Hardware Interfacing

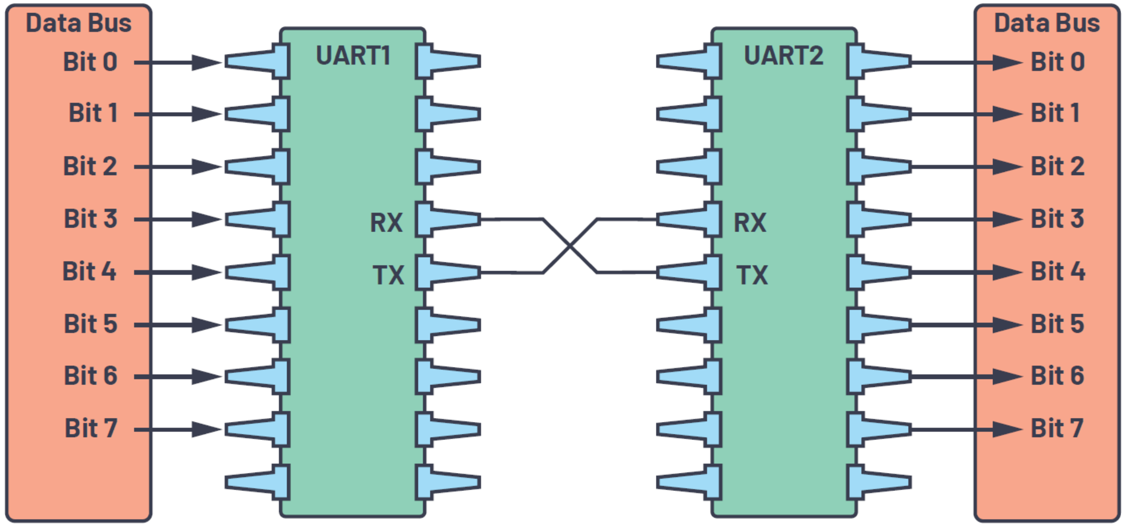
Instructor: Deepak Gangadharan

UART: Universal Asynchronous Receiver-Transmitter

- A device-to-device hardware communication protocol used by embedded systems, microcontrollers
- Example: Digital temperature sensor reporting ambient temperature to the microprocessor
- Uses asynchronous (no clock signal to synchronize the output bits) serial communication with configurable speed
- Two signals of each UART device
    - Tranmitter (Tx)
    - Receiver (Rx)

UART - continued

- Transmitting UART connected to a data bus that sends data in parallel
- Data is now sent on the transmission line serially to the receiving UART
- Serial data converted to parallel by the receiving device
- Baud rate – Rate at which information is transferred to a communication channel
  Typical baud rates are 9600, 19200, 115200 bits per second
- Baud rate set the same on both transmitting and receiving device
- Baud rate set will be the maximum number of bits that can be transferred per second



UART synchronization

- Does not use a clock signal for synchronization of transmitter and receiver devices
- Both transmitter and receiver generates and receives bitstreams based on their individual clock signals
- Synchronization managed by setting the same baud rate
- Allowable difference in baud rate is upto 10% before timing of bits deviates beyond acceptable limits

UART Data Transmission

- Transmission mode in the form of a packet
- A packet consists of a start bit, data frame, a parity bit, and stop bits
- Start Bit
    - When not transmitting data, UART data transmission line is held at high voltage level
    - To start transfer, transmitting UART pulls the line from high to low for 1 clock cycle
    - When receiving UART detects the high to low voltage transition, it reads the received bits in the data frame at the baud rate

Start Bit (1 bit)	Data Frame (5 to 9 Data Bits)	Parity Bits (0 to 1 bit)	Stop Bits (1 to 2 bits)

UART Data Transmission

- Data Frame
  - Can be 5 bits to 8 bits long if a parity bit is used
  - If no parity bit used, the data frame can be 9 bits long
  - In most cases, data bit has LSB first
- Parity
  - Parity is evenness or oddness of a number
  - Indicates if any data has changed during transmission
  - Bits can be changed by electromagnetic radiation, mismatched baud rates, or long distance transfers
  - After data received, checks if the parity bit (even (0) or odd (1) parity) matches with the data (i.e., even number of 1s or odd number of 1s)
- Stop
  - Signals end of data packet and transmission ends by driving Tx line from a low to high voltage for 1 or 2 bit duration

Start Bit (1 bit)	Data Frame (5 to 9 Data Bits)	Parity Bits (0 to 1 bit)	Stop Bits (1 to 2 bits)

