ARM Microcontroller Internship Programme

Trainer: *Kiran Jaybhave*

Sunbeam – Industrial Training Programme

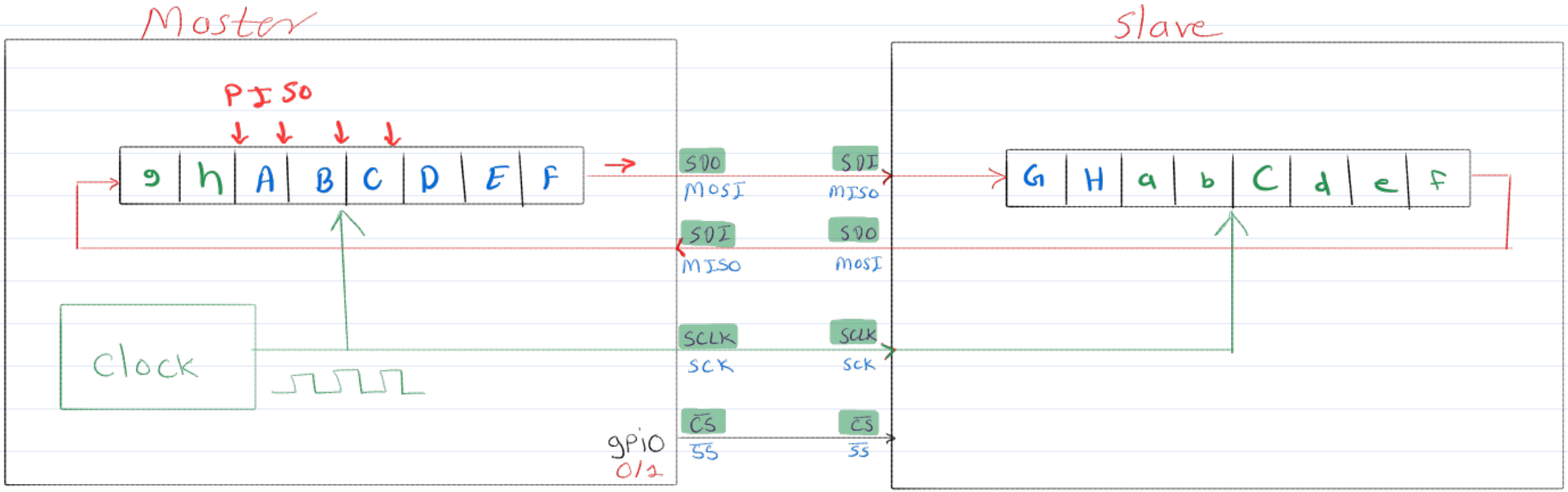
Sunbeam Pune (Hinjawadi)

SPI Protocol

- **SPI – Introduction (Serial Peripheral Interface)**
  - SPI (Serial Peripheral Interface) is a **high-speed, synchronous serial communication protocol** used for short-distance communication between a **master device** and one or more **slave devices**.
  - It was originally developed by **Motorola** and is widely used in embedded systems to connect peripherals such as **sensors, displays, memory chips, ADC/DACs**, and communication modules.
  - SPI is known for its **simple hardware design**, **fast data transfer**, and **full-duplex communication**, making it ideal for applications where **speed is more important than pin count**.
  - SPI is faster than I^2C and UART, but it uses more wires.

Physical Characteristics of SPI (Serial Peripheral Interface)

