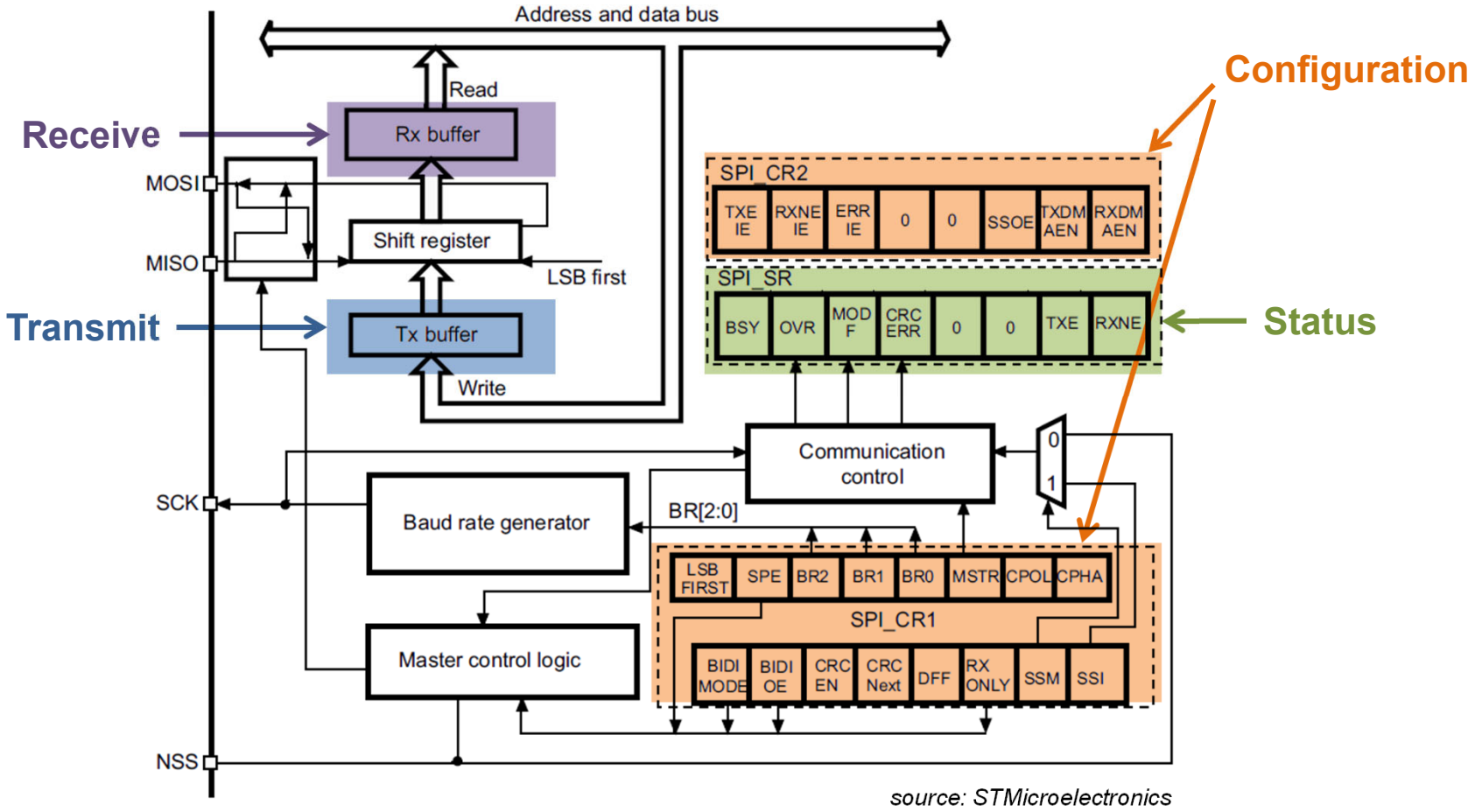


Source: Wikipedia

School of Engineering
InES Institute of Embedded Systems

- **Implementation**

Receive data	read from SPI_DR register
Transmit data	write to SPI_DR register

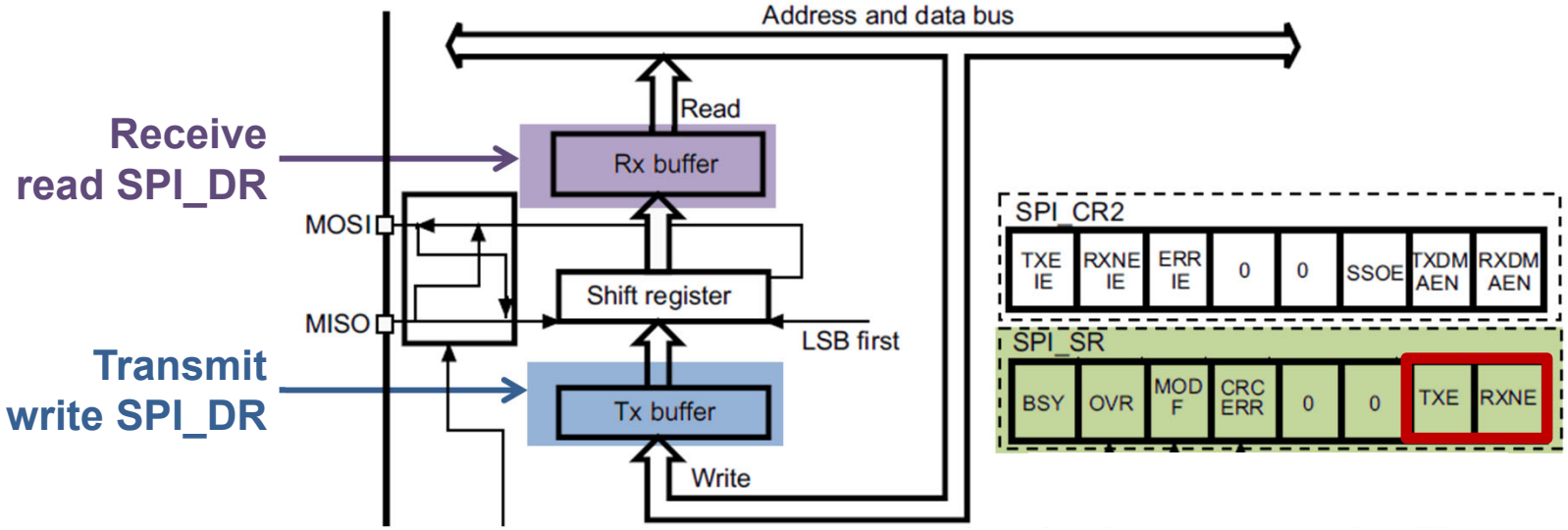


source: STMicroelectronics

School of Engineering
InES Institute of Embedded Systems

Synchronizing Hardware and Software

- When shall software access the shift register?
  - **TXE** **TX Buffer Empty** **Software can write next TX Byte to register SPI_DR**
  - **RXNE** **RX Buffer Not Empty** **A byte has been received. Software can read it from SPI_DR**