UART Discussion

- Advantages
  - Simple to operate
  - No clock needed
  - Parity bit to allow for error checking
- Disadvantages
  - Size of data frame limited to only 9 bits
  - Cannot use multiple master systems and slaves
  - Low data transmission speeds
  - Baud rate mismatch between Tx and Rx cannot be greater than 10%

Serial Communication using UART

- Serial object used in Arduino for using built-in UART hardware
- **SERIAL.BEGIN()**
  To communicate using the UART interface, it needs to be configured first → Easiest way to configure is by using the function Serial.begin(speed) → speed is the baud rate
- **SERIAL.AVAILABLE()**
  To check if data is waiting to be read in the UART buffer→Returns the number of bytes waiting in the buffer
- **SERIAL.READ()**
  To read the data waiting in the buffer→Returns one byte of data read from the buffer
- **SERIAL.WRITE()**
  To send data via Arduino's TX0 pins, Serial.write(val) is used, where val is the byte to be sent

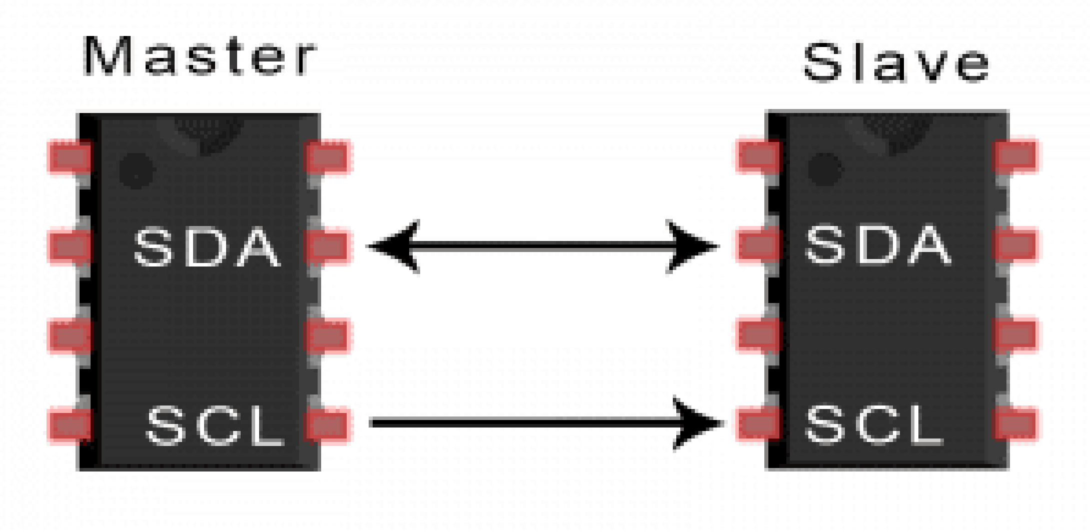
I2C Communication Protocol

- I2C – Inter-Integrated Circuit
- Bus interface connection protocol built into devices for serial communication
- Widely used for short distance communication
- Also known as Two Wired Interface (TWI)
- Combines the best features of SPI and UARTs
- Can connect multiple slaves to single master (like SPI)
- Also multiple masters can control single or multiple slaves

I2C Communication

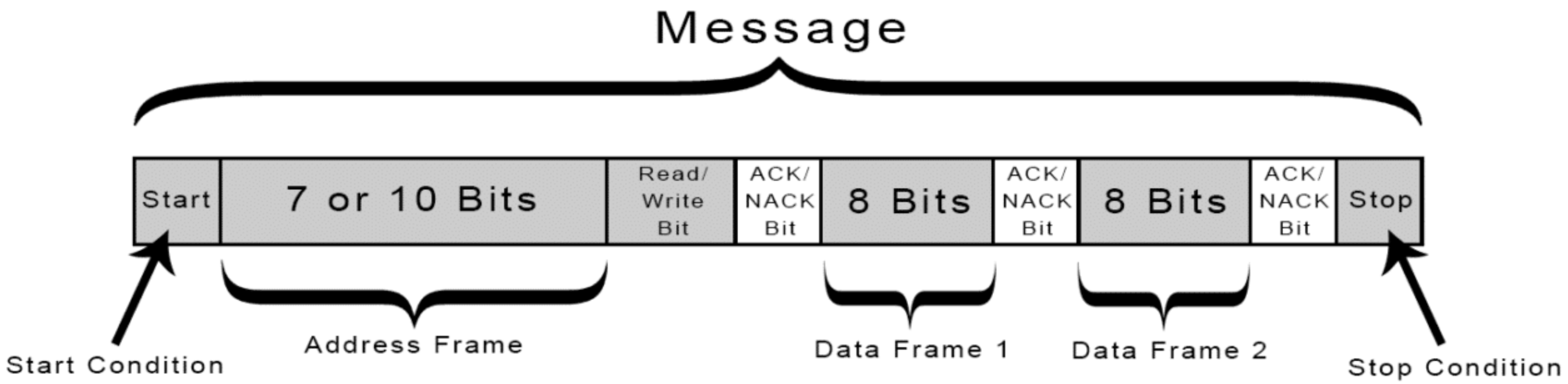
- I2C uses two wires to communicate between devices
- SDA (Serial Data) – Line for master and slave to send and receive data
- SCL (Serial Clock) – Line that carries clock signal
- Synchronous communication – Clock signal is always controlled by the master

