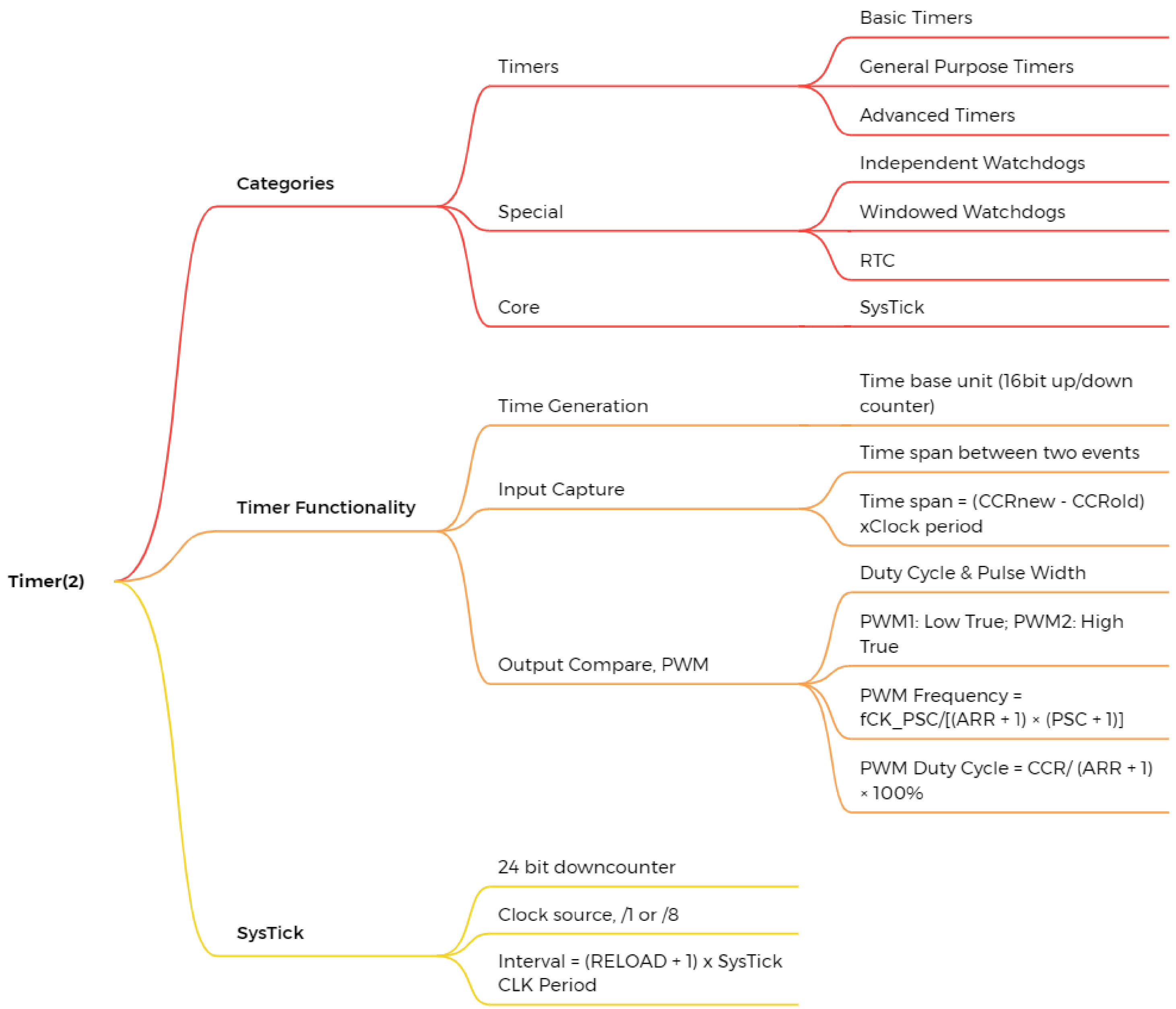
CS301
Embedded System and Microcomputer Principle

Lecture 10: I^{2}C and SPI

2024 Fall

baiyh@sustech.edu.cn

Recap



- **Timer(2)**
  - **Categories**
    - Timers
      - Basic Timers
      - General Purpose Timers
      - Advanced Timers
    - Special
      - Independent Watchdogs
      - Windowed Watchdogs
      - RTC
    - Core
      - SysTick
  - **Timer Functionality**
    - Time Generation
      - Time base unit (16bit up/down counter)
    - Input Capture
      - Time span between two events
      - Time span = (CCRnew - CCRold) xClock period
    - Output Compare, PWM
      - Duty Cycle & Pulse Width
      - PWM1: Low True; PWM2: High True
      - PWM Frequency = fCK_PSC/[(ARR + 1) × (PSC + 1)]
      - PWM Duty Cycle = CCR/ (ARR + 1) × 100%
  - **SysTick**
    - 24 bit downcounter
    - Clock source, /1 or /8
    - Interval = (RELOAD + 1) x SysTick CLK Period

Outline

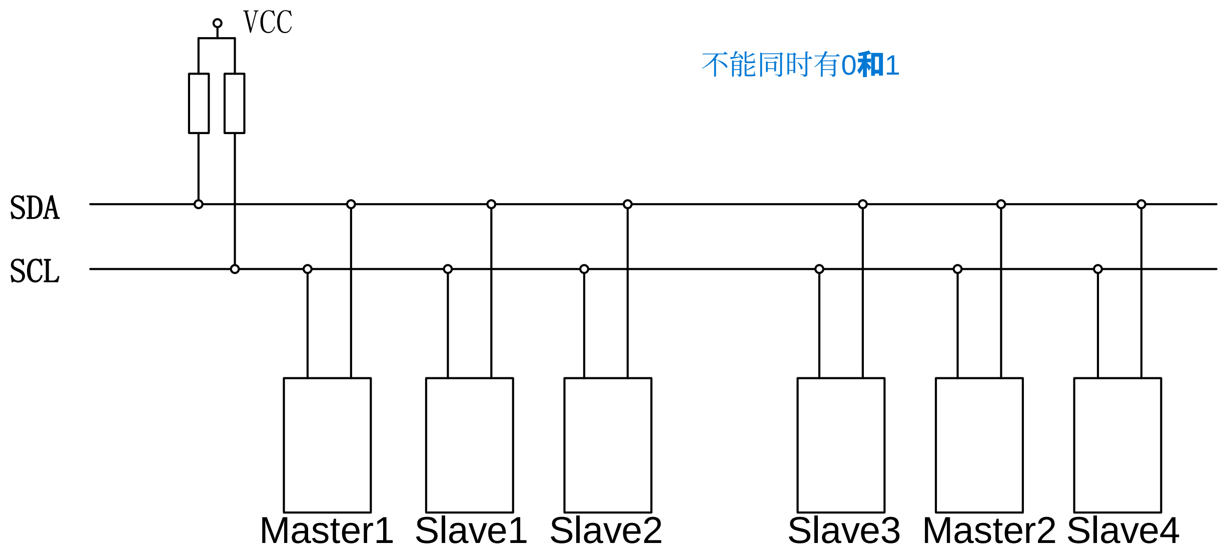
- I^2C
- SPI

Inter-Integrated Circuit (I^2C)

- Designed for low-cost, medium data rate applications
  - 2-wire communications
  - Synchronous, half-duplex
  - Start/Stop/Acknowledgment mechanism
- Characteristics
  - Serial, byte-oriented
  - Multi-master, multi-slave
  - Two bidirectional open-drain lines, plus ground
    - Serial Data Line (SDA)
    - Serial Clock Line (SCL)
  - Up to 100 kbit/s in the standard mode, up to 400 kbit/s in the fast mode

Inter-Integrated Circuit (I²C)

- I2C lines can have two possible states: Float high and Drive low
  - Pull-up resistor on the line, devices can pull the line low
  - If no device is pulling on the line, it will float high
- Multi-master bus with master and slave devices.
  - Collision detection and arbitration prevent errors with multiple active masters.



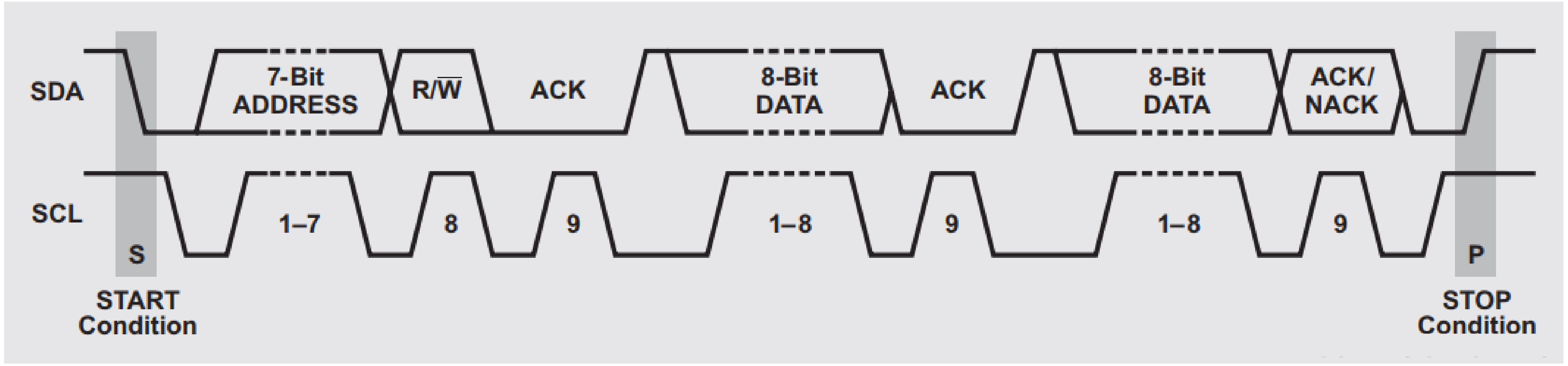
不能同时有0**和**1

Communication Steps

1. Master sends start condition **每次的通信由**master发起，**由**slave结束
2. Master sends address and direction **从哪个**slave read or write
3. Slave sends ack to response Master
4. Transmitter sends a byte of data
5. Receiver sends Acknowledge

