Microprocessor Applications

Serial Peripheral Interface (SPI)

UF UNIVERSITY *of* FLORIDA

UF ECE

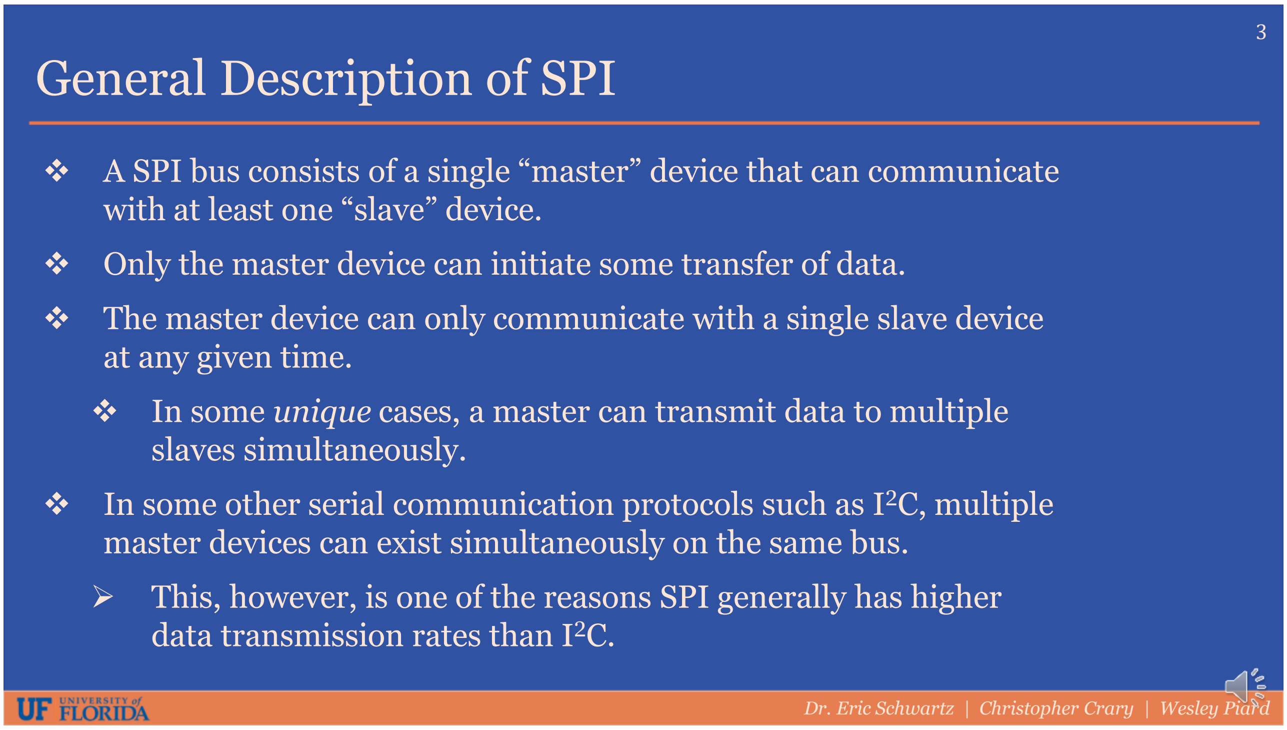
Dr. Eric Schwartz
Christopher Crary
Wesley Piard

1

General Description of SPI

- ❖ Serial Peripheral Interface (**SPI**) is a synchronous serial communication interface that was originally defined by Motorola.
- ❖ It is one of the most common serial communication interfaces, and it is typically utilized by components such as sensors, memory, etc.
- ❖ It is generally chosen for applications where short-distance, full-duplex communication is required.