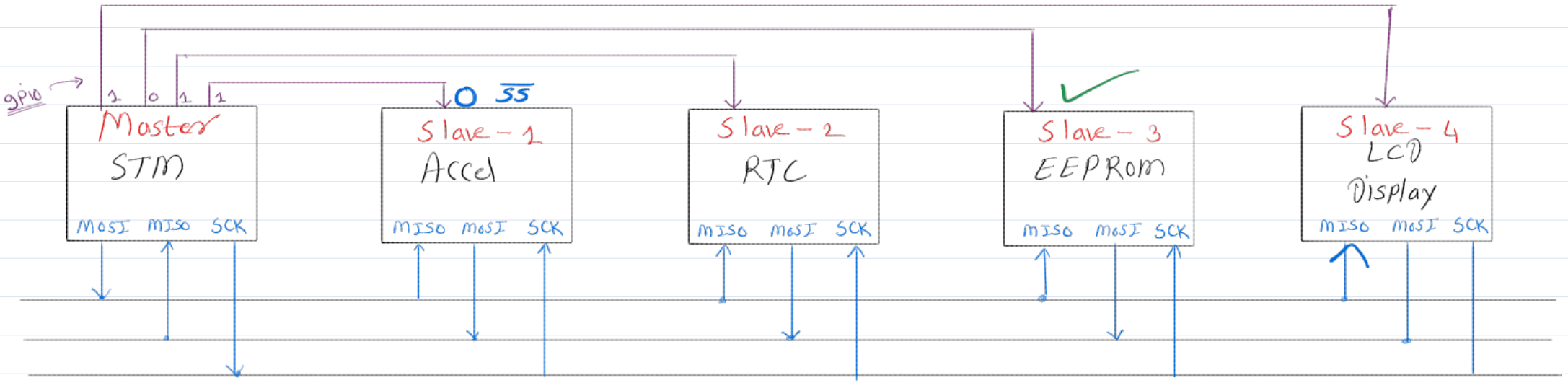
- **Type of Protocol**
  - **SPI is a synchronous serial communication protocol**
  - Communication is controlled by a **clock signal generated by the Master**
  - Data transfer occurs **bit-by-bit** in synchronization with the clock.
- **Communication Mode**
  - **Full-Duplex Protocol**
    - Data transmission and reception happen **simultaneously**
    - Master can send data to Slave **while** receiving data from Slave at the same time.
    - Achieved using **separate data lines** (MOSI & MISO)
  - **Bus Protocol**
    - SPI is a **bus-based protocol**
    - One **Master** device can communicate with:
      - One Slave (single-slave configuration)
      - Multiple Slaves (multi-slave configuration)
    - Each Slave is selected using a **dedicated Slave Select (SS) line**


Master
PISO
g
h
A
B
C
D
E
F
SDO
MOSI
SDI
MISO
SDI
MISO
SDO
MOSI
SCLK
SCK
SCLK
SCK
CS
SS
CS
SS
gpio
0/1
clock
Slave
G
H
a
b
C
d
e
f

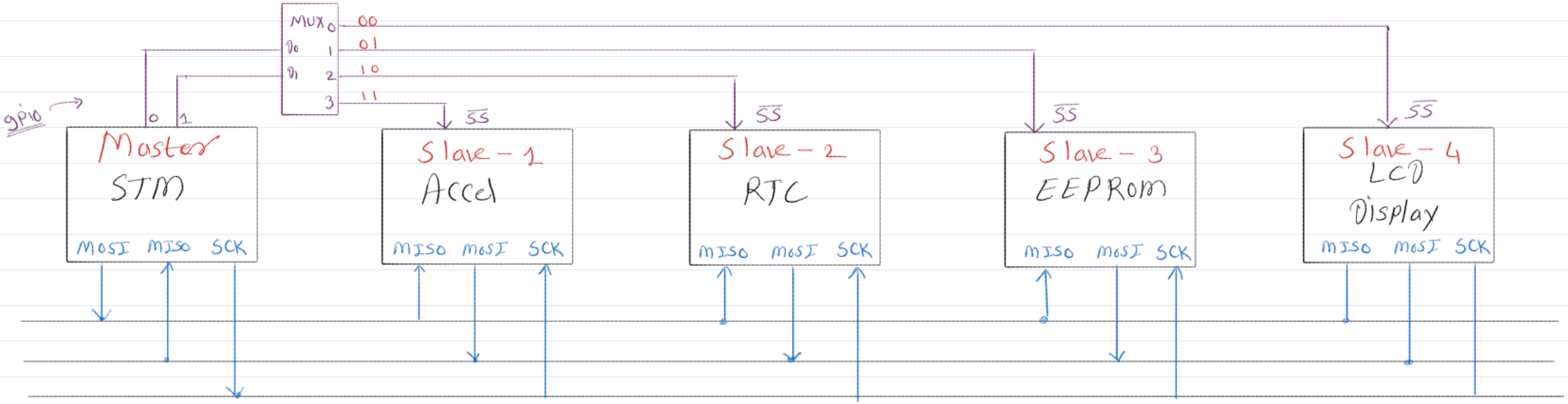
- **Wires / Connectivity (4-Wire Protocol)**
  - SPI typically uses **4 signal lines** for communication:

Signal Name	Alternate Name	Direction	Description
MOSI	SDO	Master → Slave	Data sent from Master to Slave
MISO	SDI	Slave → Master	Data sent from Slave to Master
SCLK / SCK	Clock	Master → Slave	Clock generated by Master
SS / CE	Chip Enable	Master → Slave	Selects the active Slave