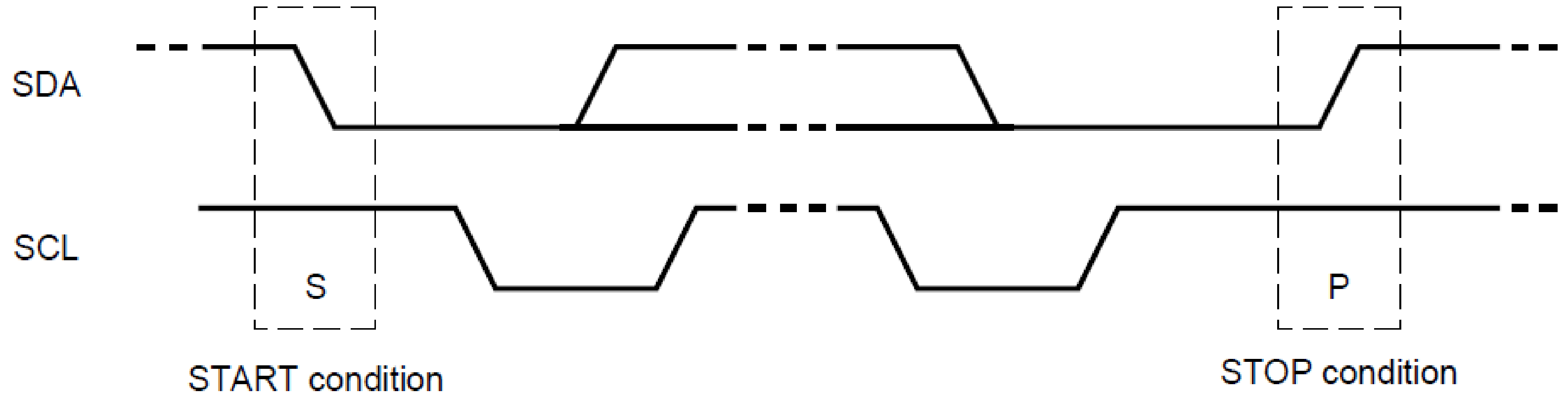
… Repeate step 4,5

n. Master sends stop condition



Start/Stop Signals

- START Condition (S)
  - SDA 1→0 transition when SCL = 1
- STOP Condition (S)
  - SDA 0→1 transition when SCL = 1
- Repeated Start(Sr)
- Bus state
  - Busy – after S and before next P
  - Free after P and before next S

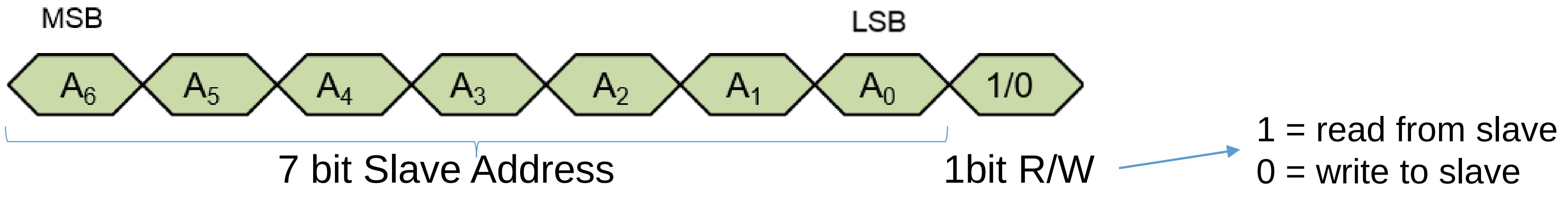


Data Frame

- Data bits are transferred after start condition
- Transmission is byte oriented (Frame)
  - 8 bits + 1 acknowledge bit
- Most significant bit first
- Address of the slave is also data
  - First byte is address
  - During first byte transfer
    - Master is the transmitter
    - Addressed slave is receiver
  - Next bytes: depend on the last bit in the address byte

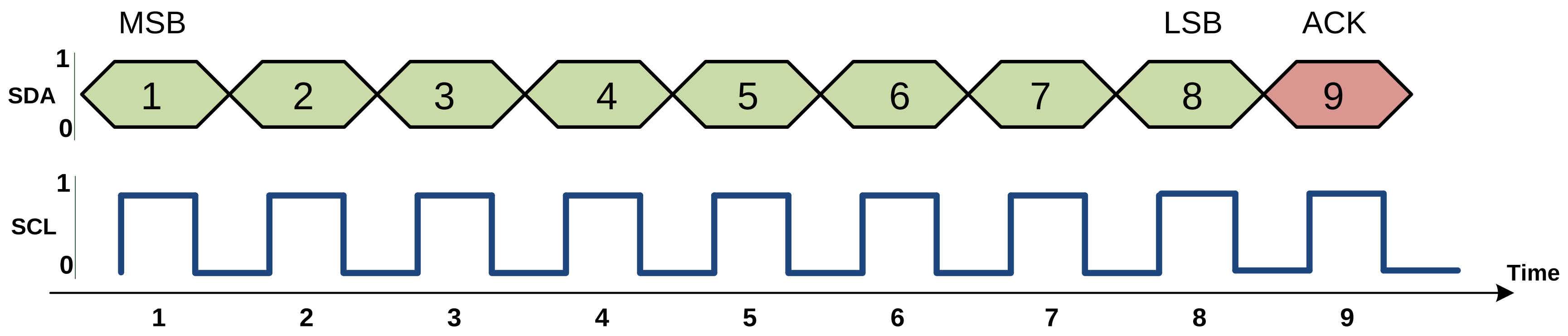
Addressing Scheme

- First byte transmitted by master
  - 7-bit: address
  - 1 bit: direction (R/W)
    - 0 master writes data
    - 1 master receives data
- Master may generate repeated start and address another device
- Each device listens to address
  - If address matches device switches state according to R/W bit



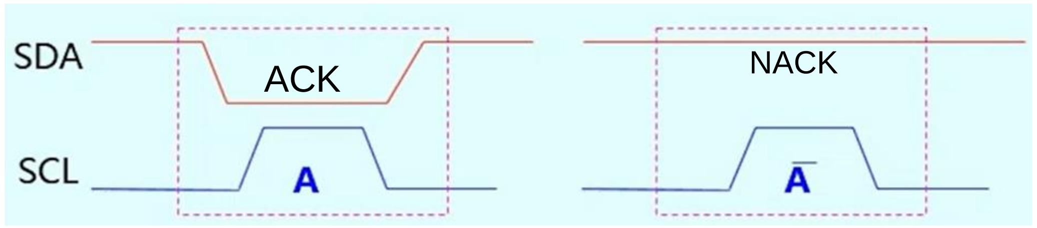
Transmission Scheme

- Each frame is 9 bits long.
- First 8 bits are put on SDA by the transmitter, The bytes are sent most significant bit first.
- The 9th bit is an acknowledge by the receiver
  - ACK (pull down) or NACK (leave high)



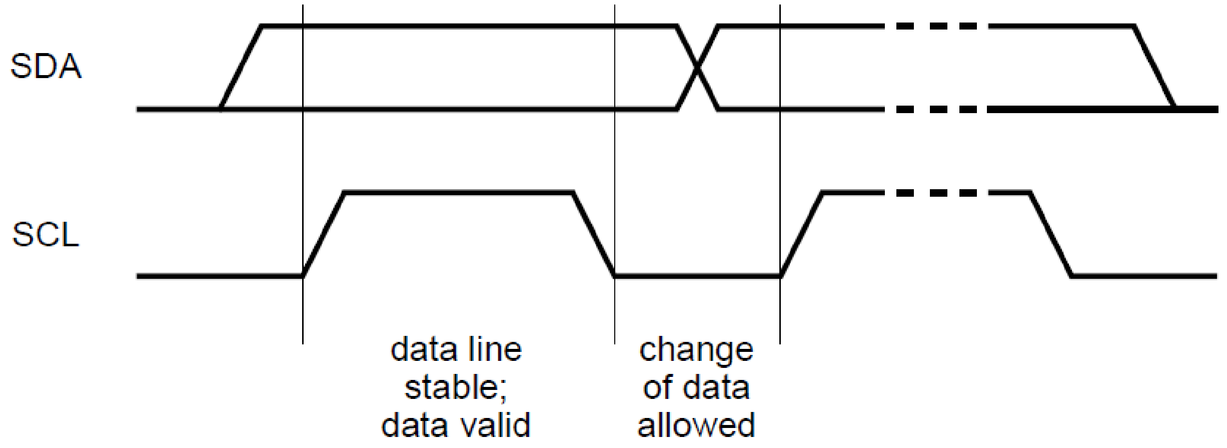
🟩 Sent by sender

🟥 Sent by receiver



Signal Synchronization

- When I^{2}C bus is transmitting data
  - When SCL = 0, transmitter sends a bit of data onto the data line, SDA is allowed to change
  - When SCL = 1, receiver reads a bit of data from the data line. SDA must remain stable