2

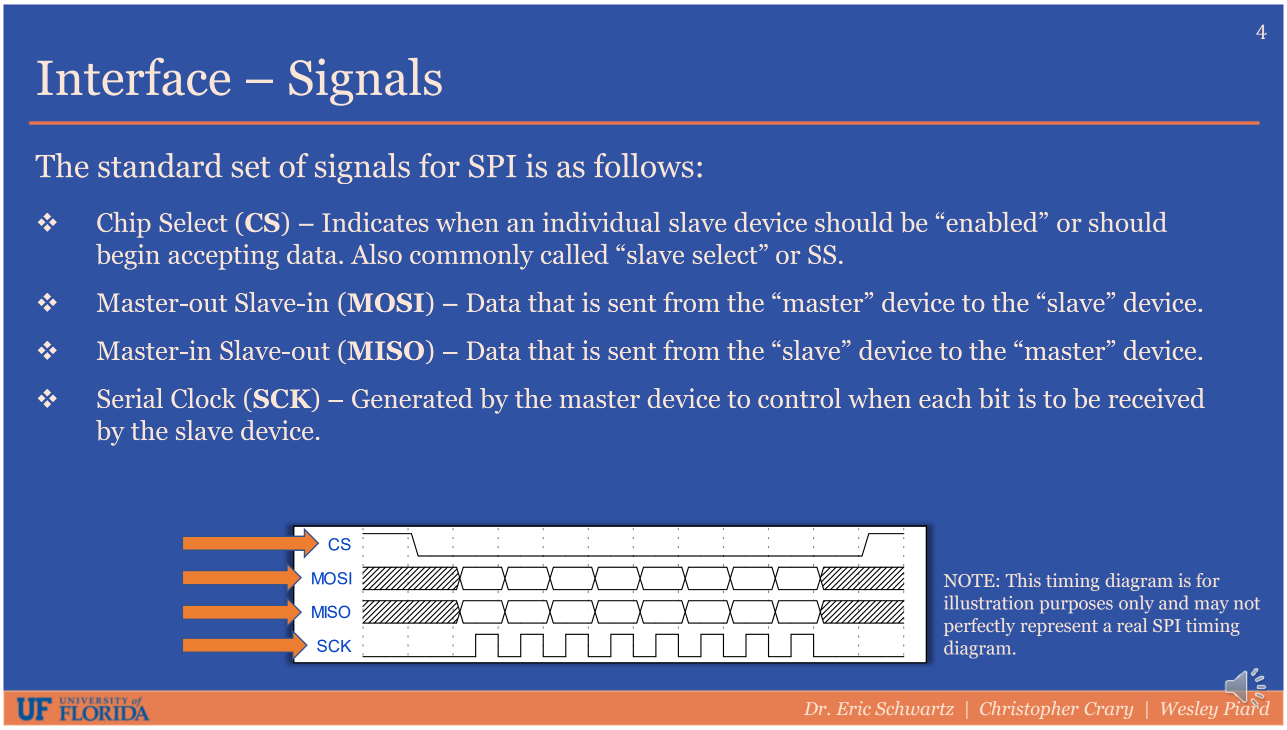


General Description of SPI

- ❖ A SPI bus consists of a single "master" device that can communicate with at least one "slave" device.
- ❖ Only the master device can initiate some transfer of data.
- ❖ The master device can only communicate with a single slave device at any given time.
  - ❖ In some *unique* cases, a master can transmit data to multiple slaves simultaneously.
- ❖ In some other serial communication protocols such as I^2C, multiple master devices can exist simultaneously on the same bus.
  - ➢ This, however, is one of the reasons SPI generally has higher data transmission rates than I^2C.

UF UNIVERSITY of FLORIDA Dr. Eric Schwartz | Christopher Crary | Wesley Piard

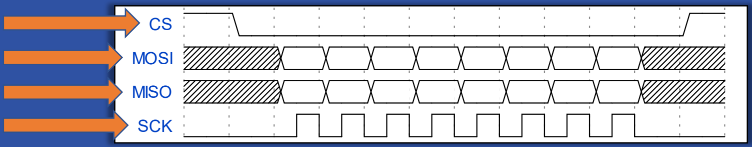
3



Interface – Signals

The standard set of signals for SPI is as follows:

- ❖ Chip Select (**CS**) – Indicates when an individual slave device should be "enabled" or should begin accepting data. Also commonly called "slave select" or SS.
- ❖ Master-out Slave-in (**MOSI**) – Data that is sent from the "master" device to the "slave" device.
- ❖ Master-in Slave-out (**MISO**) – Data that is sent from the "slave" device to the "master" device.
- ❖ Serial Clock (**SCK**) – Generated by the master device to control when each bit is to be received by the slave device.



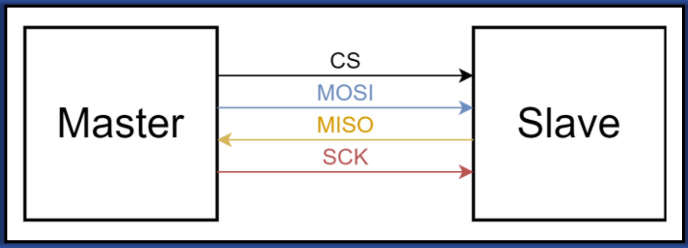
NOTE: This timing diagram is for illustration purposes only and may not perfectly represent a real SPI timing diagram.

UF UNIVERSITY of FLORIDA Dr. Eric Schwartz | Christopher Crary | Wesley Piard

4

Interface – Connections

❖ The four signals listed on the previous slide are depicted in this diagram.



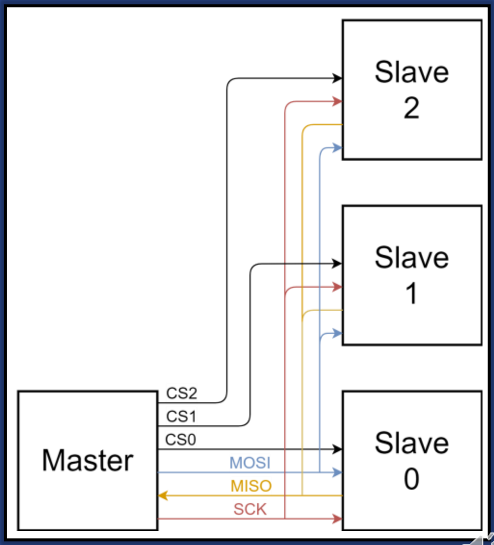
5

Interface – Multiple Slave Devices

❖ If multiple slave devices are required, they will **each require their own chip select**.