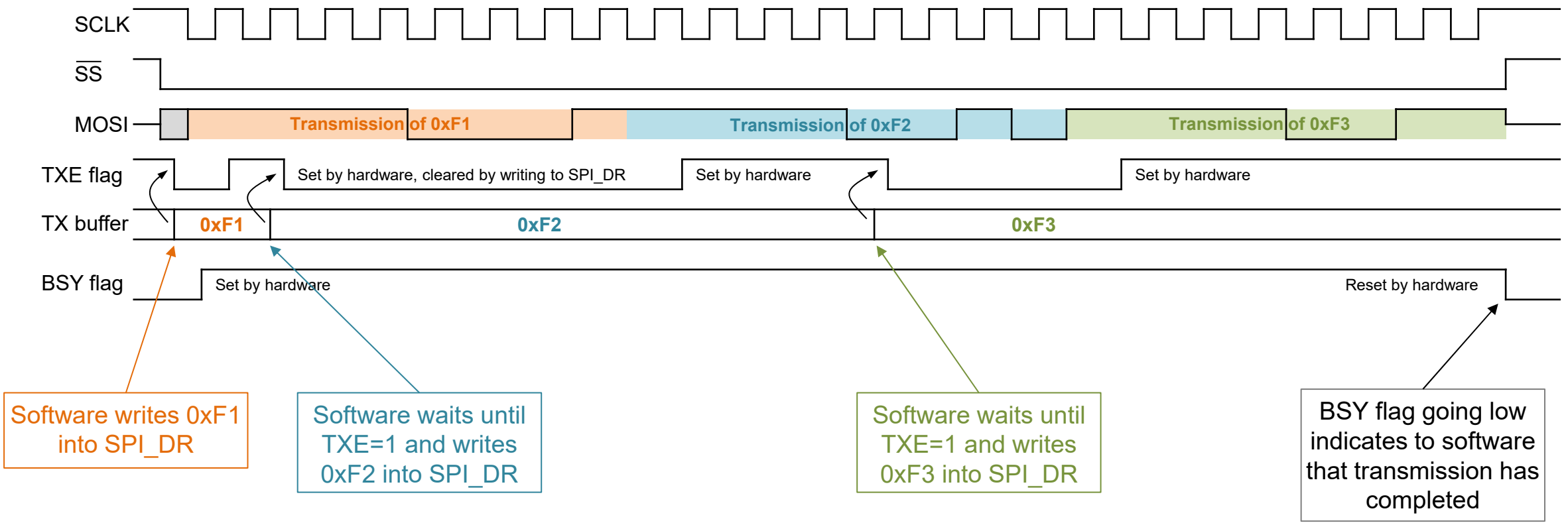
source: STMicroelectronics

School of Engineering
InES Institute of Embedded Systems

Transmitting in Software

Example with
CPOL=1, CPHA=1, MSB first

- Example: SW wants to transmit bytes 0xF1, 0xF2, 0xF3

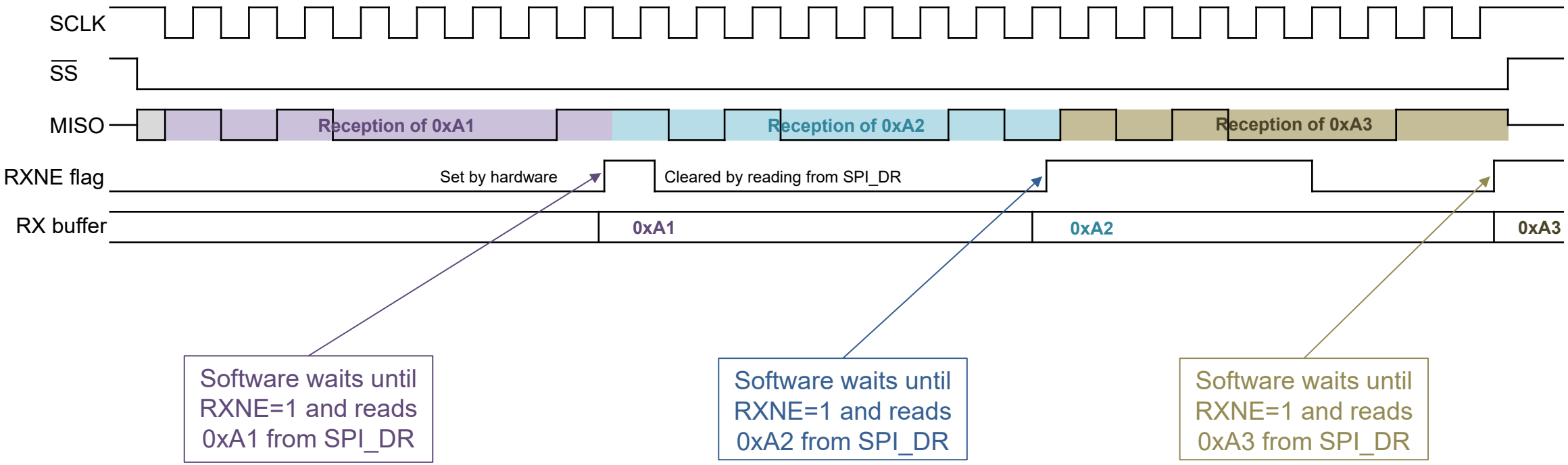


School of Engineering
InES Institute of Embedded Systems

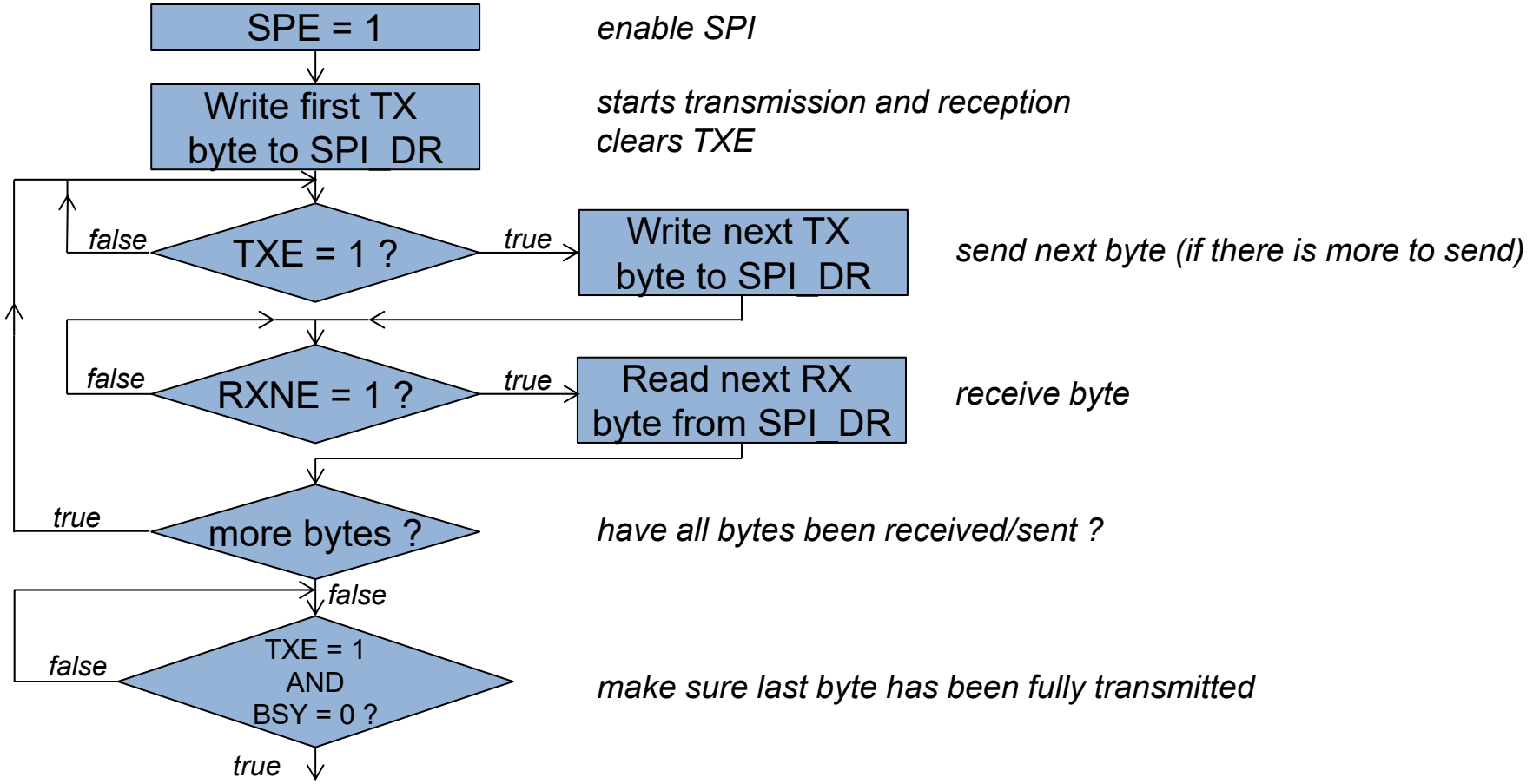
Receiving in Software

Example with
CPOL=1, CPHA=1, MSB first

- Example: SW receives bytes 0xA1, 0xA2, 0xA3



- **Software: Simultaneously Handling Data Transmission and Reception**
  - Full duplex: Check TXE and RXNE bits



- **Interrupts**
  - Interrupts can be used instead of polling for TXE and RXNE bits

Position	Priority	Type of priority	Acronym	Description	Address
31	38	settable	I2C1_EV	I^2C1 event interrupt	0x0000 00BC
32	39	settable	I2C1_ER	I^2C1 error interrupt	0x0000 00C0
33	40	settable	I2C2_EV	I^2C2 event interrupt	0x0000 00C4
34	41	settable	I2C2_ER	I^2C2 error interrupt	0x0000 00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000 00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000 00D0
37	44	settable	USART1	USART1 global interrupt	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt	0x0000 00DC

SPI – STM32F4xxx

School of Engineering
InES Institute of Embedded Systems

- **SPI Registers**
  - Total of 6 SPI blocks
    - Set of registers for each of them

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																	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	*LSBFIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CR2																									TXEIE	RXNEIE	ERRIE	FRF	Reserved	*SSOE*	TXDMAEN	RXDMAEN
	Reset value																									0	0	0	0		0	0	0
0x08	SPI_SR																								FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																								0	0	0	0	0	0	0	1	0
0x0C	SPI_DR																	DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Configuration