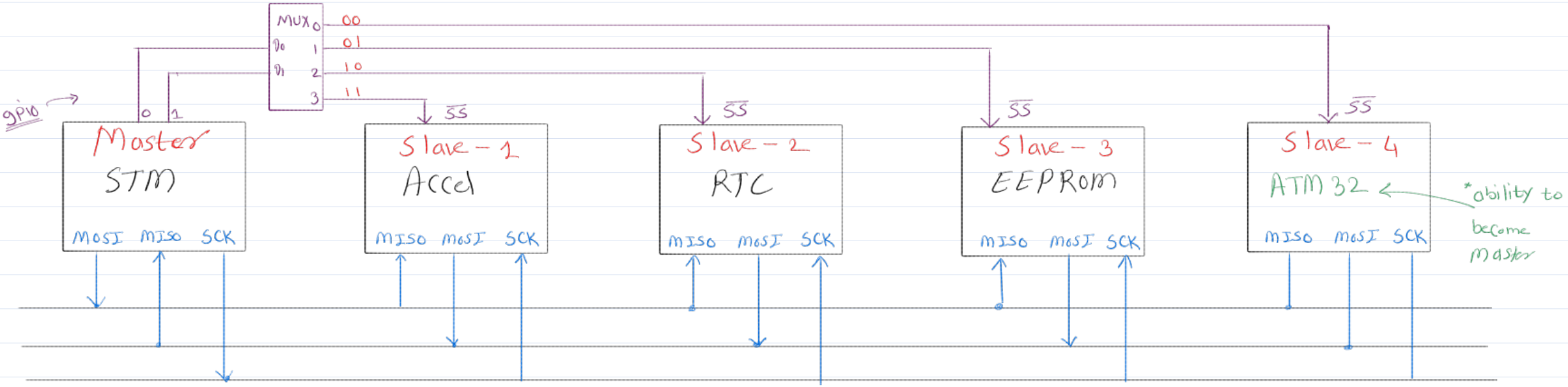
* General dig of SPI



* Using Mux we can Reduce GPIO Pins



* Multiple devices Capable to become slave.



* But master is one device at a time
* The if Any device Generating clock He is the Current Master.
* Multiple Master Not at Same time
* Multiple device becomes master but at a time one only. one.

- **Internal Block Diagram**
  - SPI uses **one shift register in Master and one in Slave**
  - Both shift registers are connected in a **circular fashion**
  - On each clock pulse:
    - One bit is shifted **out**
    - One bit is shifted **in**
  - This enables **true full-duplex communication**
  - **Same clock → simultaneous send & receive**
- **Voltage Levels (Logic Levels)**
  - SPI uses **TTL/CMOS voltage levels**
  - Logic levels depend on MCU supply voltage

Logic Level	Voltage
Logic 0	0 V
Logic 1	3.3 V or 5 V

  - STM32 typically uses **3.3 V logic**

- **Frequency / Speed**
  - SPI is a **high-speed protocol**
  - Clock frequency is configurable by the Master
  - Typical speeds:
    - Few MHz to **tens of MHz**
  - Faster than:
    - UART
    - I²C
  - **No start/stop bits → higher data throughput**

Logical Characteristics of SPI (Serial Peripheral Interface)

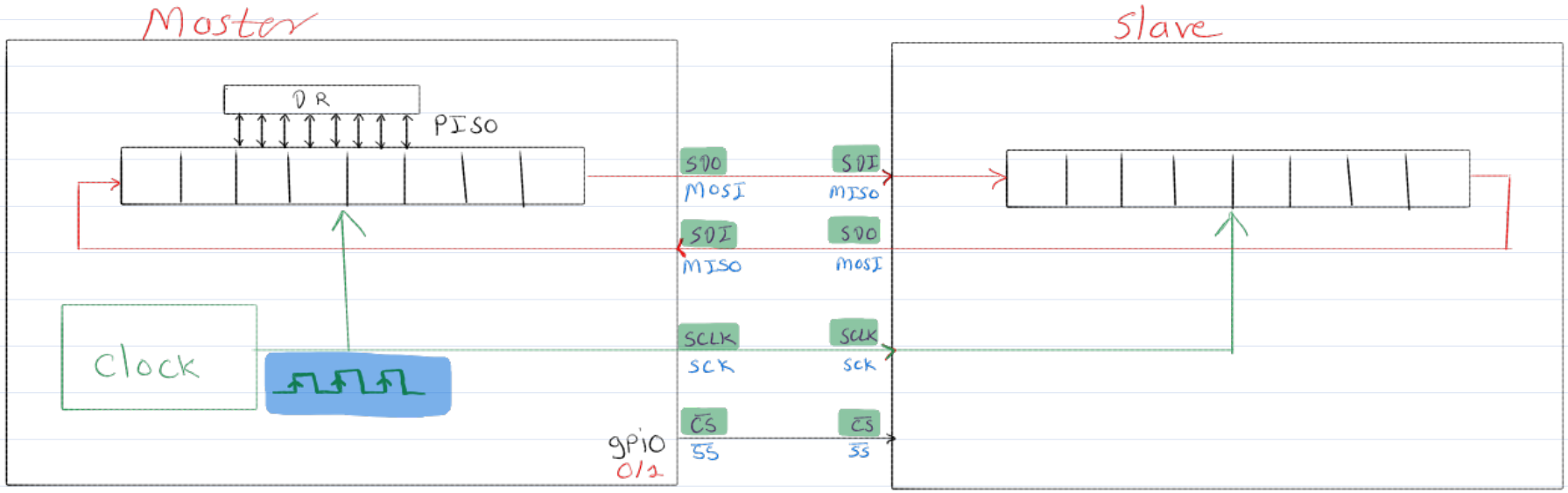
- **Data Bit Transfer**
  - SPI transfers data **bit-by-bit** synchronized with the **serial clock (SCLK)**
  - Data is shifted **MSB first or LSB first** (configurable)
  - On every clock pulse:
    - One bit is **transmitted**
    - One bit is **received**
  - Enables **simultaneous send and receive** (full-duplex)
  - **Clock Polarity (CPOL) – Clock Base Level**
    - CPOL defines the **idle (base) level** of the clock when no data transfer is occurring

