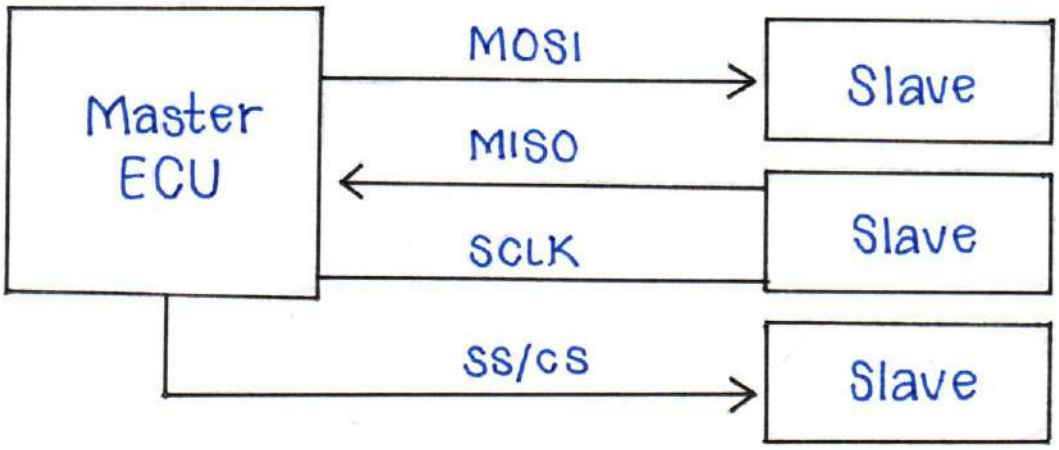
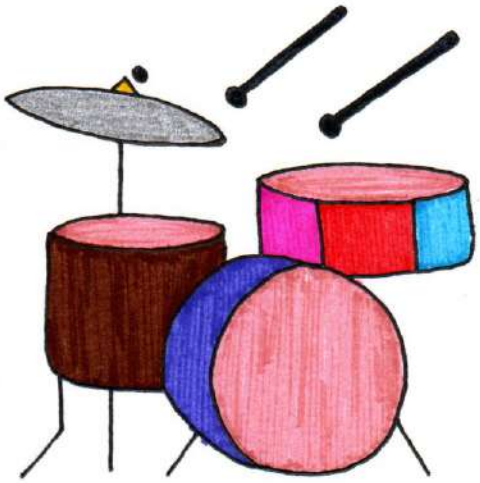
SPI PROTOCOL

Complete Interview Guide



- Full-duplex communication
- Faster than UART/I²C
- Multiple slaves with separate CS lines
- Widely used in automotive sensors, displays, memory



SPI is like a band with a drummer — the Master sets the beat, and everyone plays in sync.

Simplified • Automotive-Focused • Interview-Ready

SPI PROTOCOL

1. Introduction :-

- SPI (Serial Peripheral Interface) is a synchrous, full-duplex, master-slave communication protocol developed by Motorola.
- It is widely used in embedded and automotive systems to connect microcontrollers with peripherals like sensors, EEPROMs, ADC/DACs, Flash memory, and displays.
- Communication is always initiated by the Master, which generates the clock.

2. Basic SPI Communication Diagram :-

Master		Slave
MOSI	----------▶	MOSI (Data In)
MISO	◀----------	MISO (Data Out)
SCK	----------▶	SCK (Clock In)
CS	----------▶	CS (Active Low)

- MOSI :- Master Out Slave In
- MISO :- Master In Slave Out
- SCK :- Serial Clock from Master
- CS/SS :- Chip Select (active low, enables one slave)

3. Key Features :-

- Full-duplex communication.
- High speed (up to tens of MHz).
- Simple hardware interface (MOSI, MISO, SCK, CS).
- Supports multiple slaves (each needs CS).
- Configurable word size (commonly 8 bits, can be 16/32).

4. SPI Frame Format (with Modes) :-

```
Clock (SCK):   _-_-_-_-_-_-_-_
MOSI (Tx):     1 0 1 0 1 1 0 1
MISO (Rx):     0 1 0 0 1 0 1 1
```

- Data is shifted bit by bit with each clock pulse.
- MSB-first (default) or LSB first depending on configuration.
- Modes (CPOL & CPHA) :

Mode	CPOL	CPHA	Clock Idle	Data Sampled
0	0	0	Low	Rising Edge
1	0	1	Low	Falling Edge
2	1	0	High	Falling Edge
3	1	1	High	Rising Edge

- CPOL :- Clock Polarity.
- CPHA :- Clock Phase.

5. Practical Automotive Use Cases :-

- Sensors : Pressure, accelerometers, and temperature sensors connect via SPI to the ECU.
- Flash Memory : Infotainment systems and dashboards use SPI Flash for Firmware storage.
- Displays : Instrument clusters and infotainment displays often use SPI controllers.
- Reliability : SPI ensures fast, deterministics communication, which is crucial for automotive safety.

Interview Q&A

1. What is SPI and why it is used ?

⇒ SPI (Serial Peripheral Interface) is a synchronous, full-duplex communication protocol where data is exchanged between a master (usually MCU/processor) and one or more slaves (sensors, memory devices, displays).

- Used because of its high speed, simplicity and low overhead.
- Ideal for short range, board-level communication in embedd/automotive systems.
- Typical use case : Flash memory, ADC/DAC chips, sensors, infotainment system ECUs.

2. Explain the 4 SPI modes.

⇒ Defined by Clock Polarity (CPOL) and Clock Phase (CPHA) :

- Mode 0 :- Clock idle low, data sampled on rising edge.
- Mode 1 :- Clock idle low, data sampled on falling edge.
- Mode 2 :- Clock idle high, data sampled on falling edge.
- Mode 3 :- Clock idle high, data sampled on rising edge.