Status

Transmit / Receive

source: STM32F42xxx Reference Manual

```
#define SPI1 ((reg_spi_t *) 0x40013000)
#define SPI2 ((reg_spi_t *) 0x40003800)
#define SPI3 ((reg_spi_t *) 0x40003c00)
#define SPI4 ((reg_spi_t *) 0x40013400)
#define SPI5 ((reg_spi_t *) 0x40015000)
#define SPI6 ((reg_spi_t *) 0x40015400)
```

Register Bits

SPI_CR1 SPI control register 1

BIDIMODE	Bidirectional data mode enable
BIDIOE	Output enable in bidir mode
CRCEN	Hardware CRC calculation enable
CRCNEXT	CRC transfer next
DFF	Data frame format (8-bit vs. 16-bit)
RXONLY	Receive only
SSM	Software slave management
SSI	Internal slave select
LSBFIRST	Frame format (bit order)
SPE	SPI enable
BR[2:0]	Baud rate control
MSTR	Master selection (master vs. slave)
CPOL	Clock polarity
CPHA	Clock phase

SPI_DR SPI data register

DR[15:0]	Data register

SPI_CR2 SPI control register 2

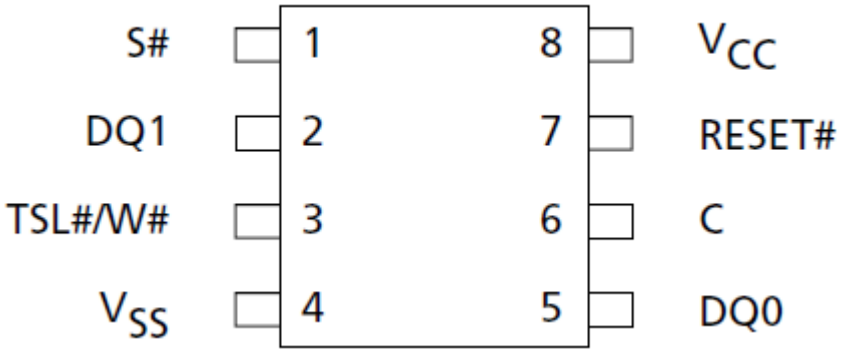
TXEIE	Tx buffer empty interrupt enable
RXNEIE	RX buffer not empty interrupt enable
ERRIE	Error interrupt enable
FRF	Frame format (Motorola vs. TI mode)
SSOE	SS output enable
TXDMAEN	Txbuffer DMA enable

SPI_SR SPI status register

FRE	Frame format error
BSY	Busy flag (Txbuffer not empty)
OVR	Overrun flag
MODF	Mode fault
CRCERR	CRC error flag
UDR	Underrun flag
CHSIDE	Channel side (not used for SPI)
TXE	Transmit buffer empty
RXNE	Receive buffer not empty

SPI – Flash Devices

- **Save Board Area**
  - E.g. Micron M25PE40 Serial Flash Memory
    - 4 Mbit NOR flash
    - 6 x 5 mm package size
    - SCLK: up to 75 MHz



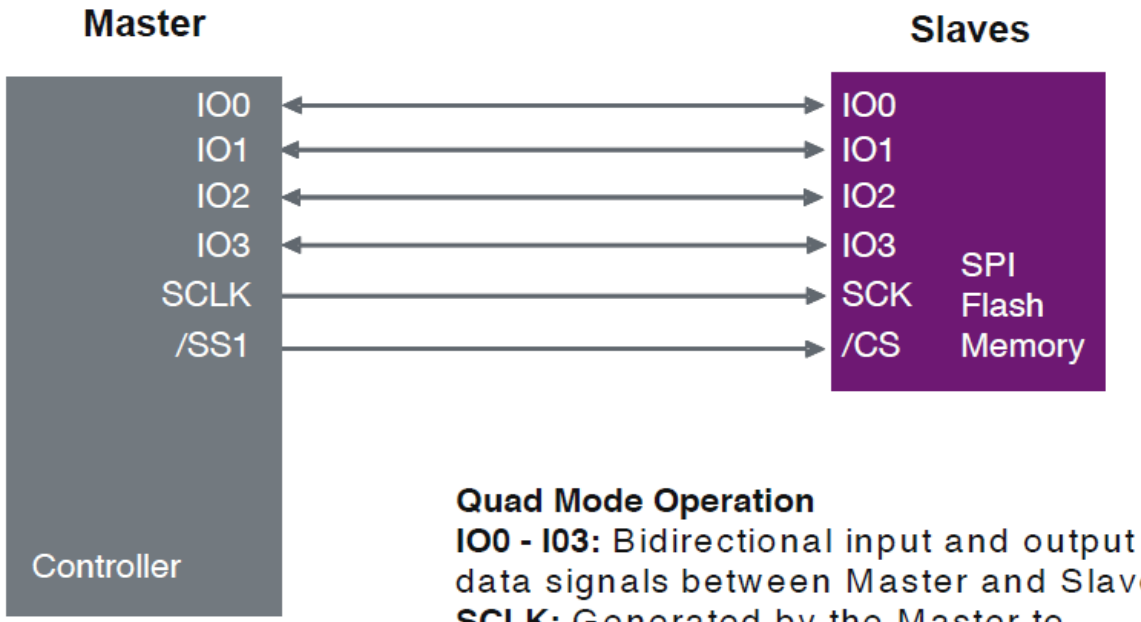
Signal Name	Function	Direction
C	Serial clock	Input
DQ0	Serial data	Input
DQ1	Serial data	Output
S#	Chip select	Input
W#	Write Protect	Input
RESET#	Reset	Input
V_{CC}	Supply voltage	–
V_{SS}	Ground	–