Master vs. Slave

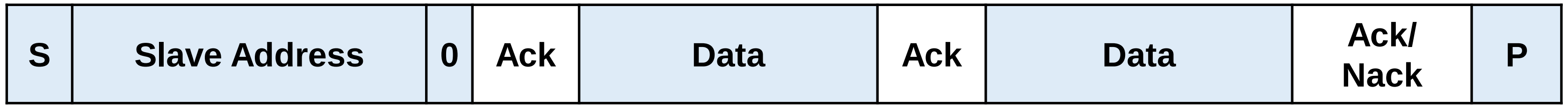
- Each device can be both Master and Slave
- Master
  - Begins the communication
  - Chooses the slave
  - Makes clock
  - Sends or receives data
- Slave
  - Responds to the master
  - Each slave has a unique 7-bit address

Working Modes

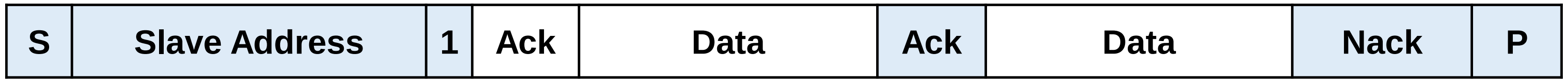
- Master-sender
  - Master issues START and ADDRESS, and then transmits data to the addressed slave device
- Master-receiver
  - Master issues START and ADDRESS, and receives data from the addressed slave device
- Slave-sender
  - Master issues START and the ADDRESS of the slave, and then the slave sends data to the master
- Slave-receiver
  - Master issues START and the ADDRESS of the slave, and then the slave receives data from the master.

Typical I^{2}C Timing Diagram

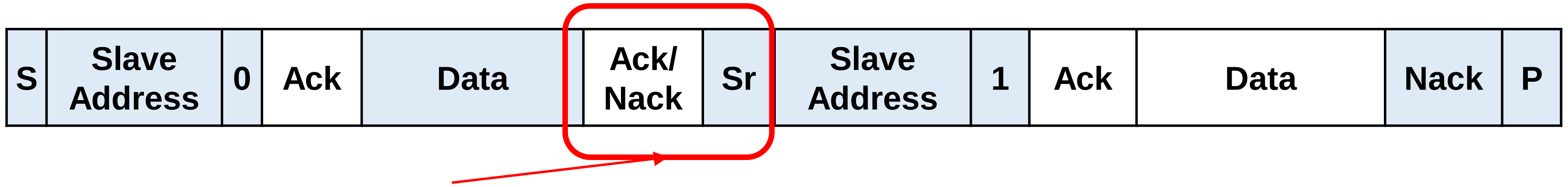
- Master sends data to Slave



- Slave sends data to Master



- Master sends data to Slave first, then Slave sends data to Master (Repeated Start)

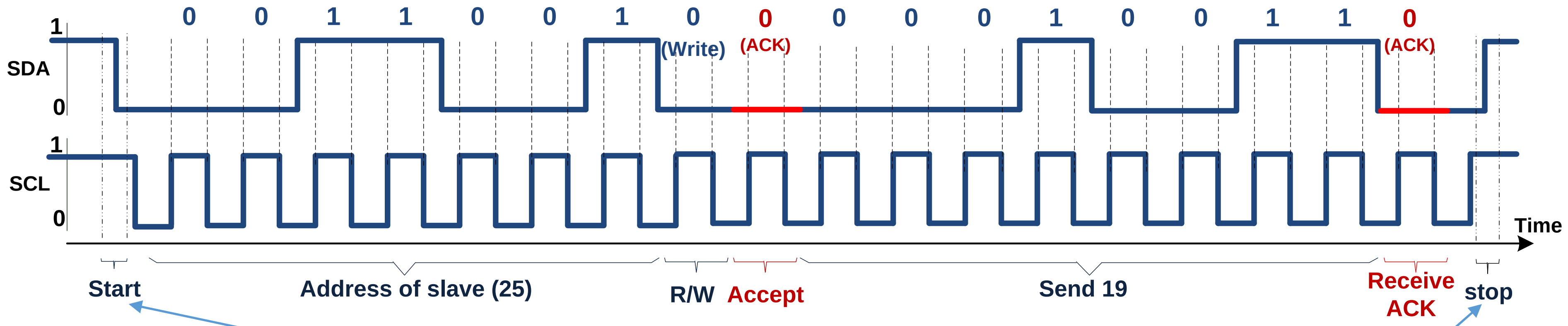


No Stop to avoid bus being taken control by other master device

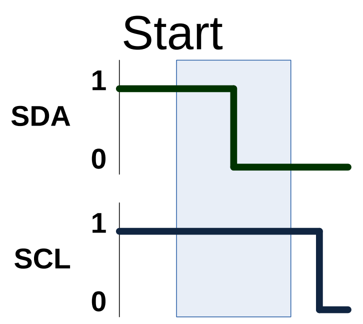
The blue blocks indicates data transmitted from the master to the slave, while the white blocks represents data transmitted from the slave to the master.

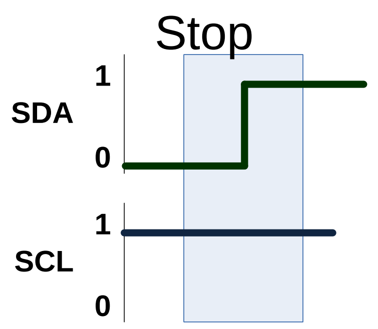
Sending Example

- Show how a master sends 19 to device 25.



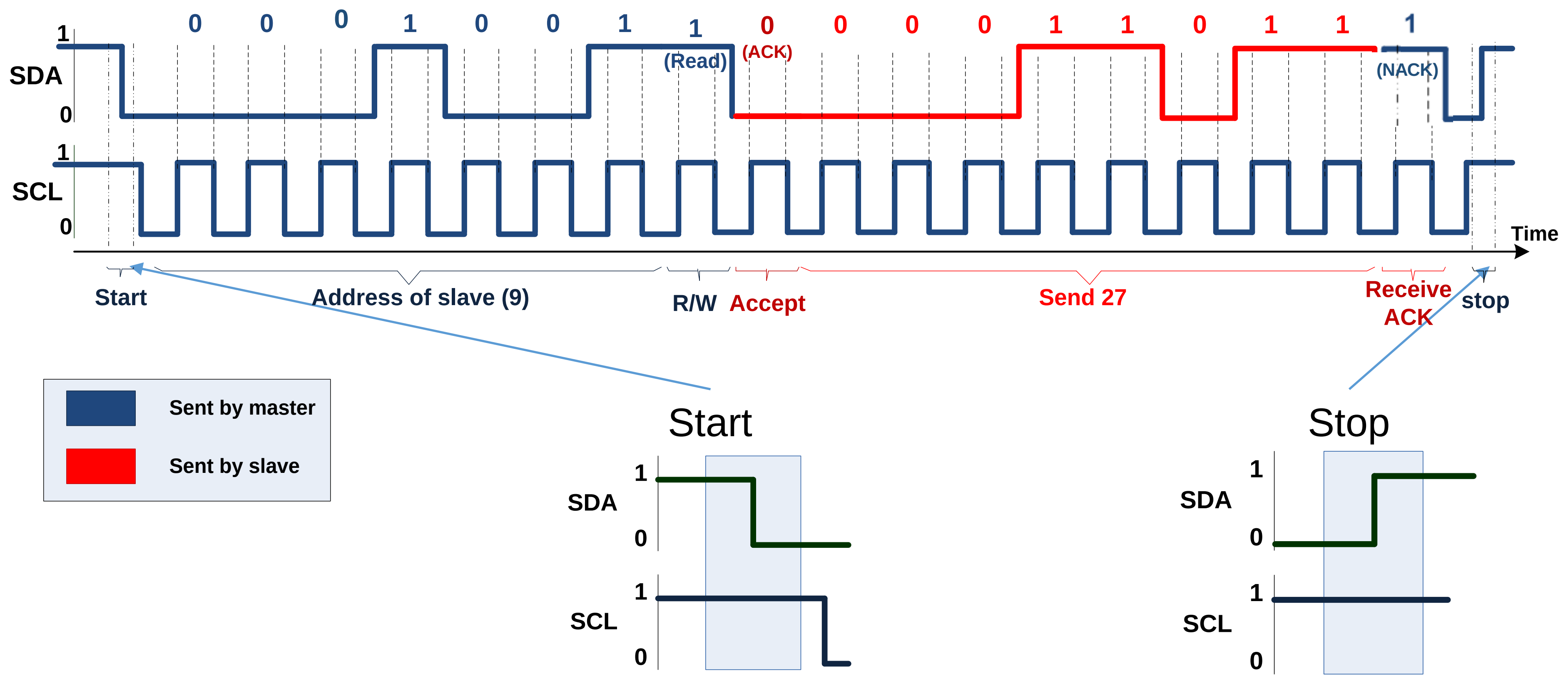
🟦	Sent by master
🟥	Sent by slave





Receiving Example

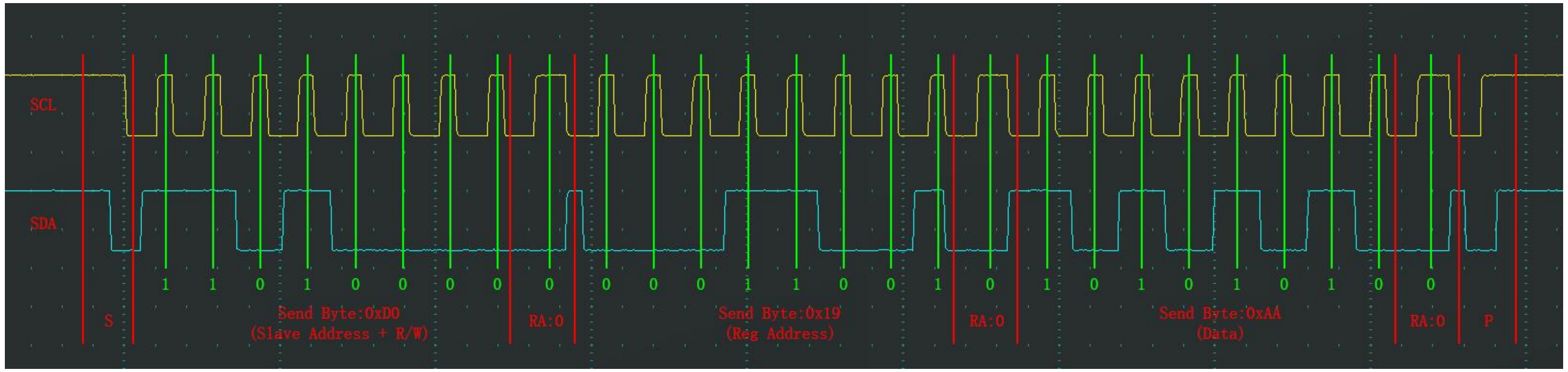
- Show how a Master receives 27 from Device 9