Source: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/



Working of I2C

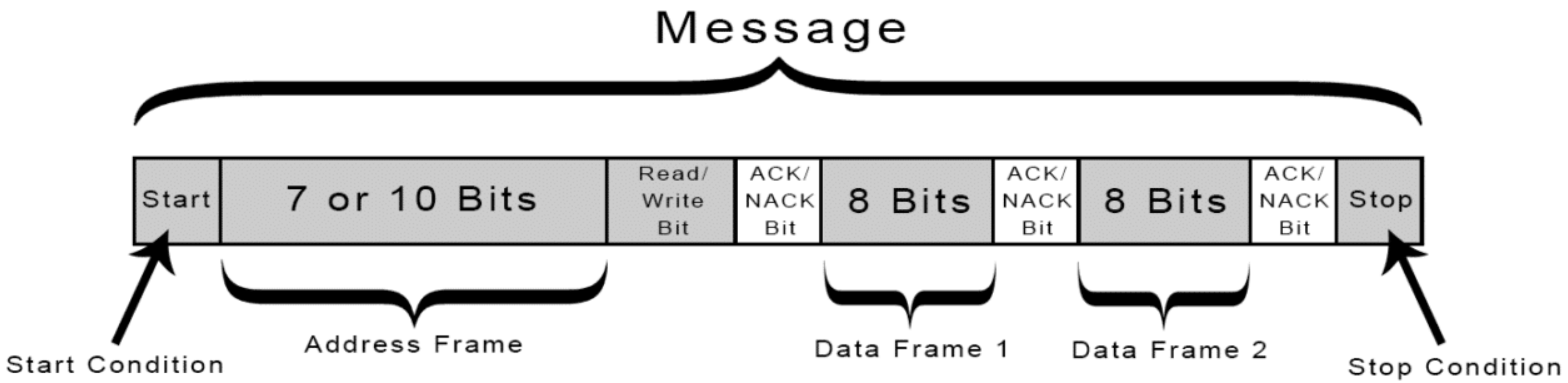
- Data is transferred in messages
- Messages broken up into frames of data
- Each message contains an address frame (binary address of the slave) and one or more data frames
- Also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame



Source: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Working of I2C

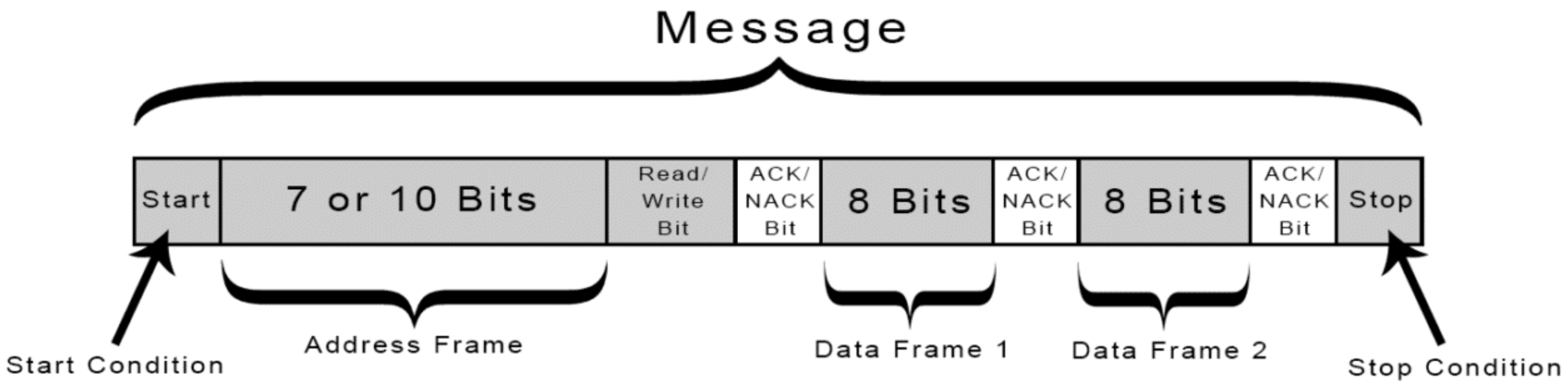
- **Start Condition**: SDA line switches from high to low before SCL line switches from high to low
- **Stop Condition**: SDA line switches from low to high after SCL line switches from low to high
- **Address Frame**: A 7 or 10 bit address unique to each slave
- **Read/Write bit**: Specifies whether master is sending data to the slave (low voltage level) or requesting data from it (high voltage level)
- **ACK/NACK bit**: If an address or data frame was successfully received, an ACK bit is returned to the sender



