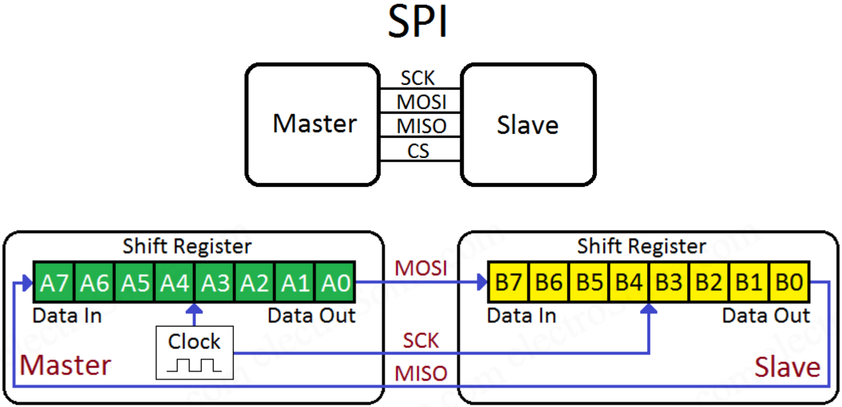
Module 9

SPI interface

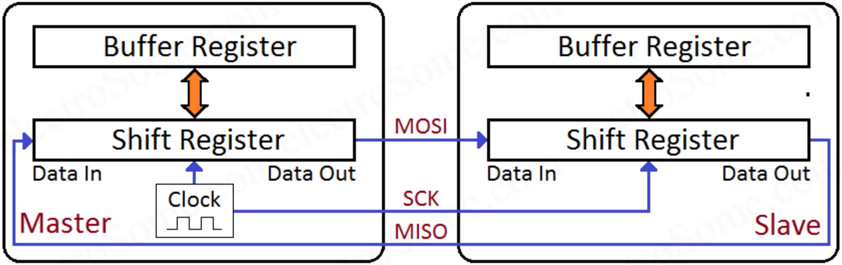
SPI features

- *Serial Peripheral Interface* (SPI) is a synchronous serial communication interface
- Supports full-duplex communication between one *SPI master* (a microcontroller) and one or more *SPI slaves* (peripherals)
- It uses minimum three lines (excluding the ground):
  - Serial Clock (SCLK)
  - Master In Slave Out (MISO)
  - Master Out Slave In (MOSI)
- High bitrates (comparing to UART) are possible on short distance
- Logic high voltage level depends on power supply

SPI internal structure (1)

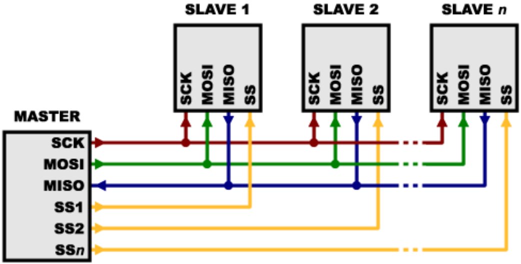


SPI internal structure (2)