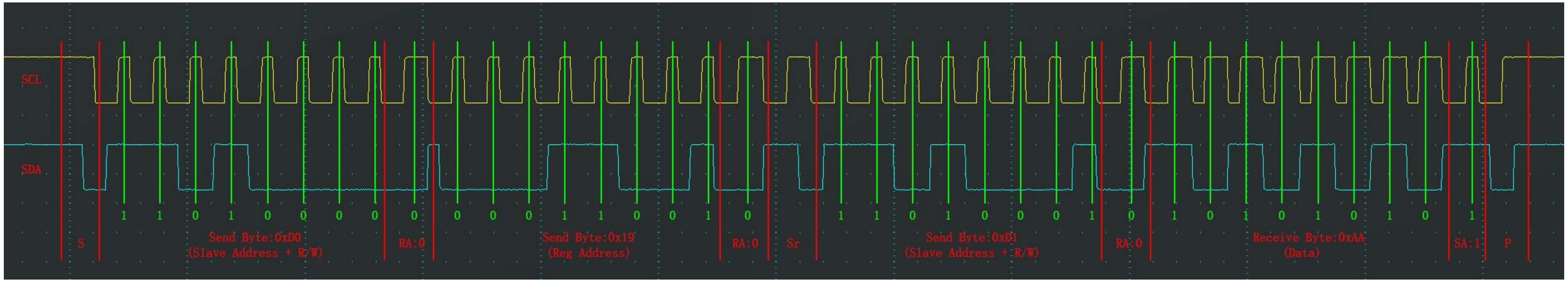
🟦	Sent by master
🟥	Sent by slave

Waveform captured from logic analyzer

- Master sends a byte to specific address of a specific slave device



- Master reads a byte from the specific address of a specific slave device

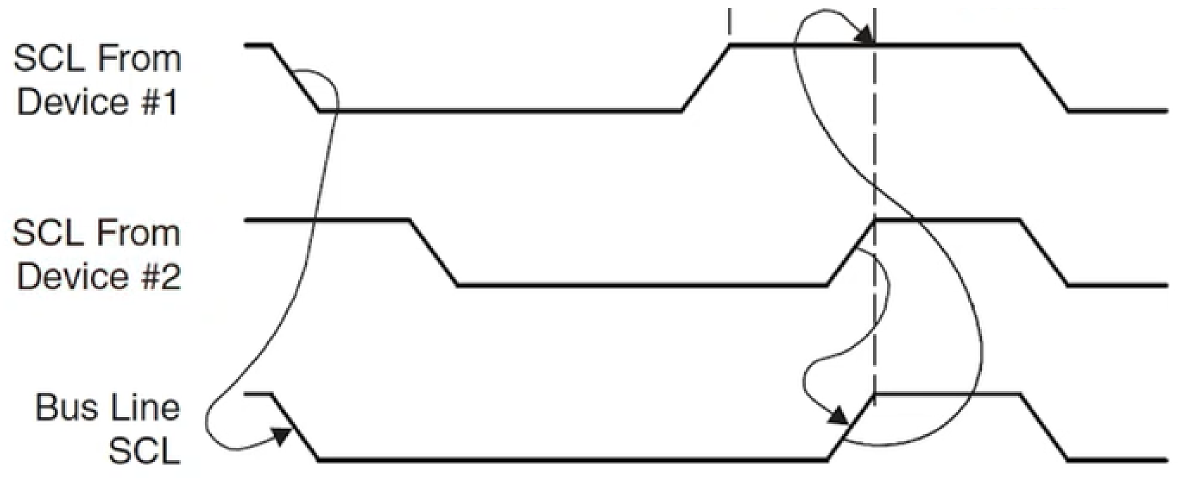


Bus Arbitration

- I2C designed as a multi-master bus
  - Any one of several different devices may act as the master at various times
  - No global master to generate clock: A master drives both SCL and SDL
- When two devices try to drive SDL to different values
  - “Wired-AND” bus. Listen to the bus to be sure that is not interfering with another message
  - If the device is trying to send a logic 1 but hears logic 0
  - It immediately stop transmission and gives the other sender priority

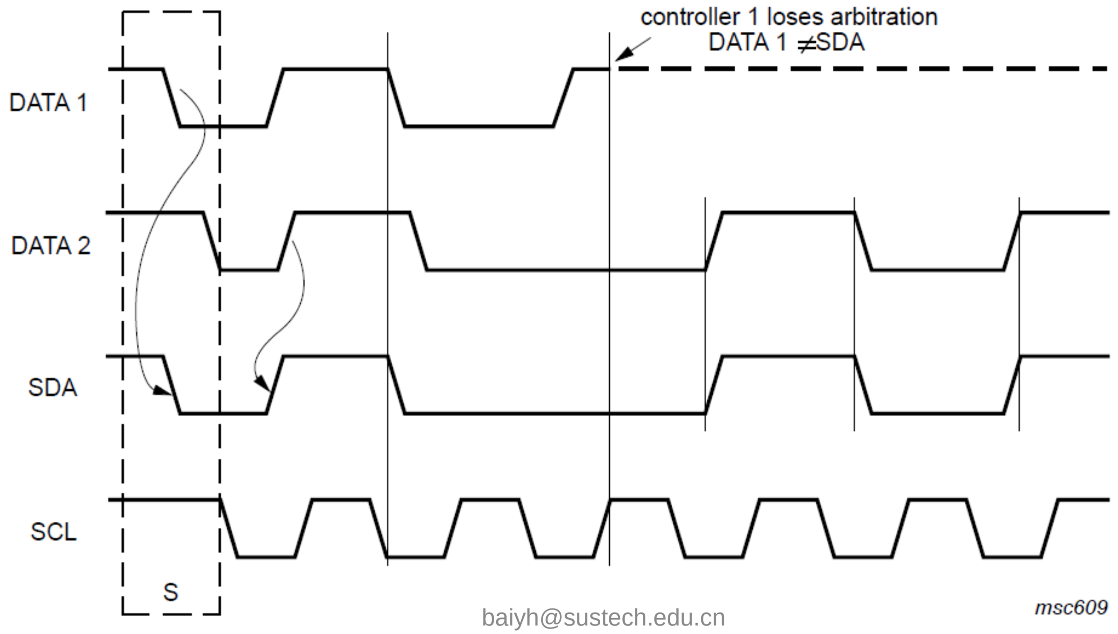
Multiple Masters: Clock

- “Wired-AND” bus: A sender can pull the lines to low, even if other senders are trying to drive the lines to high

