NC STATE UNIVERSITY

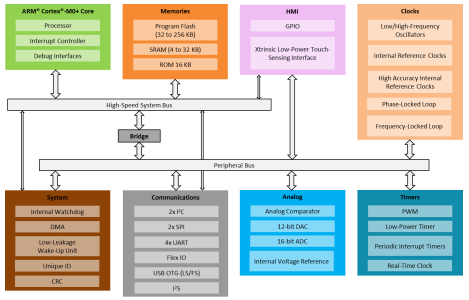
Embedded Communication Networks

Section 1:
Communication Concepts

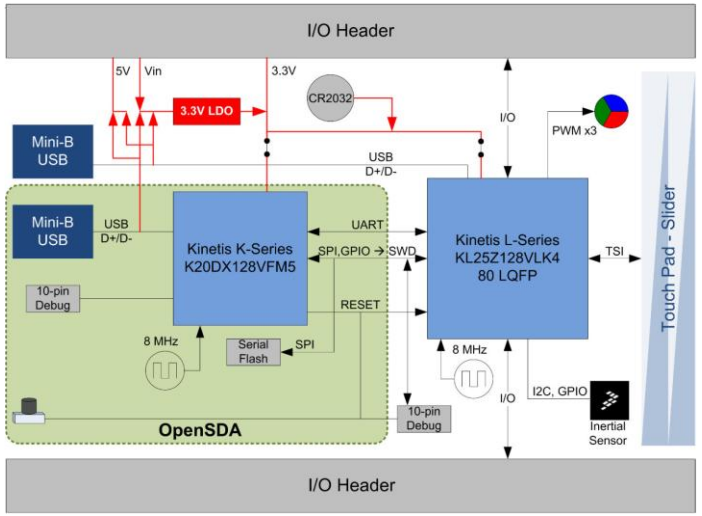
Overview of Embedded System Communications

How far does the message go?