Why Important : Because master and slave must use the same mode or data corruption occurs.

3. Can SPI handles multiple slaves ?

⇒ Yes. Two approaches.

1. Chip Select (CS) :- Each slave has its own CS pin from the master. Only one CS active at a time.
2. Daisy Chain :- Devices connected in series, data passes through each slave. (Used in shift registers or LED drivers).

4. What is Daisy Chaining in SPI ?

⇒ A technique where multiple slaves are connected in series.

- MOSI of master → Slave 1 → Slave 2 → Slave 3 ..... → back to master.
- Data shifts through the chain (like a pipeline).

- Useful when many devices are present but pin count must be minimized.

5. Why is SPI is faster than I2C ?

⇒ Because :

- No addressing overhead (I^2C has address + ACK Cycles).
- Dedicated lines for Tx / Rx (Full duplex).
- Can run at much higher clock speeds (10 - 100 MHz vs ~ 400 KHz - 1 MHz for I2C).
- Simple protocol - less overhead.

6. What happens if two slaves are selected simultaneously ?

⇒ Both slave drive the MISO line at the same time → bus conflict → corrupted data and possible hardware damage (if not protected by resistors).

Good Practice : Always ensure only one CS line is active.

7. Typical SPI clock speeds in automotive systems ?

⇒ Varies by devices and use case :

- 1-5 MHz → Common for sensors (temperature, pressure, accelerometers).
- 10-20 MHz → Used for Flash memories, EEPROMs, infotainment storage.
- Higher clocks possible (50-100 MHz) in advanced ECUs but depends on hardware design & EMI constraints.

8. What is BSP in SPI ?

⇒ BSP = Board Support Package.

- Provides low-level drivers for hardware peripherals (like SPI, I2C, CAN) for a specific board/processor.
- In SPI context : BSP contains initialization code, pinmux settings, and drives APIs to access SPI hardware from Linux or RTOS.
- Without BSP, engineers would need to manually configure registers.

9. SPI vs UART vs I^2C :

⇒

Features	SPI	I^2C	UART
Duplex	Full (MOSI/MISO)	Half (bidirectional SDA)	Full (Tx/Rx)
Speed	High (10s of MHz)	Medium (upto a few MHz)	Medium (upto a few MHz)
Pins	4+ (MOSI, SCK, MISO, CS per slave)	2 (SDA, SCL)	2 (Tx/Rx)
Multi-slave	Yes (via CS lines or daisy chain)	Yes (addressing)	No
Distance	Short (board-level)	Short-medium	Medium
Use Case	Flash, sensors, display.	Configuration ICs, sensors.	PC ⟷ MCU, GPS Module.

10. How does Linux handles SPI ?

⇒ Linux provides SPI Access via the spidev framework.

- Devices appear as /dev/spidevX.Y .
- User-space applications use ioctl() calls to configure mode, speed, and transfer data.
- Example steps :
  1. Open device : open("/dev/spidev0.0", O_RDWR)
  2. Configure with ioctl(fd, SPI_IOC_MESSAGE_WR_MODE, &mode)
  3. Transfer with ioctl(fd, SPI_IOC_MESSAGE, &tr).

Note :- In automotive Linux systems, SPI is commonly used in MCUs, gateways, infotainment systems units for memory, sensors, and peripheral communication.

❀ Advantages and Disadvantages of SPI :

⇒ Advantages :

- Fastes than I^2C and UART . ✓
- Full-duplex transfes . ✓
- Flexible configuration . ✓
- Simple fos point-to-point communication . ✓

Disadvantages :

- Mose pins sequised (CS fos each slave) . ✗
- No addsessing mechanism like I^2C . ✗
- No inbuilt essos detection . ✗
- Limited to shost-distance communication . ✗

Practical Code Snippets

(A) Bare-Metal MCU Example (C)

```
// SPI Master Initialization
void SPI_Init(void) {
    DDRB |= (1<<PB5) | (1<<PB7);// MOSI, SCK as output
    DDRB &= ~(1<<PB6);          // MISO as input
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0); // Enable,
}                                            Master, f/16
void SPI_Transmit(uint8_t data) {
    SPDR = data;
    while(!(*SPSR & (1<<*SPIF))); // Wait for completion
}
uint8_t SPI_Receive(void) {
    while(!(*SPSR & (1<<*SPIF)));
    return *SPDR;
}.
```

(B) Linux User-Space Example (*spidev)

```
#include <*stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <*sys/ioctl.h>
#include <linux/*spi/*spidev.h>
int main() {
    int fd = open("/dev/*spidev0.0", O_RDWR);
    uint8_t tx[] = {0xAA, 0x55};
    uint8_t *x[2] = {0};
    uint32_t *speed = 500000;
    uint8_t bits = 8, mode = 0;
```

```
ioctl(fd, SPI_IOC_WR_MODE, &mode);
ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &bits);
ioctl(fd, SPI_IOC_WR_MAX_SPEED_Hz, &speed);
struct spi_ioc_transfer tr = {
        .tx_buf = (unsigned long)tx,
        .rx_buf = (unsigned long)rx,
        .len = 2,
        .speed_hz = speed,
        .bits_per_word = bits,
};
ioctl(fd, SPI_IOC_MESSAGE(1), &tr);
printf("Received: %02X %02X\n", rx[0], rx[1]);
close(fd);
return 0;
}.
```

Summary :

- SPI :- fast, full-duplex, short-range communication protocol.
- Uses MOSI, MISO, SCK, CS.
- Operates in 4 modes (CPOL & CPHA).
- Key in automotive and embedded systems (sensors, memory, displays).
- Must Know :- advantages, disadvantages, interview Q&A, and practical code.