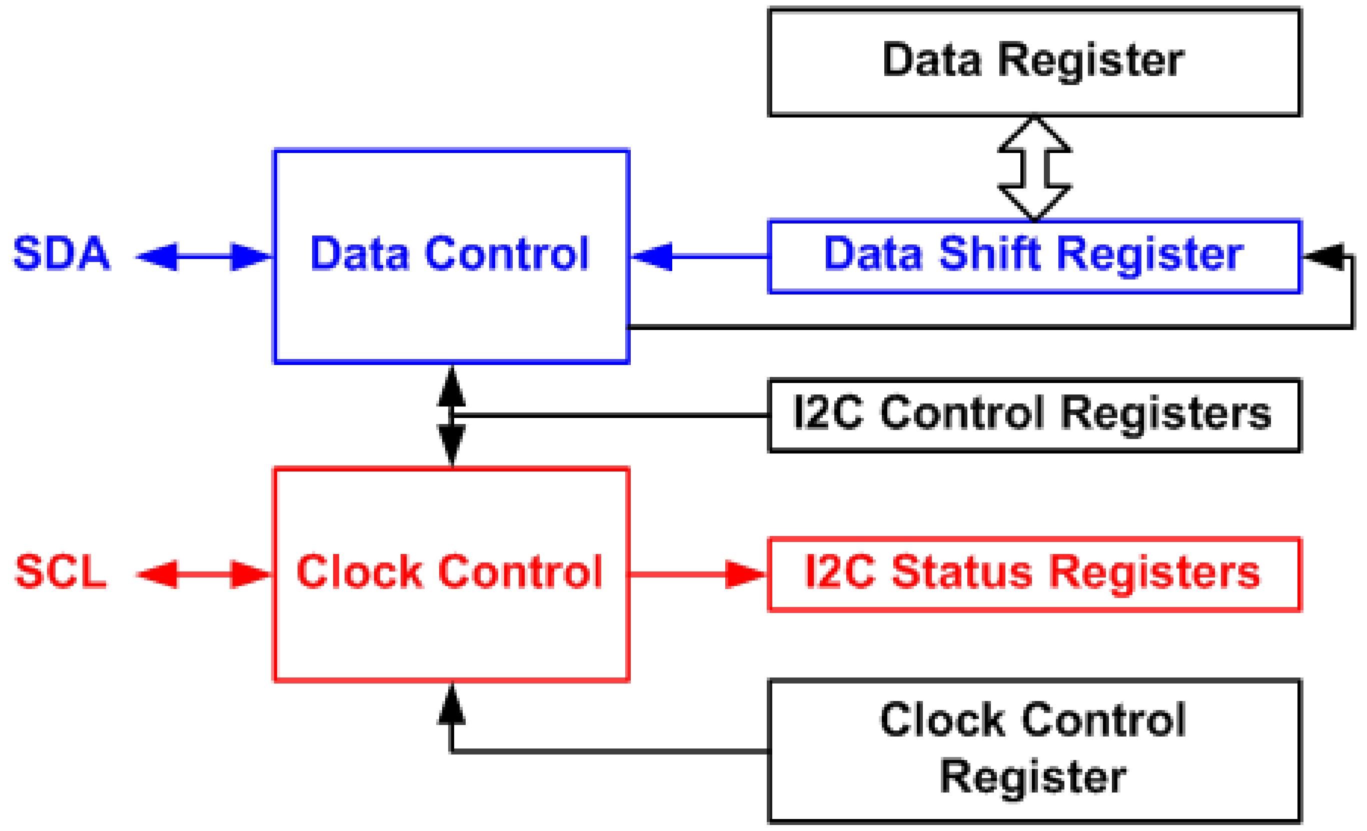


Multiple Masters: Data

- Arbitration for multiple masters:
  - During data transfer, the master constantly checks whether the SDA voltage level matches what it has sent.
  - When two masters generate a START setting concurrently, the first master which detects SDA low while it has intended to set SDA high will lose the arbitration and let the other master complete the data transfer.

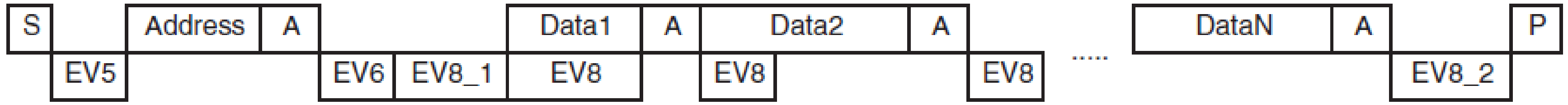


STM32 I2C Module



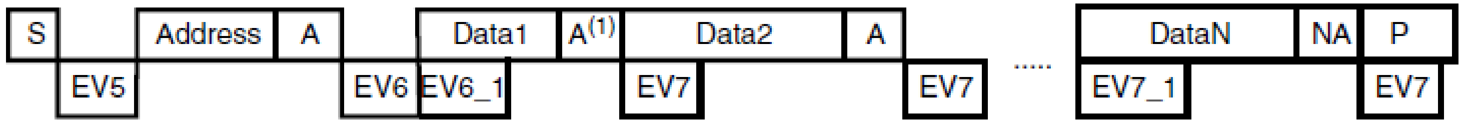
STM32 I2C Transfer Sequence Diagram

7-bit master transmitter



EV5: SB=1, cleared by reading SR1 register followed by writing DR register with Address.
EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2.
EV8_1: TxE=1, shift register empty, data register empty, write Data1 in DR.
EV8: TxE=1, shift register not empty,.data register empty, cleared by writing DR register
EV8_2: TxE=1, BTF = 1, Program Stop request. TxE and BTF are cleared by hardware by the Stop condition
EV9: ADD10=1, cleared by reading SR1 register followed by writing DR register.

7-bit master receiver



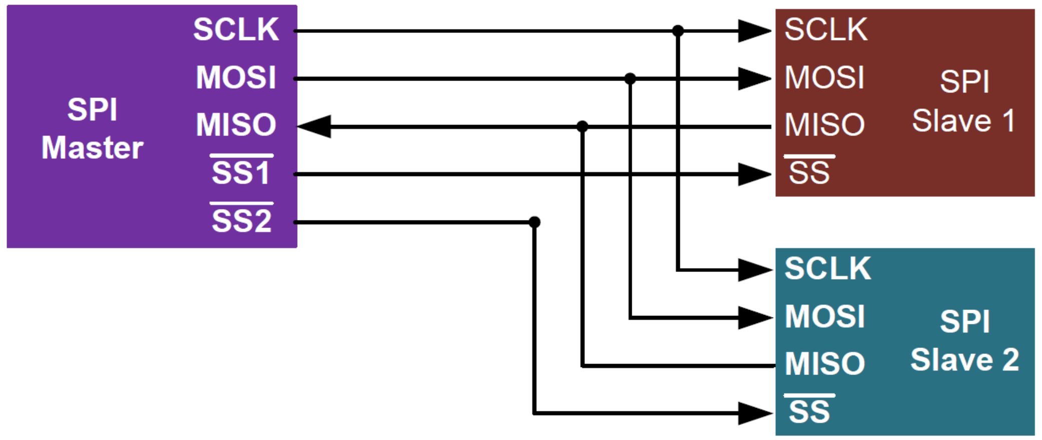
EV5: SB=1, cleared by reading SR1 register followed by writing DR register.
EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2. In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.
EV6_1: no associated flag event, used for 1 byte reception only. The Acknowledge disable and Stop condition generation are made just after EV6, that is after ADDR is cleared.
EV7: RxNE=1 cleared by reading DR register.
EV7_1: RxNE=1 cleared by reading DR register, program ACK=0 and STOP request
EV9: ADD10=1, cleared by reading SR1 register followed by writing DR register.

Outline

- I^{2}C
- SPI

Serial Peripheral Interface (SPI)

- Synchronous full-duplex communication
- Single master, multiple slaves
- No Start/Stop or slave acknowledgment
- Master sets corresponding SS signal to communicate with slave device
- More than 10 Mbit/s