  ➢ Without individual chip select signals, data transmission errors would occur.

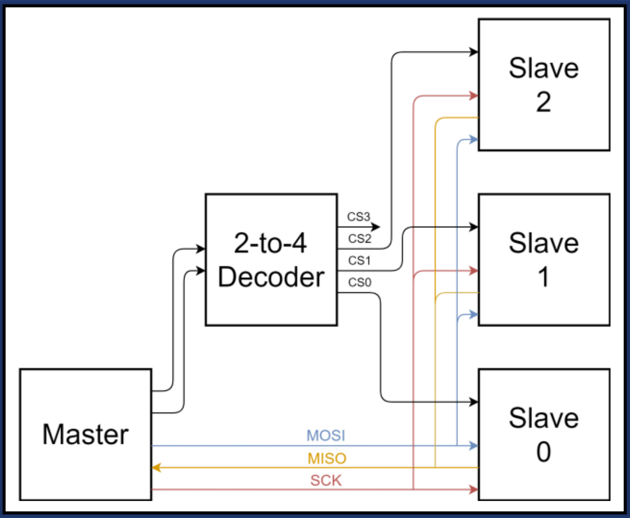
❖ Thus, more hardware connections are required for more slave devices.



6

Interface – Decoder Example

- A binary decoder can be used to limit the number of I/O pins used as chip selects by the master device.
- In the example given on the right, the decoder allows the master device to enable up to four slave devices with only two I/O pins.
- The extra cost of a decoder must be justified.

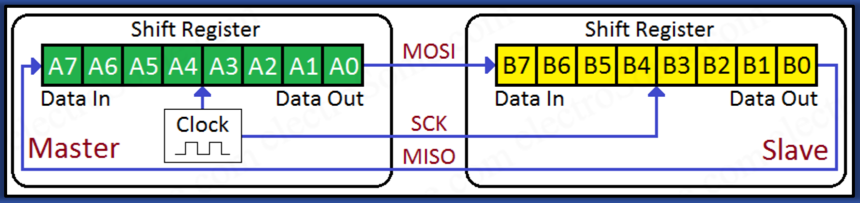


Interface – Shift Register Dynamics

- As mentioned at the beginning of this presentation, SPI is a full-duplex communication interface.
  - It is possible, if desired, to leave either the MOSI or MISO lines disconnected, if data only needs to be transferred in one direction.
- To implement the protocol, a shift register is generally utilized by both the master device and the slave device(s), where all shift registers are controlled by the same clock signal. Data is shifted out of the master, bit by bit, into a slave via the MOSI line; at the same time, data is shifted into the master device from a slave device via the MISO line.

Interface – Shift Register Dynamics

- ❖ As each bit gets shifted out of the master device, a bit from the slave device is shifted in.
- ❖ In this animation, the least-significant bit (LSb) is sent first, but SPI can generally be configured to alternatively send the most-significant bit (MSb) first.