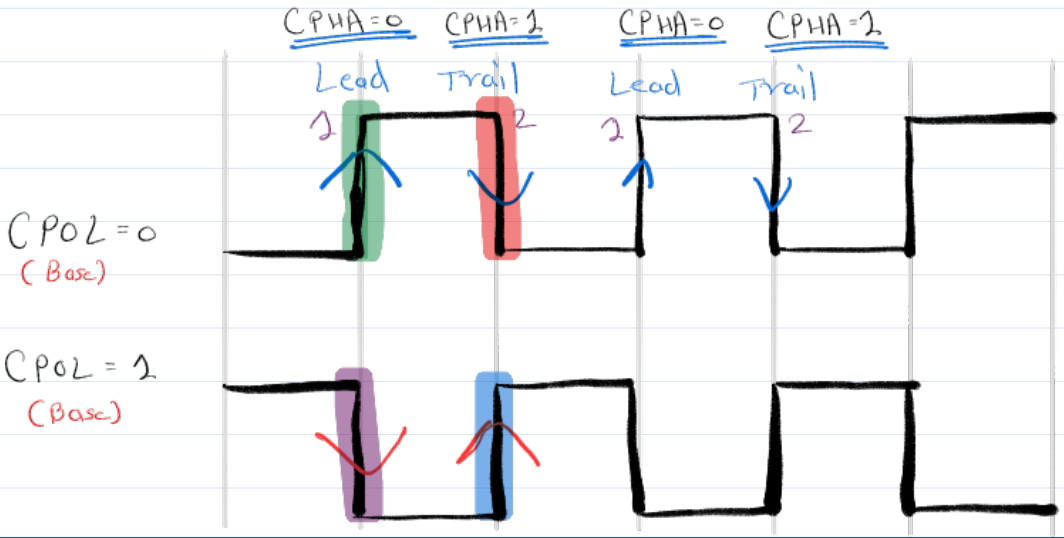
CPOL	Clock Idle State
0	Clock is LOW when idle
1	Clock is HIGH when idle

  - **Clock Phase (CPHA) – Data Sampling Edge**
    - CPHA defines **when data is sampled** relative to the clock edge