SCLK: serial clock

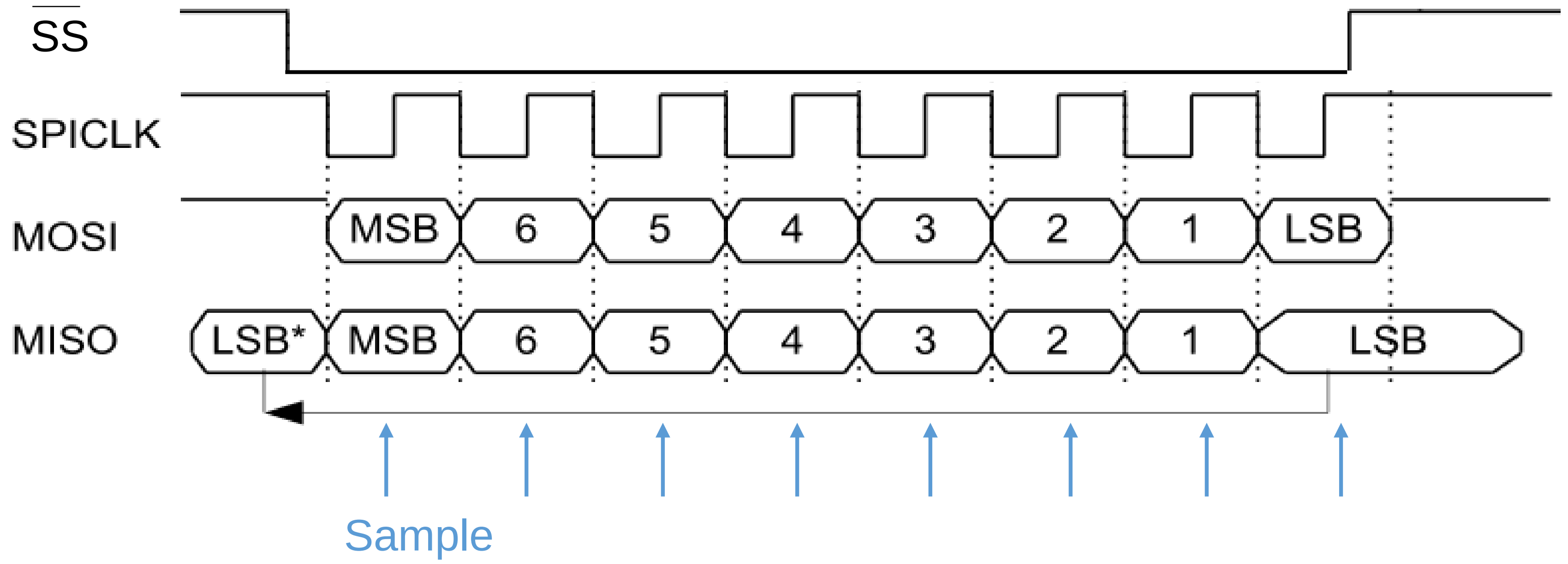
SS: slave select

MOSI: master out slave in

MISO: master in slave out

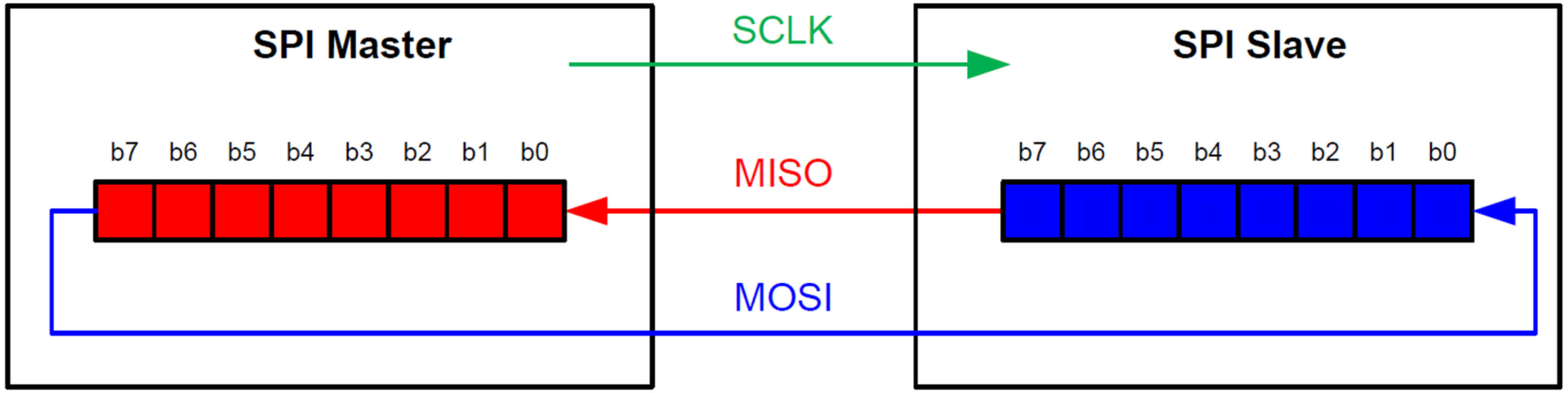
SPI Timing Diagram

- During data transmission, MSbit is sent first (can be programmed to LSbit first)
- No acknowledge bit
- Data is transmit at rising edge or falling edge of SPICLK, the same bit data is read at the following falling edge or rising edge of SPICLK



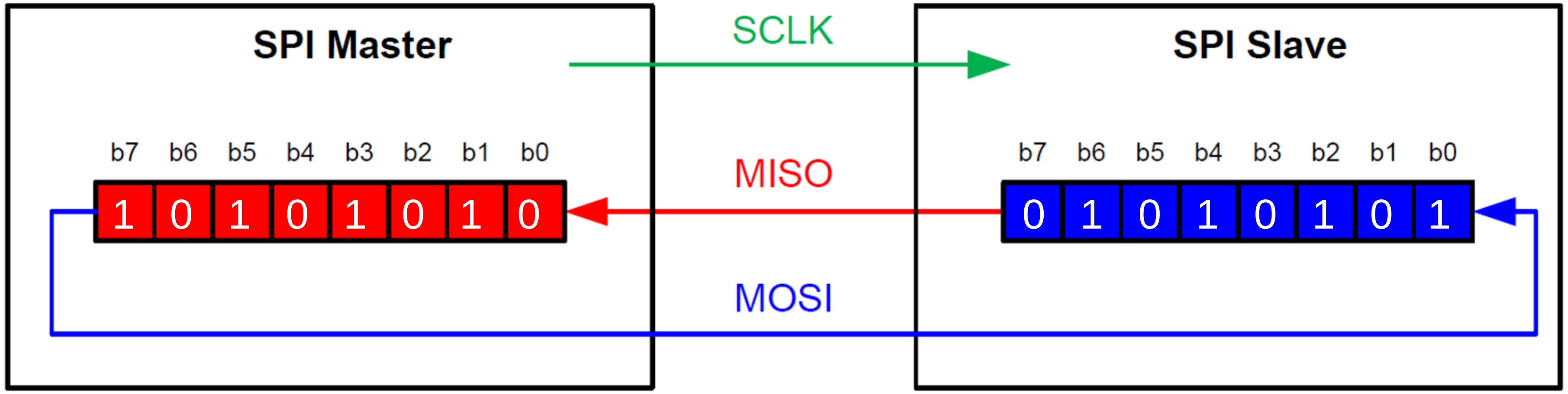
SPI Synchronous Data Exchange

- Master has to provide clock to slave
- Synchronous exchange
  - Master shift out a bit to slave, and shifts in a bit from slave.
- Only master can start the data transfer.



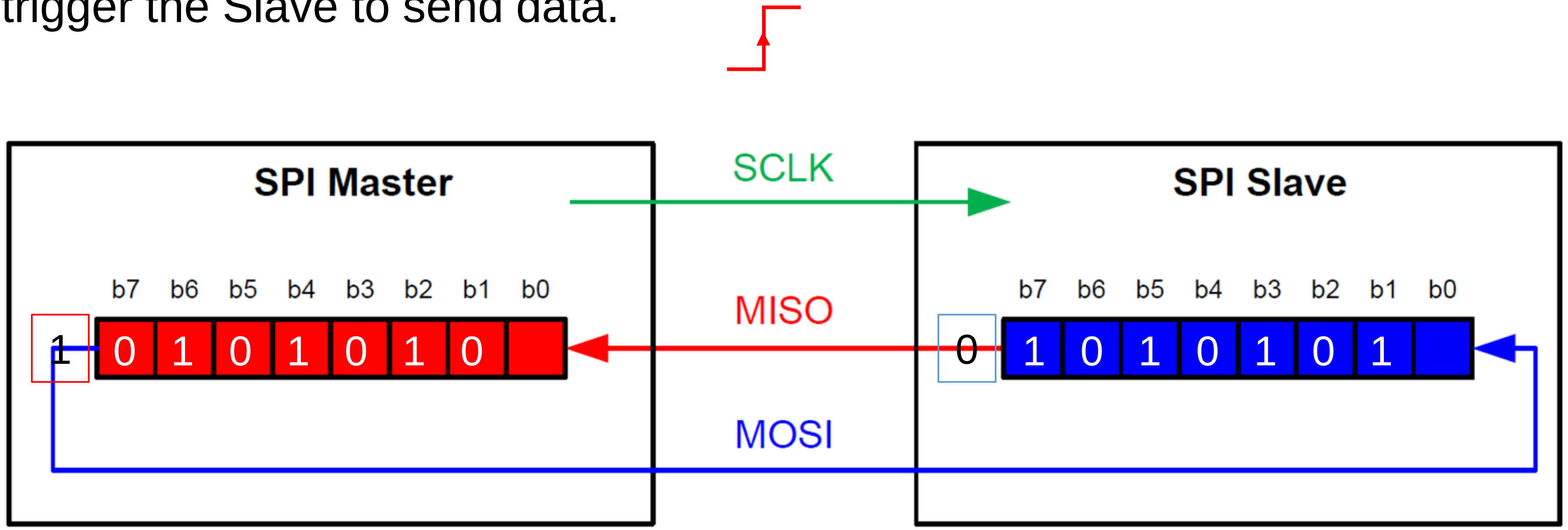
Data Exchange Example (1)

- If the Master only performs write operations to the Slave, it can ignore the data received from the Slave.
- If the Master needs to perform a read operation from the Slave, it should send an empty data to trigger the Slave to send data.



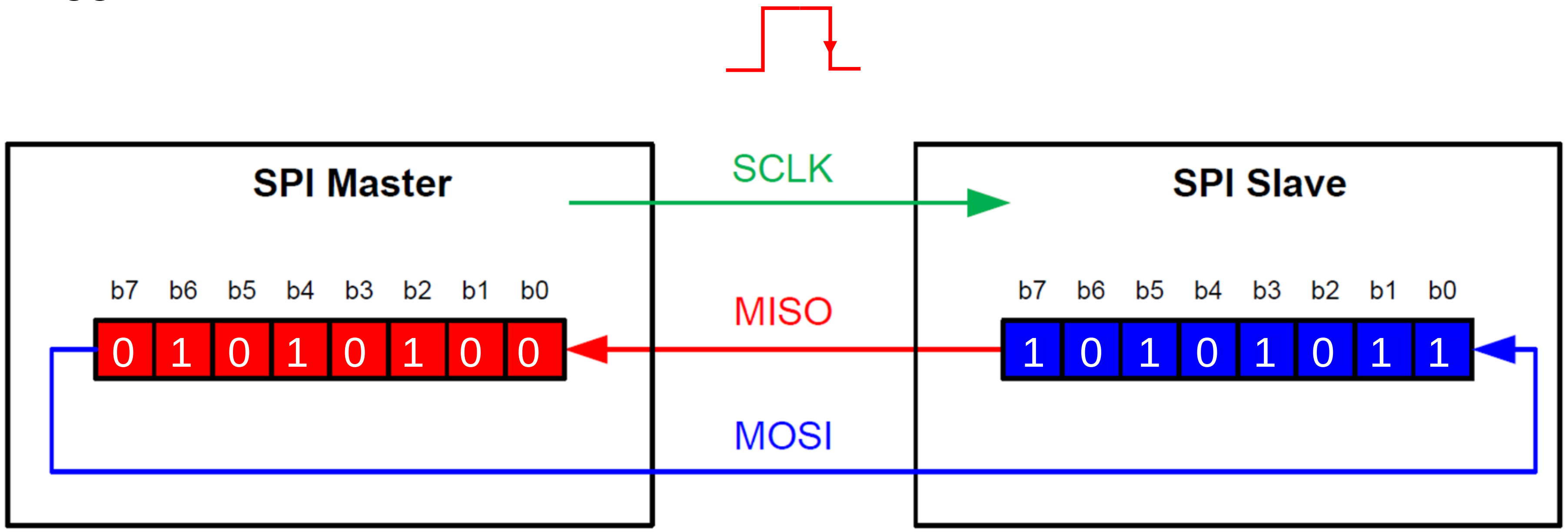
Data Exchange Example (1)

- If the Master only performs write operations to the Slave, it can ignore the data received from the Slave.
- If the Master needs to perform a read operation from the Slave, it should send an empty data to trigger the Slave to send data.