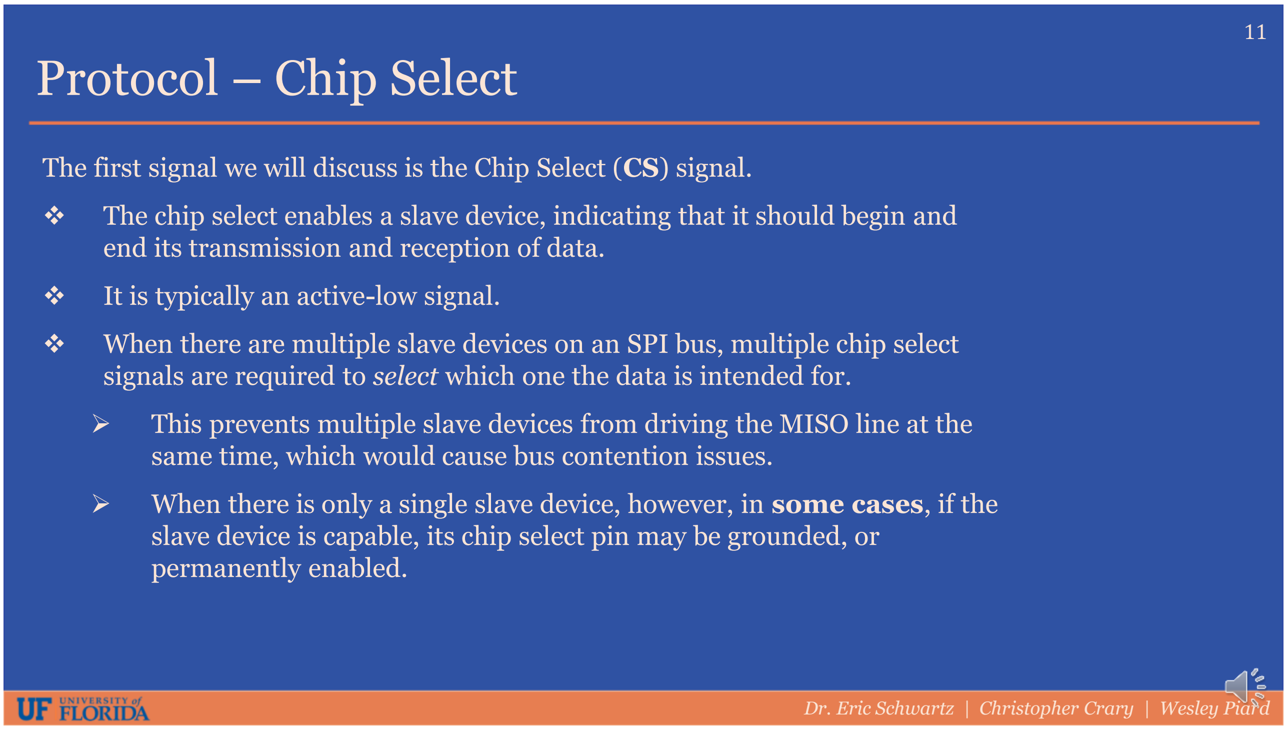


Animation credit: https://electrosome.com/spi/

Protocol

Now that an interface for SPI has been defined, we can move on to the protocol itself.

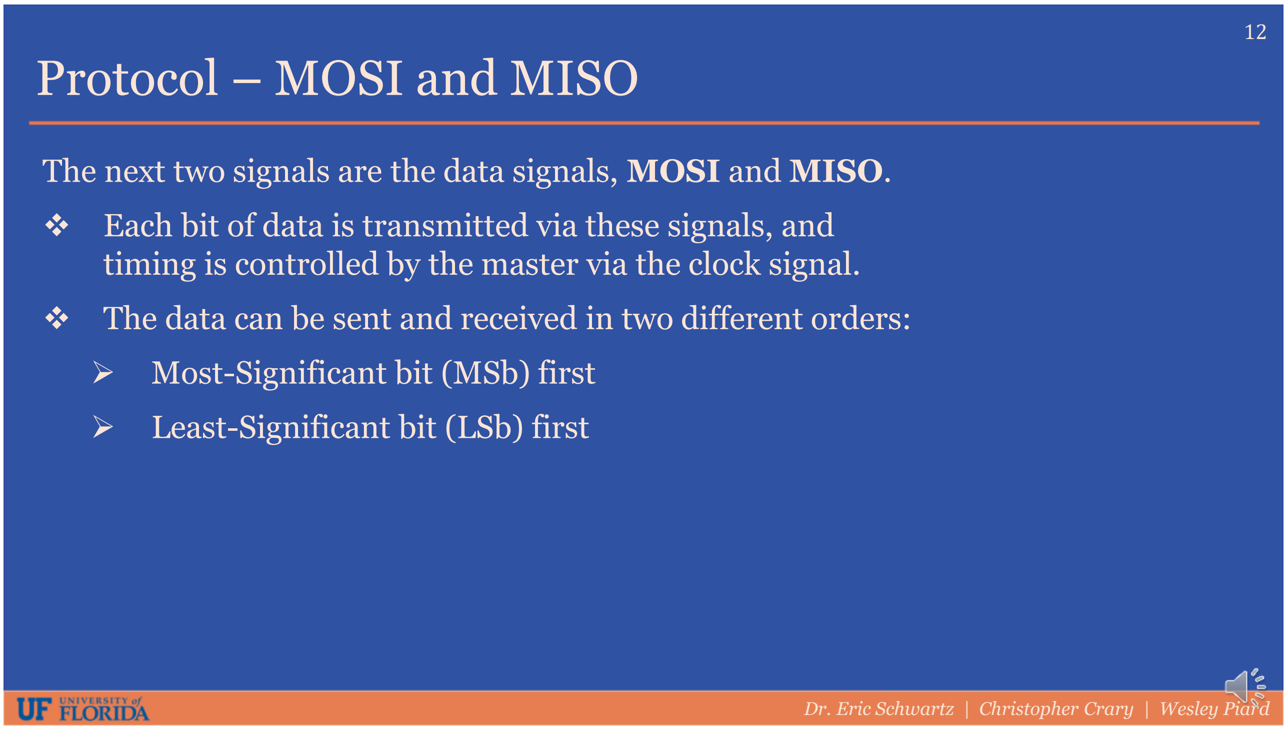
- ❖ While the *interface* describes the physical requirements and all the connections, the *protocol* describes the set of rules and characteristics that dictates how each signal must operate to ensure compatibility across SPI devices.



Protocol – Chip Select

The first signal we will discuss is the Chip Select (**CS**) signal.