Source: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Working of I2C - Steps

- Master sends start condition to every connected slave by switching the SDA line from high to low before switching the SCL line from high to low
- Master sends each slave 7 or 10 bit address along with read/write bit
- Each slave compares the address sent with its own address and if matches returns an ACK bit by pulling SDA line low for one bit
- If the address does not match, then slave leaves SDA line high
- Master sends or receives the data frame
- Receiving device returns another ACK bit to the sender after successfully receiving the frame
- To stop, the master sends a stop condition by switching SDA high after switching SCL high.



Source: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

I2C discussion

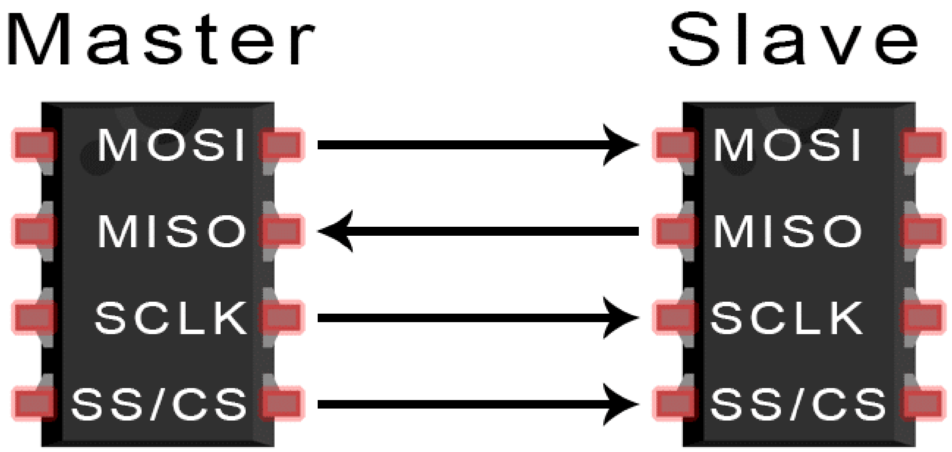
- Advantages
  - Only uses two wires
  - Supports multiple masters and multiple slaves
- Disadvantages
  - Slower data transfer rate than SPI
  - Size of data frame limited to 8 bits
  - More complicated hardware needed to implement than SPI
- Applications – OLED displays, barometric pressure sensors, gyroscope/accelerometer modules interface using I2C.

Wires Used	2
Maximum Speed	Standard mode= 100 kbps
	Fast mode= 400 kbps
	High speed mode= 3.4 Mbps
	Ultra fast mode= 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	Unlimited
Max # of Slaves	1008

SPI communication

- SPI – Serial Peripheral Interface
- It is a “synchronous” data bus → separate lines for data and clock
- Devices communicating via SPI are in a master-slave relation
- MOSI (Master Output/Slave Input) – Line for master to send data to slave
- MISO (Master Input/Slave Output) – Line for slave to send data to master
- SCLK – Clock signal
- SS/CS (Slave Select/Chip Select) – Line for master to select which slave to send data to