Within Chip



Within Board



Within System

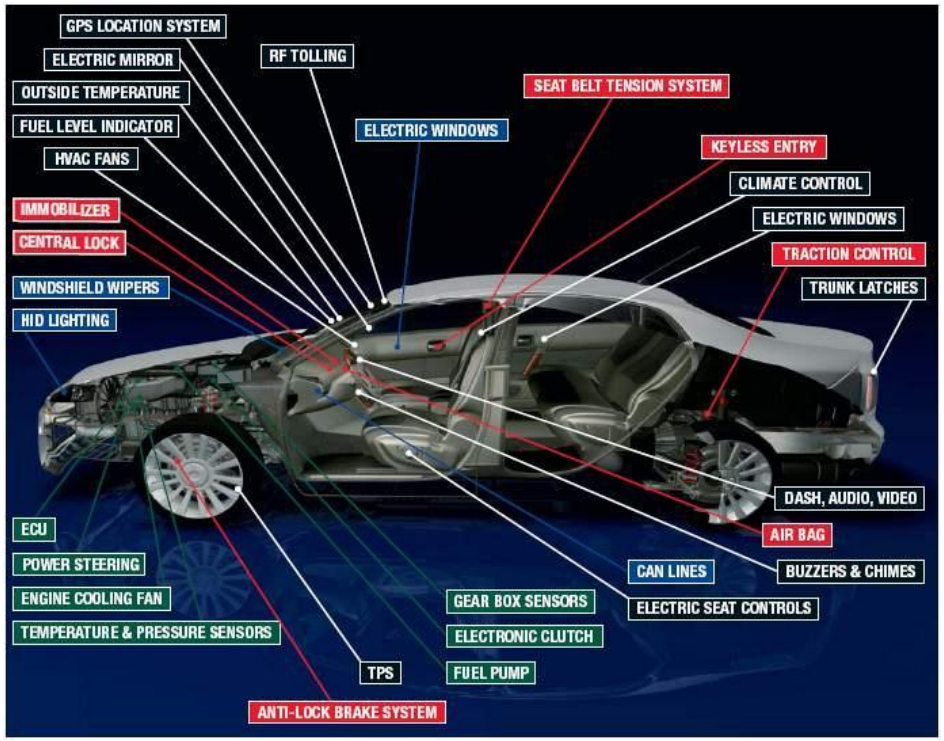


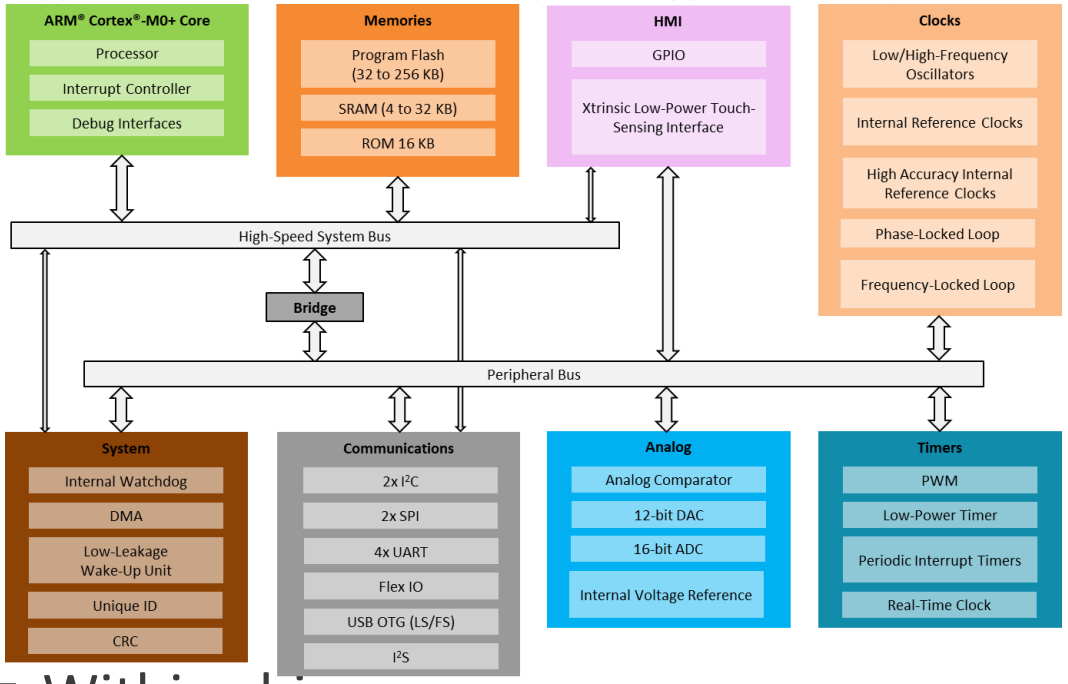
Image courtesy of AVX, Inc.