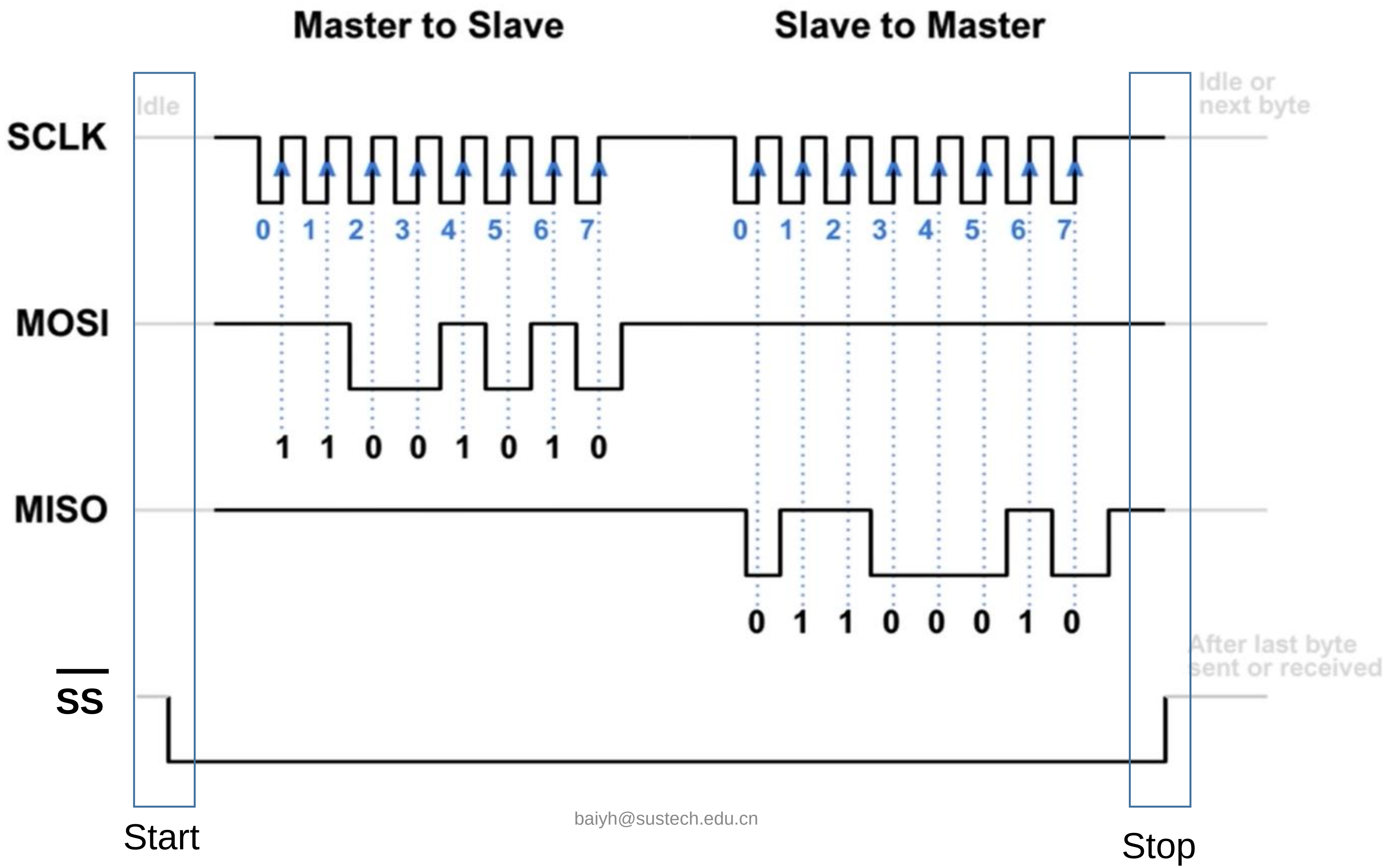


Data Exchange Example (3)

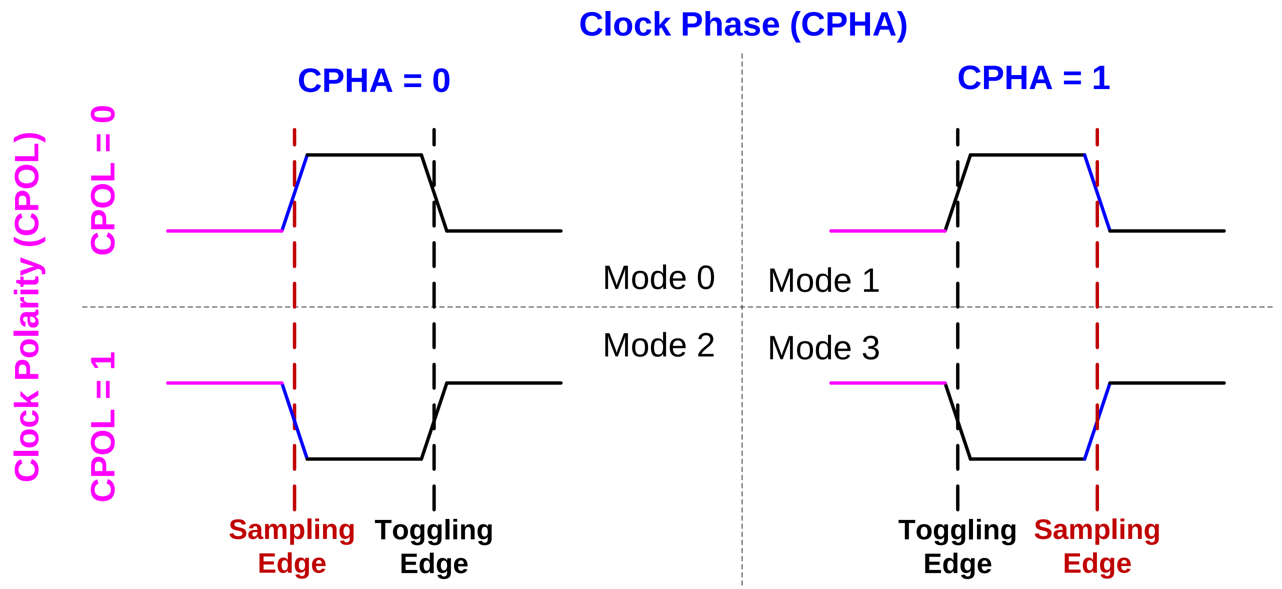
- If the Master only performs write operations to the Slave, it can ignore the data received from the Slave.
- If the Master needs to perform a read operation from the Slave, it should send an empty data to trigger the Slave to send data.



SPI Timing Diagram Example



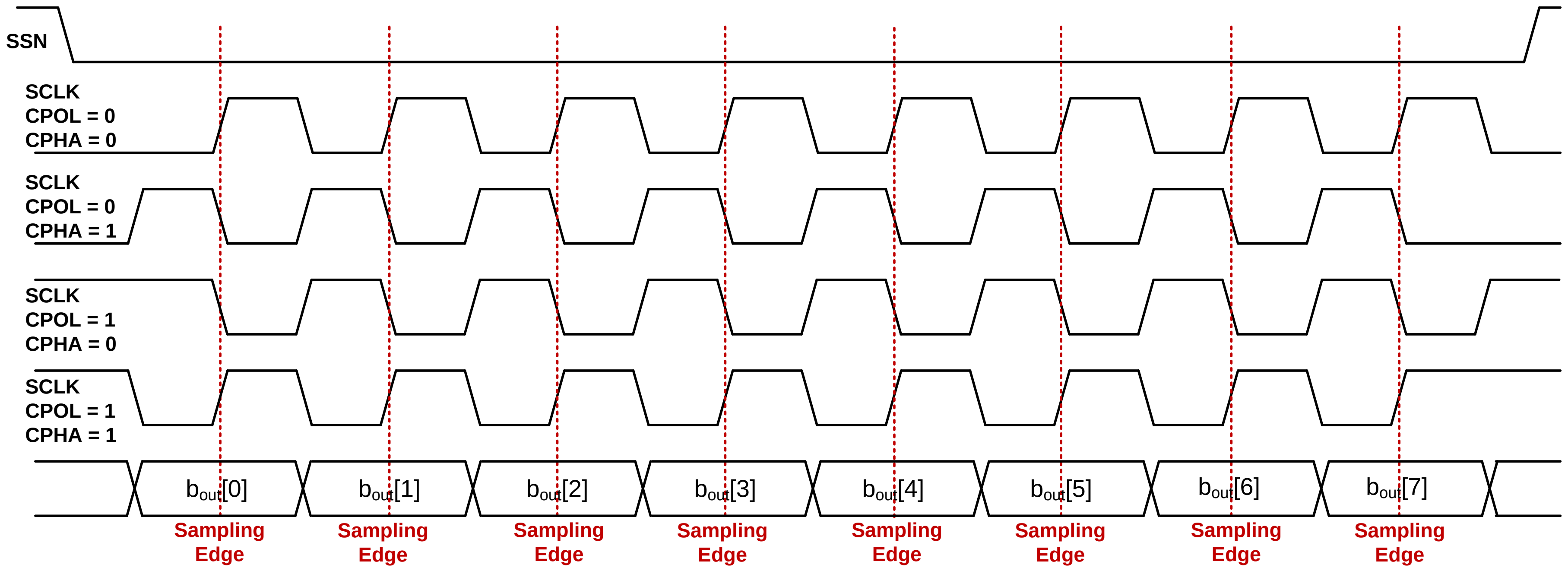
SPI Clock Phase and Polarity



- Combination of CPOL(极性) and CPHA(相位) determines the clock edge for transmitting and receiving.
- CPOL = 0 ⟶ SCLK is pushed to low during idle. Otherwise, pulled to high during idle.
- CPHA = 0 ⟶ the first clock transition (either rising or falling) is the first data capture edge. Otherwise, the second clock transition is the first data capture edge.