CPHA	Data Sampled On
0	Leading (first) edge
1	Trailing (second) edge

*Logical Char

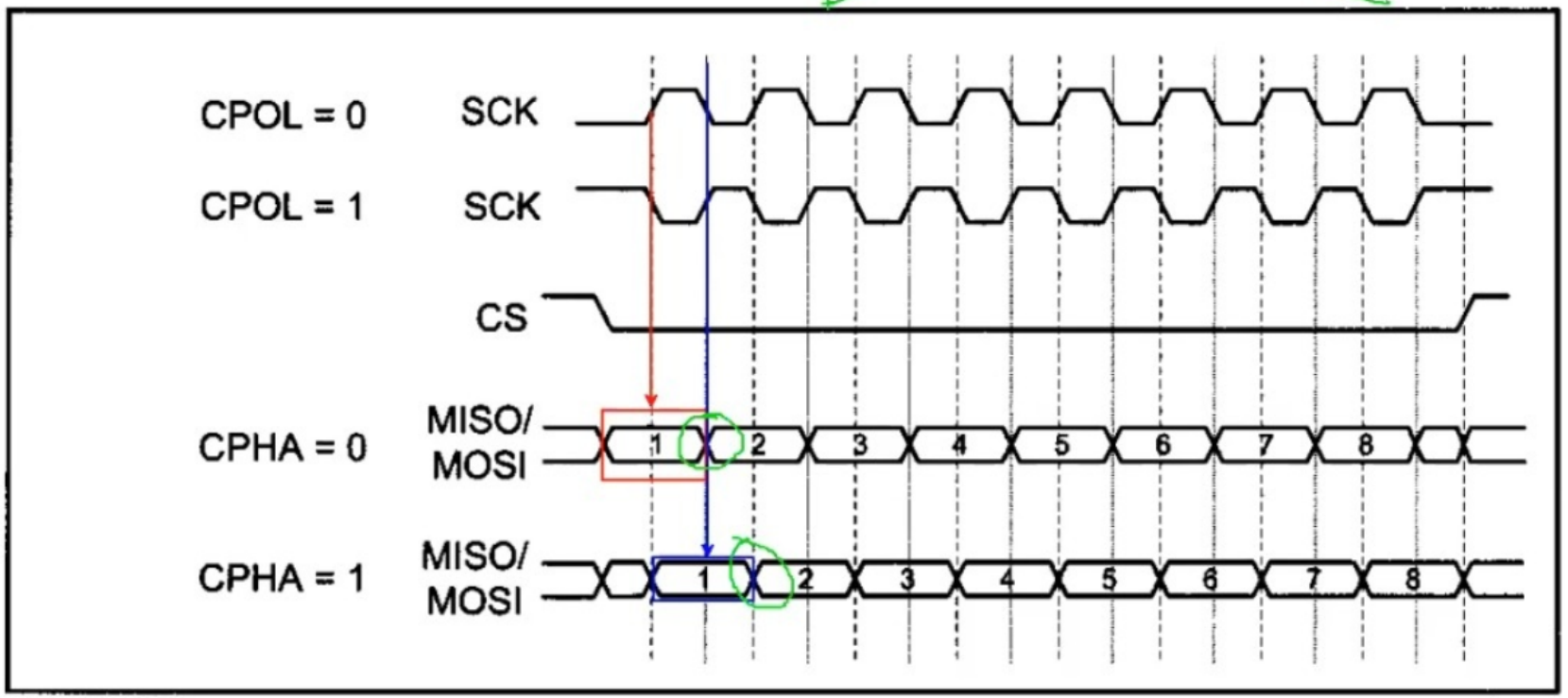




← Select mode it depends on slave

Table 17-1: SPI Clock Polarity and Phase

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



Select Mode depend on slave

- **Data Frame Format**
  - SPI data frame is usually:
    - **8-bit (1 byte)** – most common
  - One complete data frame requires **8 clock pulses** (for 8-bit)
  - **Single-Byte Transfer**
    - **Master controls slave selection**
    - **Slave selection is the responsibility of the Master**
    - Communication sequence:
      - **SS=0 + MOSI --> 8 Data Bits --> MISO + SS=1**
      - **SS=0 + MISO <-- 8 Data Bits <-- MOSI + SS=1**
  - **Multi-byte write**
    - Master sends internal address (1-byte or 2-byte) followed by Number of data bytes.
      - **SS=0 + Master --> Addr1, Addr2, Byte1, Byte2, ..., Byten --> Slave + SS=1**
  - **Multi-byte read**
    - Master sends internal address (1-byte or 2-byte) and then Slave send Number of data bytes.
      - **SS=0 + Master --> Addr1, Addr2 --> Slave + Slave --> Byte1, Byte2, ..., Byten --> Master + SS=1**

Single-byte write

The following steps are used to send (write) data in single-byte mode for SPI devices, as shown in Figure 17-4:

1. Make CE = 0 to begin writing.
2. The 8-bit address is shifted in, one bit at a time, with each edge of SCLK. Notice that A7 = 1 for the write operation, and the A7 bit goes in first.
3. After all 8 bits of the address are sent in, the SPI device expects to receive the data belonging to that address location immediately.
4. The 8-bit data is shifted in one bit at a time, with each edge of the SCLK.
5. Make CE = 1 to indicate the end of the write cycle.