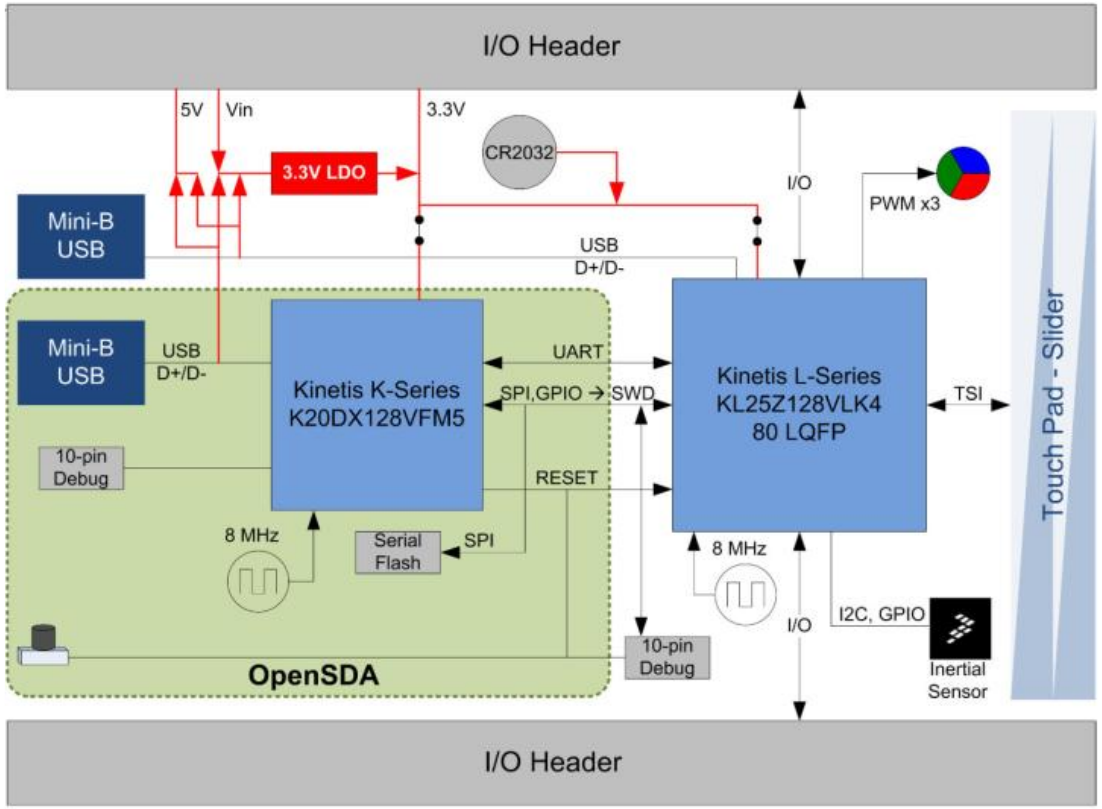
External

Image courtesy of Emerson Electric, Inc.

Communication Scenarios

Different constraints for different system sizes





- Within chip
  - Speed most important
  - Minor area constraints
  - Typical solution: use parallel buses to send data 8, 16 or 32 bits at a time.

- Within board, board-to-board
  - More signals -> more pins on IC package ($$) -> larger, heavier board ($$)
  - If too slow, use parallel bus or wider serial bus, or raise clock speed
  - Typical solutions: serial buses to send data one bit at a time. SPI, I^2S, I^2C, UART

Communication Scenarios

Different constraints for different system sizes

- External – box-to-box, or system-to-system
  - More signals -> larger cable ($$) -> heavier, larger system
  - Going outside the box makes communications more vulnerable to noise
    - Add error control: detection, acknowledgment, correction
  - Typical wired solutions: serial buses to send data one bit at a time.
    - USB (Universal Serial Bus), I^2C, UART, CAN, FlexRay
  - If fast communications are important, use more bits or raise clock speed
    - Ethernet, USB 1.2/2.0, USB 3.0
  - If portability is important, use wireless transmission
    - WiFi (802.11), LTE (cellphone network), 802.15.4

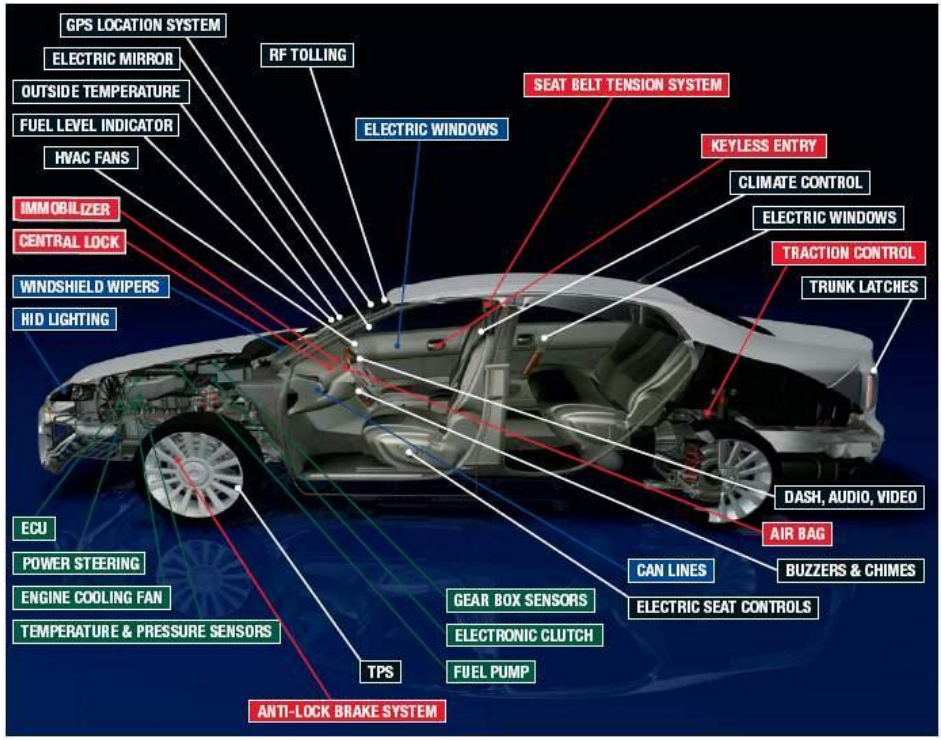


Image courtesy of AVX, Inc.