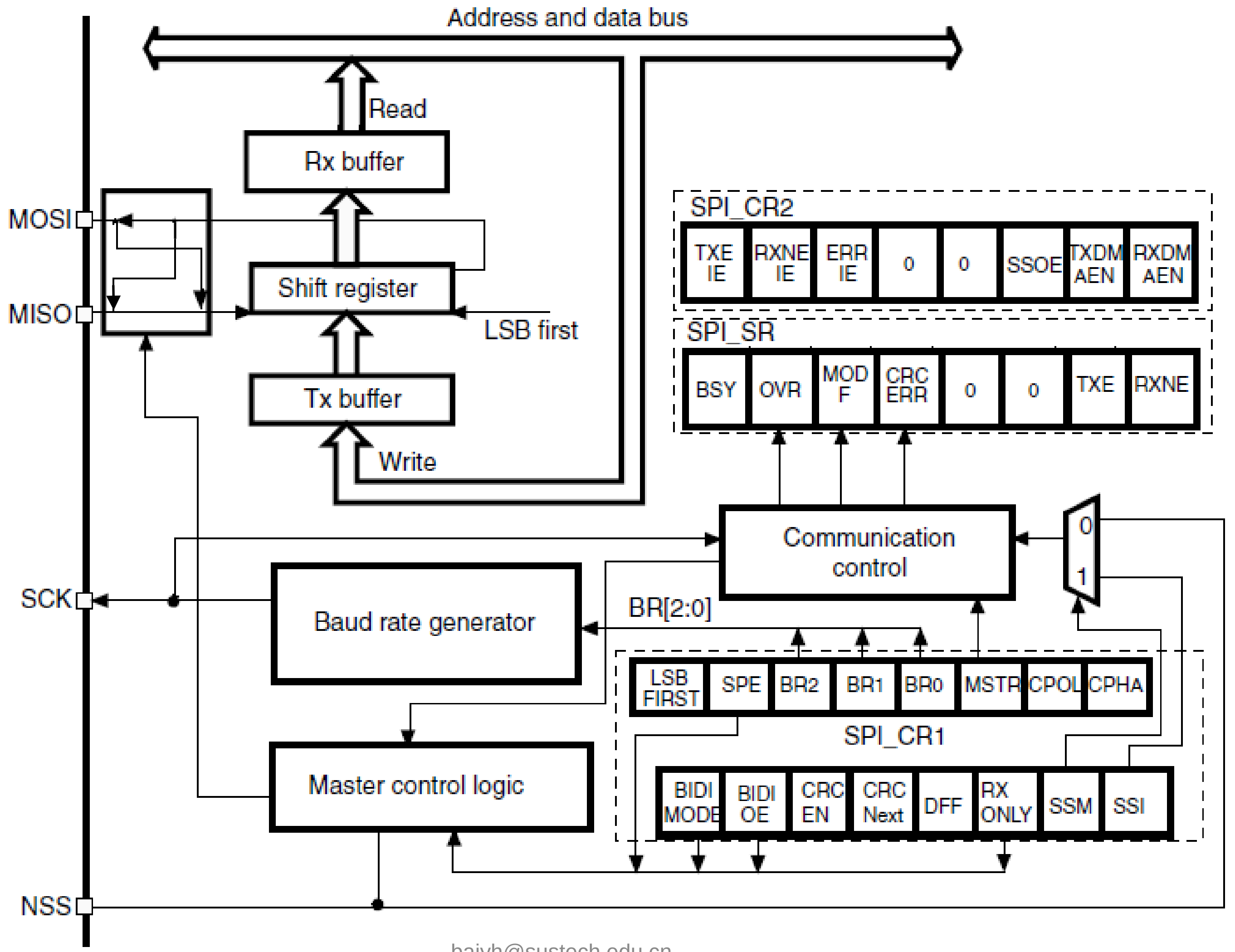
SPI Clock Phase and Polarity

在片选信号拉下去以后，时钟的电平



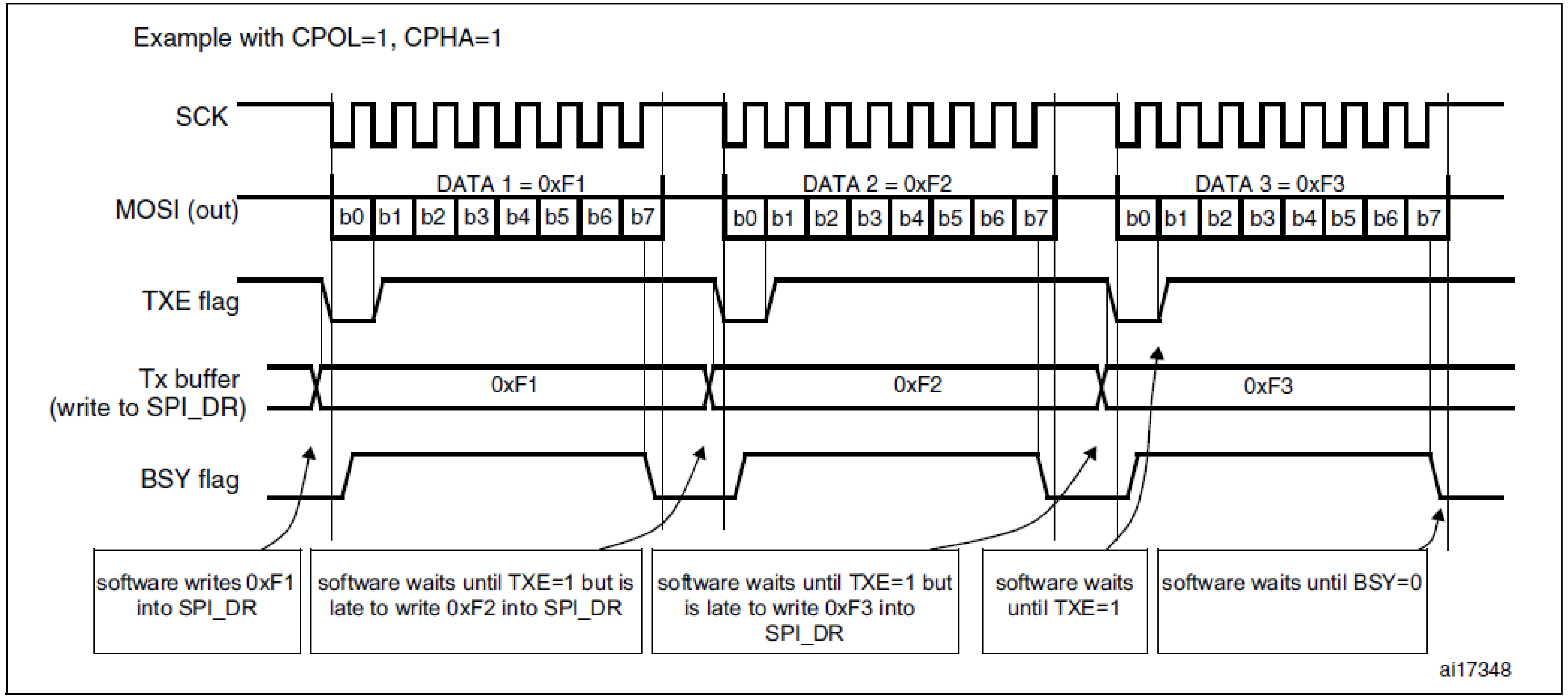
CPOL	CPHA	Data Read and Change Time	SPI Mode
0	0	Read on rising edge, changed on a falling edge	0
0	1	Read on falling edge, changed on a rising edge	1
1	0	Read on falling edge, changed on a rising edge	2
1	1	Read on rising edge, changed on a falling edge	3

STM32 SPI block diagram



Transmission sequence

- CPOL=1,CPHA=1:Read on rising edge, changed on a falling edge



Example with CPOL=1, CPHA=1
SCK
MOSI (out)
DATA 1 = 0xF1
DATA 2 = 0xF2
DATA 3 = 0xF3
b0 b1 b2 b3 b4 b5 b6 b7
TXE flag
Tx buffer (write to SPI_DR)
0xF1
0xF2
0xF3
BSY flag
software writes 0xF1 into SPI_DR
software waits until TXE=1 but is late to write 0xF2 into SPI_DR
software waits until TXE=1 but is late to write 0xF3 into SPI_DR
software waits until TXE=1
software waits until BSY=0
ai17348

SPI vs I2C

- Commonalities
  - Both utilize a serial and synchronous communication method.
  - Similar application scenarios and for short distance communications
  - Both operate in a master-slave configuration.

- Differences:
  - SPI by Motorola and I2C by Philips
  - I2C is half-duplex, while SPI is full-duplex.
  - I2C employs an acknowledgment mechanism, whereas SPI lacks such a mechanism.
  - I2C addresses devices by broadcasting slave addresses onto the bus, while SPI addresses devices by sending an chip select signal to the corresponding slave.
  - I2C has fixed clock polarity and clock phase, whereas SPI allows adjustable clock polarity and clock phase.