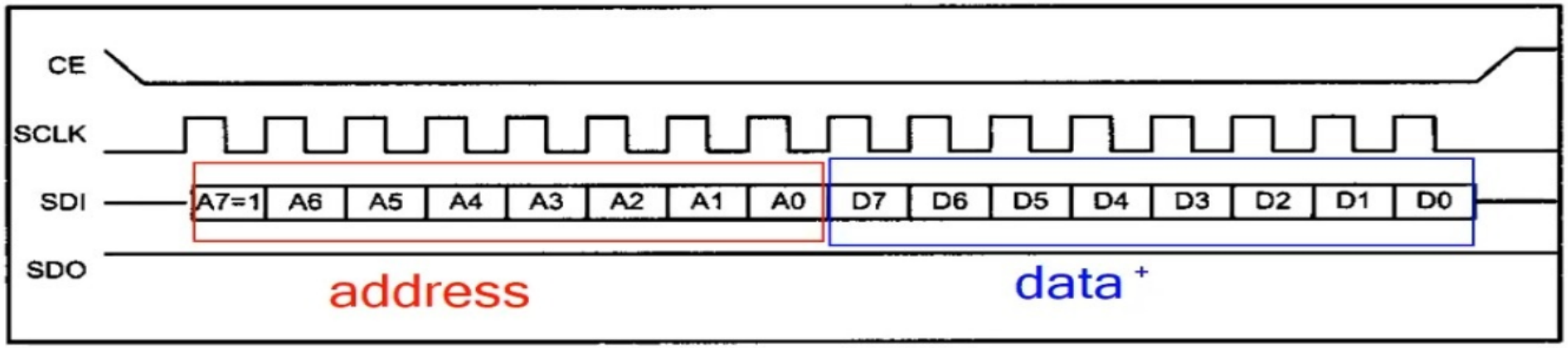


Figure 17-4. SPI Single-Byte Write Timing (Notice A7 = 1)

Multibyte burst write

Burst mode writing is an effective means of loading consecutive locations. In burst mode, we provide the address of the first location, followed by the data for that location. From then on, while CE = 0, consecutive bytes are written to consecutive memory locations. In this mode, the SPI device internally increments the address location as long as CE is LOW. The following steps are used to send (write) multiple bytes of data in burst mode for SPI devices as shown in Figure 17-5:

1. Make CE = 0 to begin writing.
2. The 8-bit address of the first location is provided and shifted in, one bit at a time, with each edge of SCLK. Notice that A7 = 1 for the write operation and the A7 bit goes in first.
3. The 8-bit data for the first location is provided and shifted in, one bit at a time, with each edge of the SCLK. From then on, we simply provide consecutive bytes of data to be placed in consecutive memory locations. In the process, CE must stay low to indicate that this is a burst mode multibyte write operation.
4. Make CE = 1 to end writing.

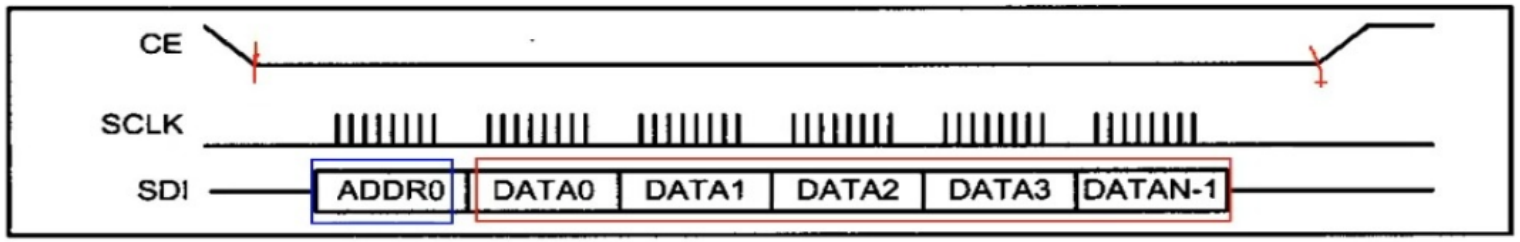


Figure 17-5. SPI Burst (Multibyte) Mode Writing

*Multibyte Read

1. Make CE = 0 to begin reading.
2. The 8-bit address of the first location is provided and shifted in, one bit at a time, with each edge of SCLK. Notice that A7 = 0 for the read operation, and the A7 bit goes in first.
3. The 8-bit data for the first location is shifted out, one bit at a time, with each edge of the SCLK. From then on, we simply keep getting consecutive bytes of data belonging to consecutive memory locations. In the process, CE must stay LOW to indicate that this is a burst mode multibyte read operation.
4. Make CE = 1 to end reading.