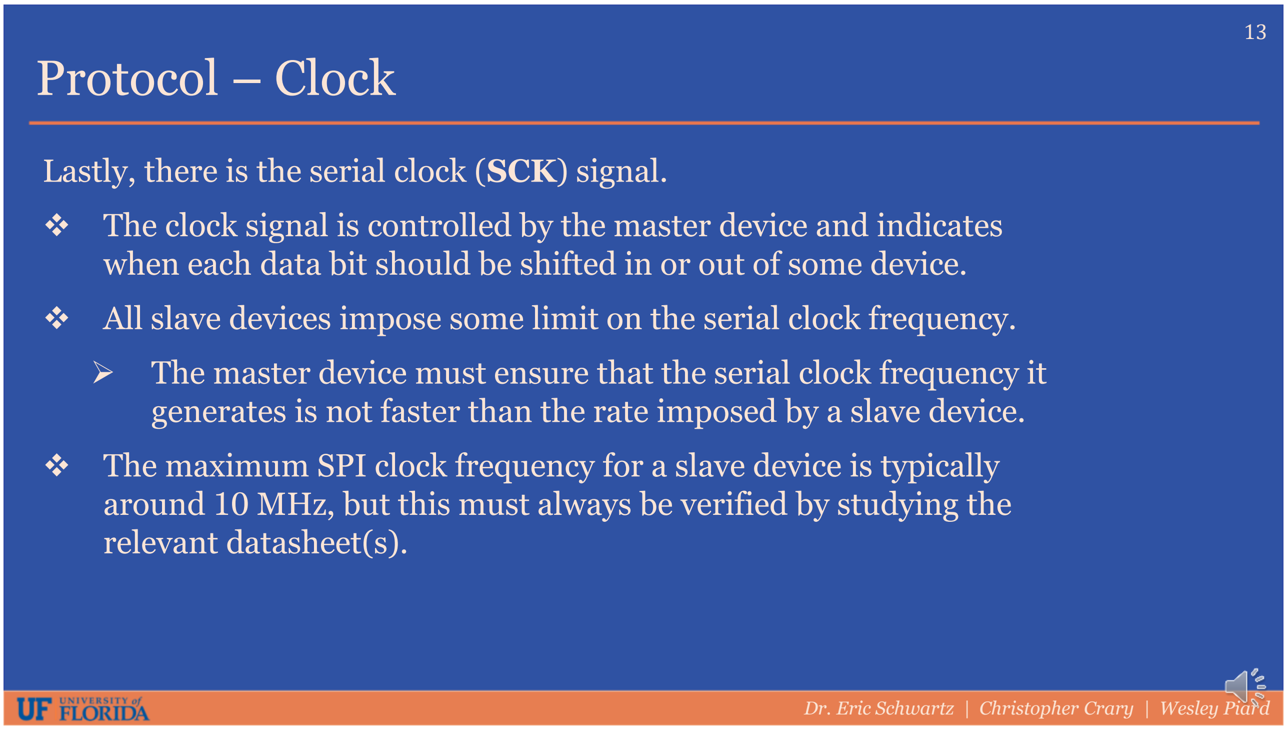
- ❖ The chip select enables a slave device, indicating that it should begin and end its transmission and reception of data.
- ❖ It is typically an active-low signal.
- ❖ When there are multiple slave devices on an SPI bus, multiple chip select signals are required to *select* which one the data is intended for.
  - ➢ This prevents multiple slave devices from driving the MISO line at the same time, which would cause bus contention issues.
  - ➢ When there is only a single slave device, however, in **some cases**, if the slave device is capable, its chip select pin may be grounded, or permanently enabled.



Protocol – MOSI and MISO

The next two signals are the data signals, **MOSI** and **MISO**.