Image courtesy of Emerson Electric, Inc.

Protocol Stack Concepts

All nodes must follow the same or compatible rules

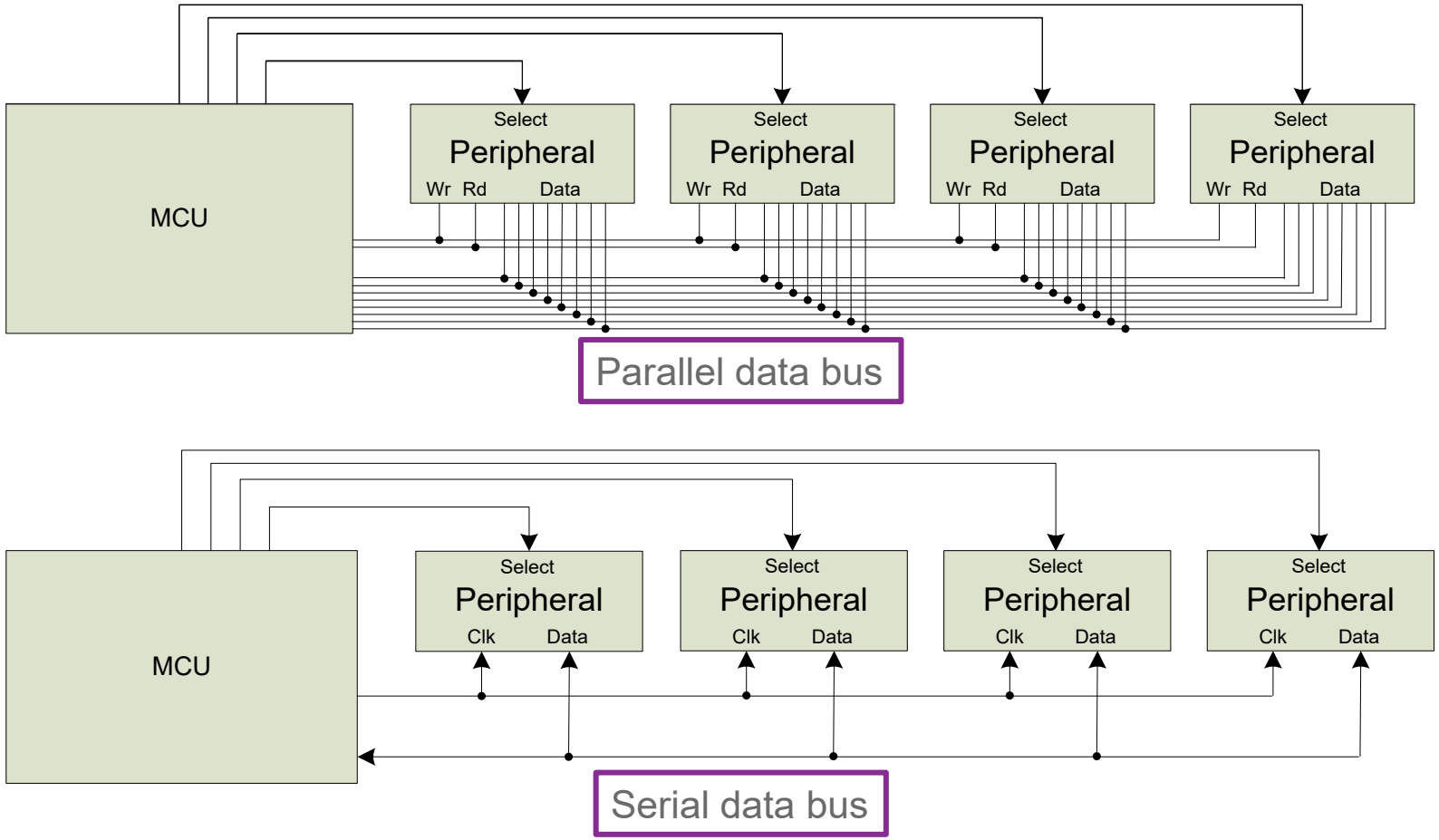
Application
Presentation
Session
Transport
Network
Data Link
Physical