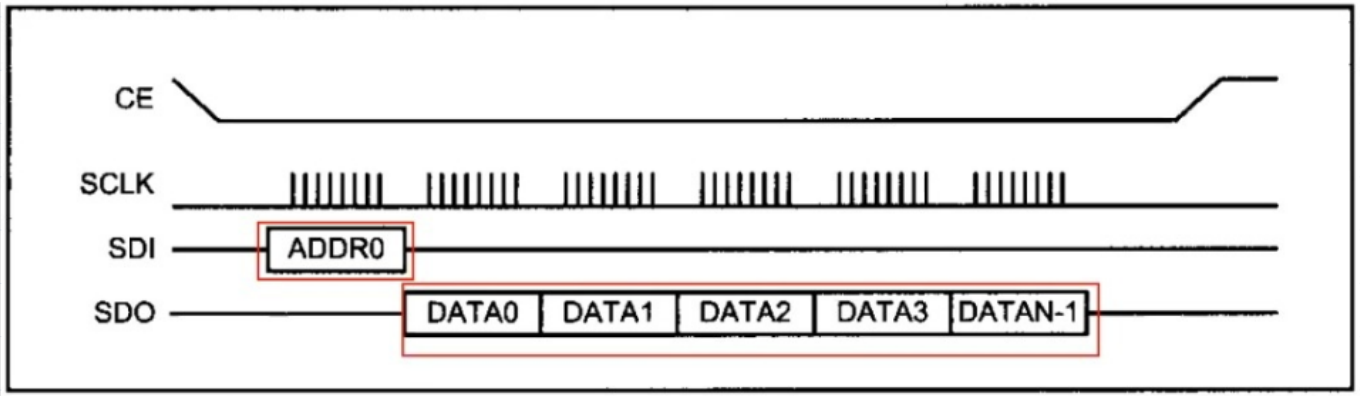


Figure 17-7. SPI Burst (Multibyte) Mode Reading

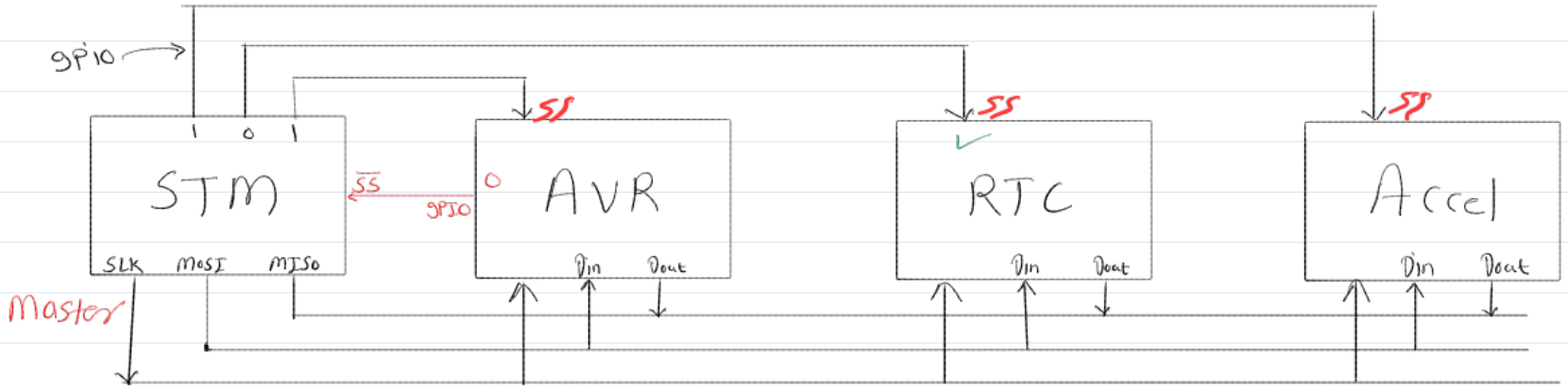
Applications / Devices of SPI

- **Accelerometer**
  - Used to measure acceleration, tilt, and motion.
  - SPI provides fast data transfer, suitable for real-time motion sensing applications.
- **EEPROM / Flash Memory**
  - Used for non-volatile storage of configuration and calibration data.
  - SPI enables faster read/write operations compared to I²C memory devices.
- **LCD Display**
  - Used to display text or graphics in embedded systems.
  - SPI reduces GPIO usage and allows high refresh rate display updates.
- **7-Segment Display**
  - Used for numeric data display such as counters and clocks.
  - SPI is used with display driver ICs to control multiple digits efficiently.
- **RTC (Real-Time Clock)**
  - Used to maintain accurate date and time information.
  - SPI-based RTCs offer reliable and faster communication with the microcontroller.

Error conditions in SPI

- **Read Overrun Error**
  - Occurs when **received data is not read in time**
  - Before the CPU reads the received data register, **new data arrives**
  - New data **overwrites old data**
  - Result: **Loss of previously received data**
  - Common in: High SPI clock speed , Slow interrupt or polling handling
- **Write Collision Error**
  - Occurs when **new data is written** before the previous transmission is completed
  - Transmit buffer gets **overwritten**
  - Result: **Incorrect or lost transmitted data**
  - Caused by: Not checking transmit buffer empty flag , Improper synchronization in code
- **Mode Fault Error**
  - Occurs when SPI is configured as **Master**
  - Another device **forces the SS line LOW**
  - This indicates **multiple masters on the bus**

* Mode fault Error

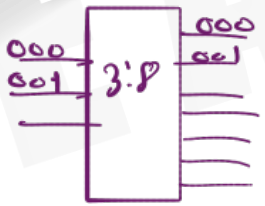


* Here STM is capable to become master / slave
* During the operation of STM is master at that time
  Suppose AVR make STM master to slave (forcefully) Mode fault occurs.