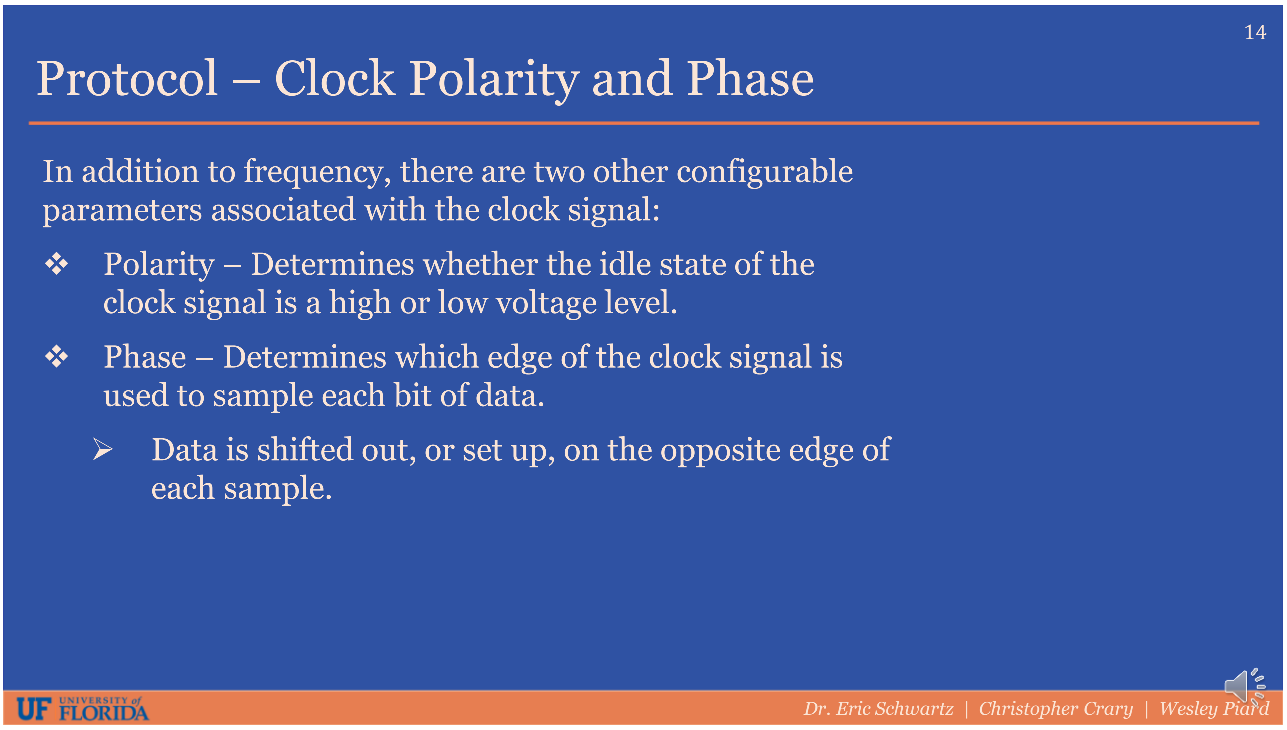
- ❖ Each bit of data is transmitted via these signals, and timing is controlled by the master via the clock signal.
- ❖ The data can be sent and received in two different orders:
  - ➢ Most-Significant bit (MSb) first
  - ➢ Least-Significant bit (LSb) first



Protocol – Clock

Lastly, there is the serial clock (**SCK**) signal.

- ❖ The clock signal is controlled by the master device and indicates when each data bit should be shifted in or out of some device.
- ❖ All slave devices impose some limit on the serial clock frequency.
  - ➢ The master device must ensure that the serial clock frequency it generates is not faster than the rate imposed by a slave device.
- ❖ The maximum SPI clock frequency for a slave device is typically around 10 MHz, but this must always be verified by studying the relevant datasheet(s).



Protocol – Clock Polarity and Phase

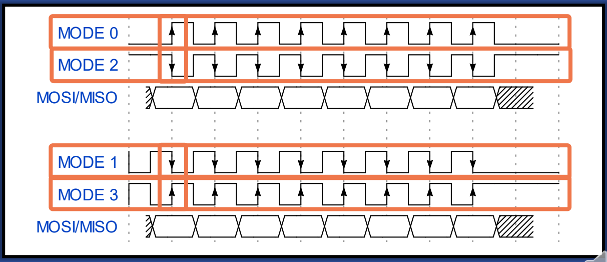
In addition to frequency, there are two other configurable parameters associated with the clock signal:

- ❖ Polarity – Determines whether the idle state of the clock signal is a high or low voltage level.
- ❖ Phase – Determines which edge of the clock signal is used to sample each bit of data.
  - ➢ Data is shifted out, or set up, on the opposite edge of each sample.

Protocol – Clock Phase and Polarity

- ❖ Since each of polarity and phase can be configured two different ways, there are generally four possible “modes” of operation.
- ❖ The master device must be configured such that its clock phase and polarity match that of a slave device.