Source: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol



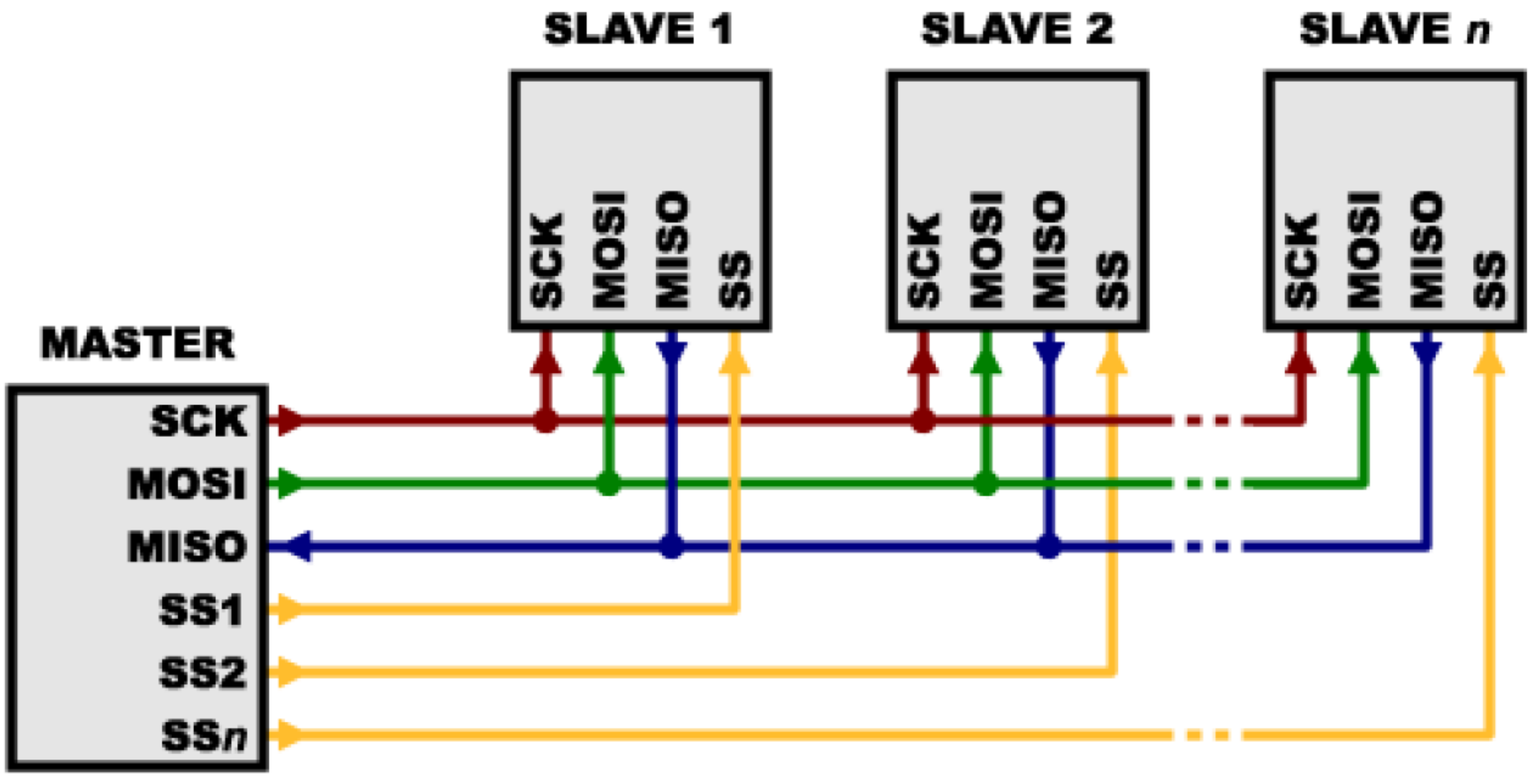
Working of SPI

- Clock signal synchronizes the output of data bits from the master to the sampling by the slave
  - Clock signal can be modified by properties of **clock polarity** and **clock phase**
  - **Clock Polarity** – Set by master for bits to be output and sampled on either the rising or falling edge of the clock
  - **Clock Phase** – Set for output and sampling to occur on either the first edge or second edge of clock cycle
- Master can choose which slave to talk to by setting the slave's CS/SS line to a low voltage
- Master can send data to the slave using the MOSI line, usually sends the MSB first
- Slave can send data to the master through MISO line, usually sends the LSB first

Working of SPI

- Master outputs the clock signal
- Master switches SS/CS pin to a low voltage level
- Master sends data one bit at a time over MOSI line. Slave reads data as they are received.
- For response, slave returns data one bit at a time over MISO line.

Selecting Slave



SPI discussion

- Advantages
  - No start and stop bits, so data can be sent continuously without interruption
  - No complicated slave addressing system like I2C
  - Higher data transfer rate than I2C
  - Separate MISO and MOSI lines, so data can be sent and received at the same time
- Disadvantages
  - Uses four wires (I2C and UARTs use two)
  - No ACK
  - No error checking
  - Only allows single master
- Applications: SD card modules, RFID card reader modules use SPI to communicate with microcontrollers

Questions?