- **Slave Abort Error**
  - Occurs when:
    - SPI Master is communicating with a Slave
    - Slave **stops responding suddenly**
  - Can be due to:
    - Power loss of slave , Hardware fault ,Incorrect slave selection
  - Master receives invalid or no data
- **CRC Error (STM32 Specific)**
  - STM32 SPI peripheral supports **hardware CRC generation and checking**
  - CRC is calculated on transmitted and received data
  - If **received CRC ≠ calculated CRC**, error occurs
  - Used in: Safety-critical ,Reliable data communication applications

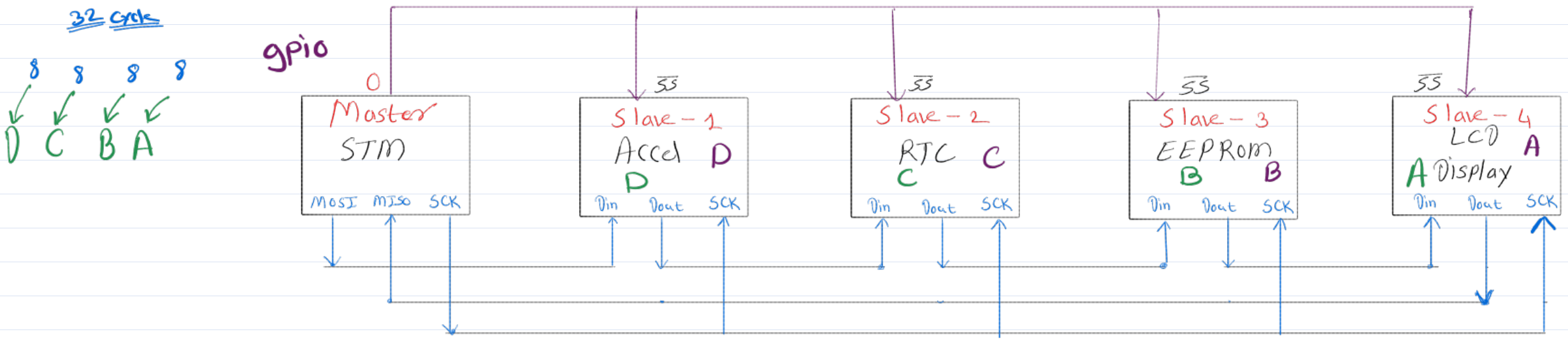
Daisy Chaining in SPI

- **Conventional SPI Slave Selection**
  - In SPI, the **Master selects a Slave using the SS (Slave Select) pin**
  - For **multiple slaves**:
    - Each slave requires a **separate SS line**
    - Master uses **multiple GPIO pins** for slave select.
  - Limitation : As number of slaves increases, **GPIO usage increases**
- **Reducing SS Pins Using a MUX / Decoder**
  - A **decoder / multiplexer (e.g., 3-to-8 decoder)** can be used
  - Using **3 GPIO pins**, the Master can select:
    - **1 out of 8 slave devices**
  - Reduces GPIO usage compared to direct SS lines But Still requires **external hardware**



- **Daisy Chaining Concept**
  - **Daisy chaining allows multiple SPI slaves to be connected in series**
  - All slaves share:
    - **Same SCLK , Same SS line**
  - Data flows through slaves **one after another** and Only **ONE SS pin** is required from Master .

* Daisy Chaining :



- **Daisy Chain Connection**
  - **MOSI of Master → SDI of Slave 1**
  - **SDO of Slave 1 → SDI of Slave 2**
  - **SDO of Slave 2 → SDI of Slave 3 → …**
  - **SDO of last Slave → MISO of Master**
- **Data Transfer in Daisy Chaining**
  - Master pulls **SS = 0**
  - Data is shifted through **all slaves**
  - Each slave:
    - Captures its own data
    - Passes remaining bits to next slave
  - Master sends **N × data bits** for N slaves
  - Example : For 3 slaves (8-bit each) → Master sends **24 clock pulses**

- **Advantages of Daisy Chaining**
  - **Only one SS pin required**
  - Reduced GPIO usage
  - Simple wiring
  - Ideal for **identical devices**
- **Limitations of Daisy Chaining**
  - Slaves must support **daisy-chain mode**
  - Data latency increases with number of slaves
  - Individual slave addressing is not possible
  - Failure of one slave may affect others

STM32 SPI

- 3 SPI ports
- Full-duplex synchronous transfers on three lines
- 8 or 16 bit transfer frame format selection
- Master or slave operation
- Multi-master mode capability
- 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
- Hardware CRC feature for reliable communication:
- Master mode fault, overrun and CRC error flags with interrupt capability

Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com