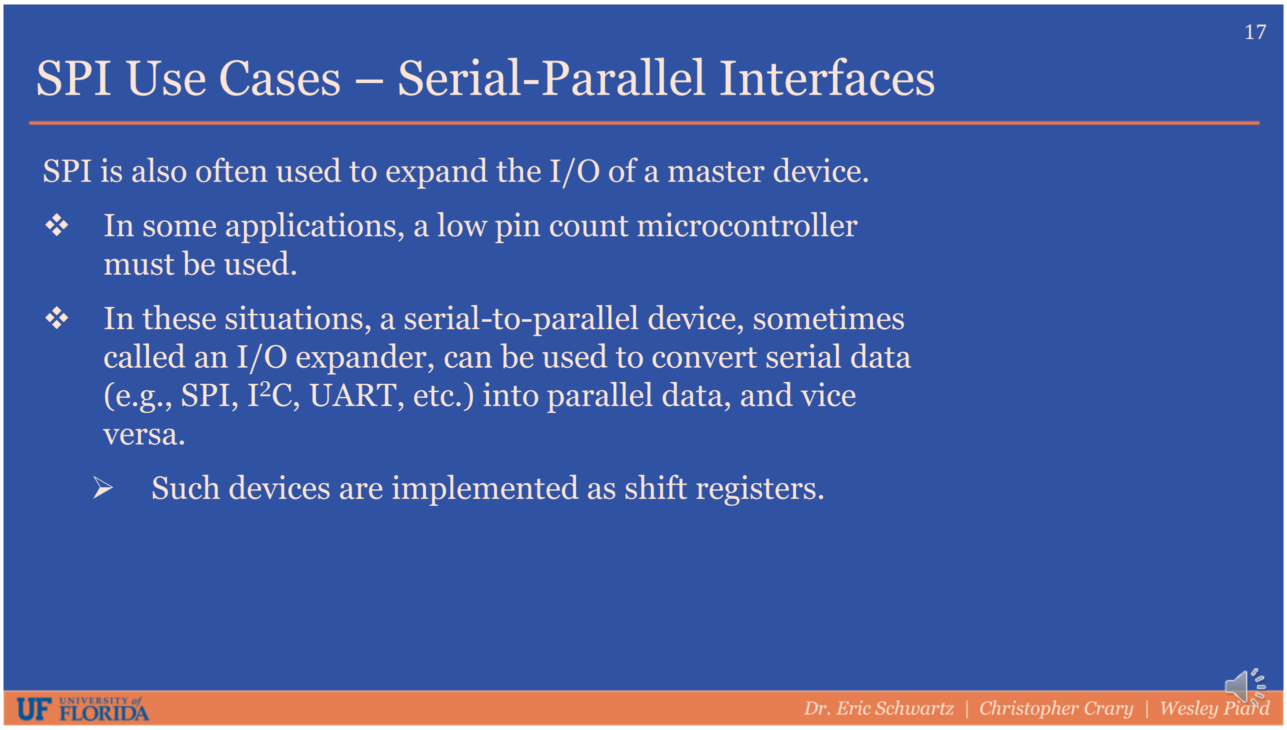
Mode	Polarity	Phase
0	Low	Rising Sample
1	Low	Falling Sample
2	High	Falling Sample
3	High	Rising Sample



SPI Use Cases – Sensors

SPI is often used in embedded systems to communicate with external sensors. Here are some common devices:

- ❖ Inertial Measurement Unit (**IMU**)
- ❖ Temperature sensor
- ❖ Barometer
- ❖ Analog-to-Digital converter (**ADC**)

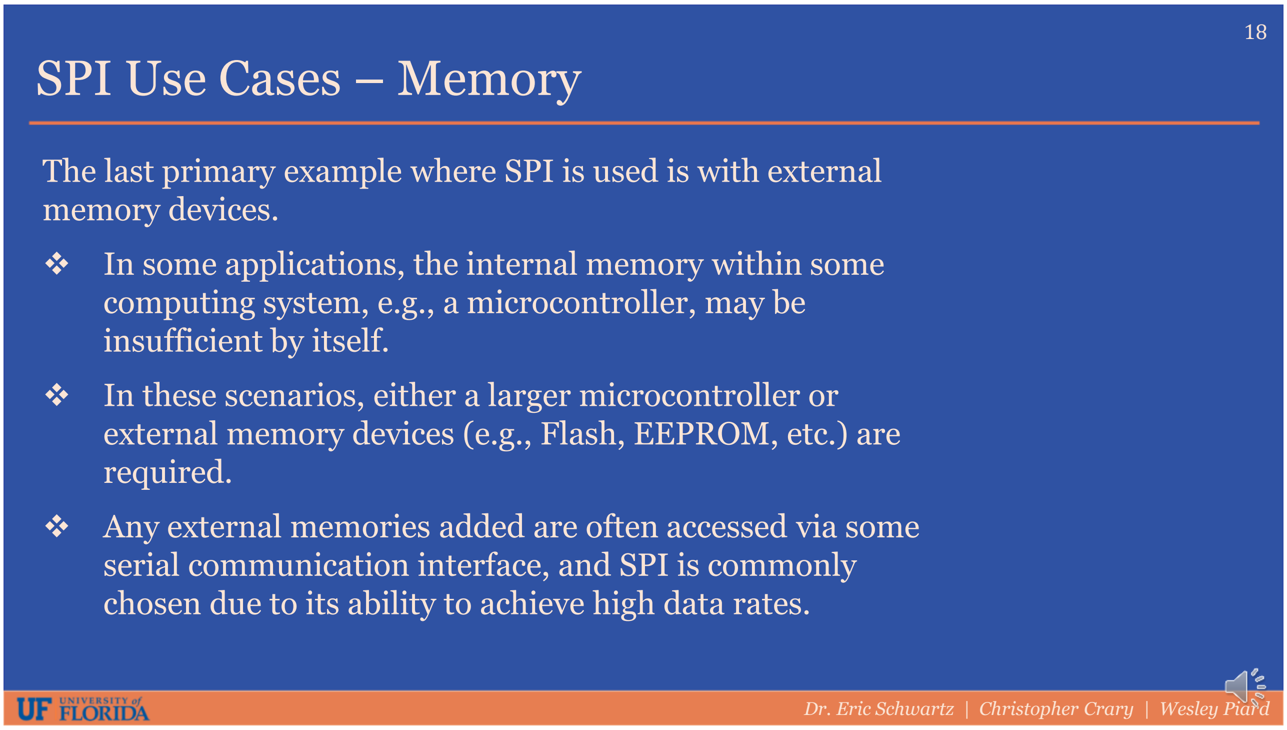


SPI Use Cases – Serial-Parallel Interfaces

SPI is also often used to expand the I/O of a master device.