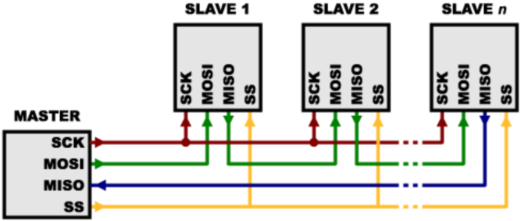


SPI configurations

Bus topology



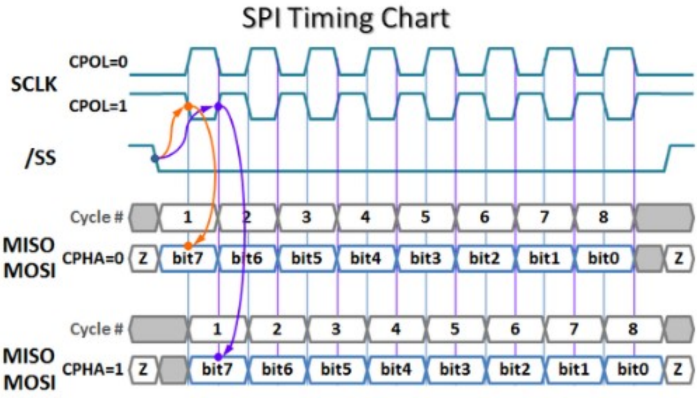
Daisy-chain topology



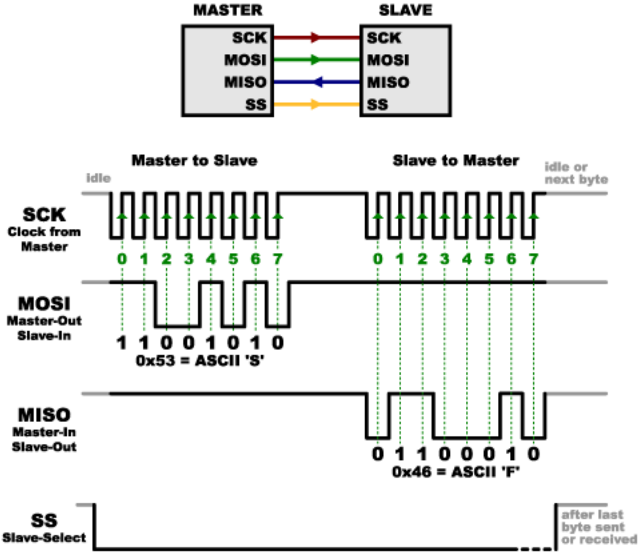
SPI modes (1)

- Idle state of the clock signal is defined with a parameter called *clock polarity* (CPOL):
    - CPOL = 0 means clock is low when idle
    - CPOL = 1 means clock is high when idle
- Setup and sample instants are determined with a *clock phase* (CPHA) parameter:
    - CPHA = 0: data is sampled on the first and changed on the second edge of the clock
    - CPHA = 1: data is changed on the first and sampled on the second edge of the clock

SPI modes (2)


SPI Timing Chart
SCLK
CPOL=0
CPOL=1
/SS
Cycle #
1
2
3
4
5
6
7
8
MISO
MOSI
CPHA=0
Z
bit7
bit6
bit5
bit4
bit3
bit2
bit1
bit0
Z
Cycle #
1
2
3
4
5
6
7
8
MISO
MOSI
CPHA=1
Z
bit7
bit6
bit5
bit4
bit3
bit2
bit1
bit0
Z

SPI example communication



Linux spidev API

SPI device in Linux can be accessed from userspace via /dev/spidevA.B:

- open() to open an SPI device
- Only basic half-duplex can be achieved using read() and write() access
- Full-duplex I/O transfer and interface configuration is achieved using ioctl() requests and struct spi_ioc_transfer structure with the following fields:
  - tx_buf: pointer to transmit buffer
  - rx_buf: pointer to receive buffer
  - len: number of bytes
  - speed_hz: bitrate in Hz
  - delay_usecs: delay between transfers in microseconds
  - bits_per_word: number of bits in a word
- close() to close the SPI device when done

Opening and closing an SPI device

- Opening:

  ```
  fd = open("/dev/spidev0.0", O_RDWR);
  if (fd == -1)
       printf("Failed to open port.\n");
  ```

- Closing:

  ```
  close(fd);
  ```

Example configuration

```
int speed = 1000000;
int mode = 2;
int size = 8;
...
ioctl(fd, SPI_IOC_WR_MODE, &mode); // mode (CPOL:CPHA)
ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &size); // word size
ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed); // max speed
```

More information can be found on
`https://www.kernel.org/doc/Documentation/spi/spidev`

Example reading/writing

```
struct spi_ioc_transfer spi;
unsigned char buffer = 0x55;
...
for (;;)
{
    spi.tx_buf = (unsigned long)(&buffer);
    spi.rx_buf = (unsigned long)(&buffer);
    spi.len = 1;
    spi.delay_usecs = 0;
    spi.speed_hz = 1000000;
    spi.bits_per_word = 8;

    ioctl(fd, SPI_IOC_MESSAGE(1), &spi);
}
```