Source: Micron

SPI – Flash Devices

- **Some Vendors Use More Than One Data Line**
  - Example: Flash using Quad I/O for Higher Bandwidth

FIGURE 1: QUAD I/O SERIAL INTERCONNECTION

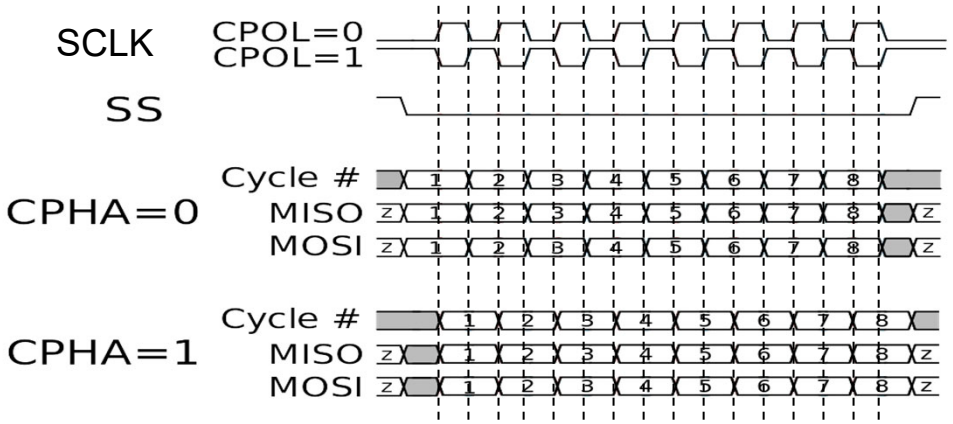


Quad Mode Operation
IO0 - I03: Bidirectional input and output data signals between Master and Slaves
SCLK: Generated by the Master to synchronize data transfers

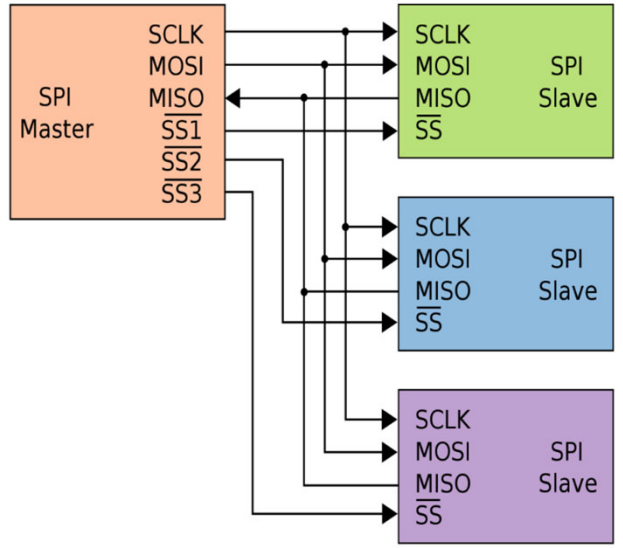
Source: Micron

Conclusions

- **SPI**
  - Master/Slave
  - Synchronous full-duplex transmission (MOSI, MISO)
  - Selection of device through Slave Select ($\overline{SS}$)
  - No acknowledge, no error detection
  - Four modes → clock polarity and clock phase



Source: Wikipedia



Source: Wikipedia