- ❖ In some applications, a low pin count microcontroller must be used.
- ❖ In these situations, a serial-to-parallel device, sometimes called an I/O expander, can be used to convert serial data (e.g., SPI, I^{2}C, UART, etc.) into parallel data, and vice versa.
  - ➢ Such devices are implemented as shift registers.

17

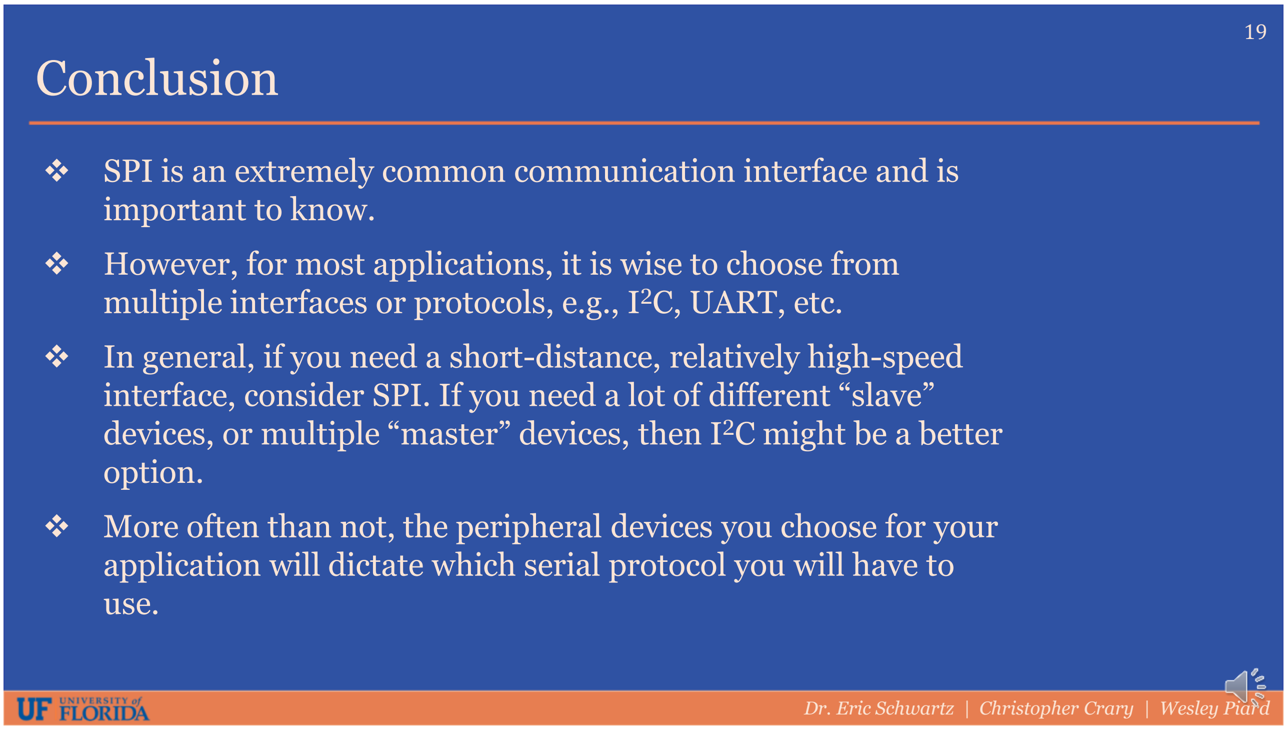


SPI Use Cases – Memory

The last primary example where SPI is used is with external memory devices.

- ❖ In some applications, the internal memory within some computing system, e.g., a microcontroller, may be insufficient by itself.
- ❖ In these scenarios, either a larger microcontroller or external memory devices (e.g., Flash, EEPROM, etc.) are required.
- ❖ Any external memories added are often accessed via some serial communication interface, and SPI is commonly chosen due to its ability to achieve high data rates.

18



Conclusion

- SPI is an extremely common communication interface and is important to know.
- However, for most applications, it is wise to choose from multiple interfaces or protocols, e.g., I^{2}C, UART, etc.
- In general, if you need a short-distance, relatively high-speed interface, consider SPI. If you need a lot of different “slave” devices, or multiple “master” devices, then I^{2}C might be a better option.
- More often than not, the peripheral devices you choose for your application will dictate which serial protocol you will have to use.