- Helpful to group these rules into layers in a stack
- Example: ***Open System Interconnection (OSI) model***
  1. ***Physical layer:*** Defines how 1s and 0s are represented. Voltage, current, electromagnetic field, light. Amplitude, duration, etc.
  2. ***Data Link layer:*** Has two layers
     - ***Media Access Control layer:*** How nodes share the communication medium. When does a node get to talk on the wire?
     - ***Logical Link Control layer:*** How data is framed how receiver is synchronized (when does the data start?), and how errors are detected
  3. ***Network layer:*** How to route data between nodes, including addressing, handling data too large to fit into one packet, congestion control, and error handling
  4. ***Transport layer:*** How to provide complete, correct data transfer between nodes (called hosts)
  5. ***Session layer:*** How to provide connections between application programs on different nodes
  6. ***Presentation layer:*** Translates data (e.g. encryption and decryption)
  7. ***Application layer:*** Consists of application programs
- OSI model defined for large networks of computing systems (e.g. Internet), not targeted to embedded systems
- Protocols for embedded systems often merge or omit layers/features if not needed

Communication Basics

Start with the foundation – the physical layer

- Communication systems usually ***serialize*** data
  - Don’t send all the bits at once
- Why?
  - Reduce number of data signals needed (1, 2, or 4 vs. 8, 32)
  - Reduces package or connector size, weight
  - Simplifies circuit design

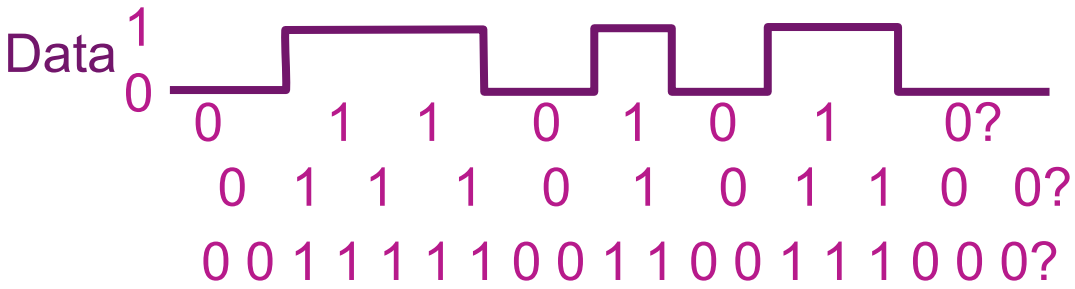


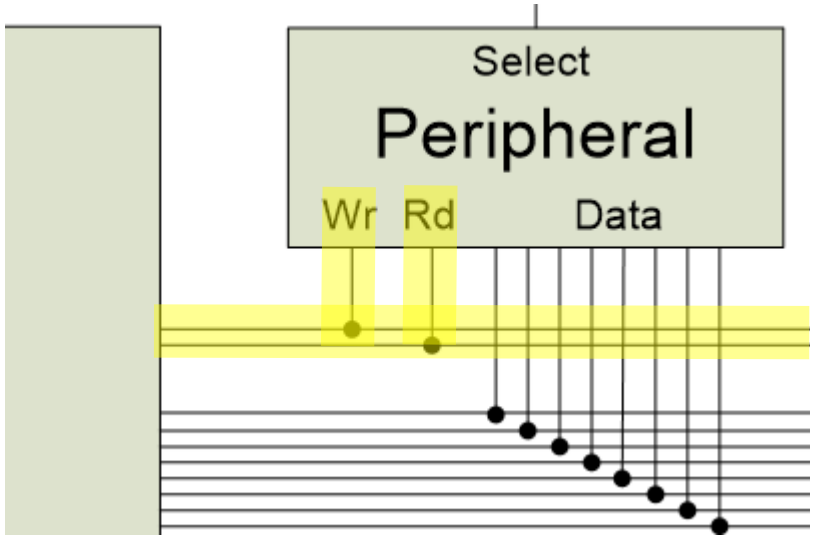
When is the Data Valid?

***When** should receiver sample the data? Data link layer*

- Two approaches
  - **Synchronous**: Transmitter sends a control signal to tell receiver when to sample data
  - **Asynchronous**: Receiver has to determine when data is valid
- **Parallel** communications typically use synchronous communication
  - Already have multiple signals for data (8, 16, 32)
  - Can usually afford an extra signal or two (e.g. Write Enable (Wr), Read Enable(Rd))
- **Serial** communications
  - Goal is to reduce number of signals, so synchronous is less attractive because of extra signals

What does this bit stream mean?

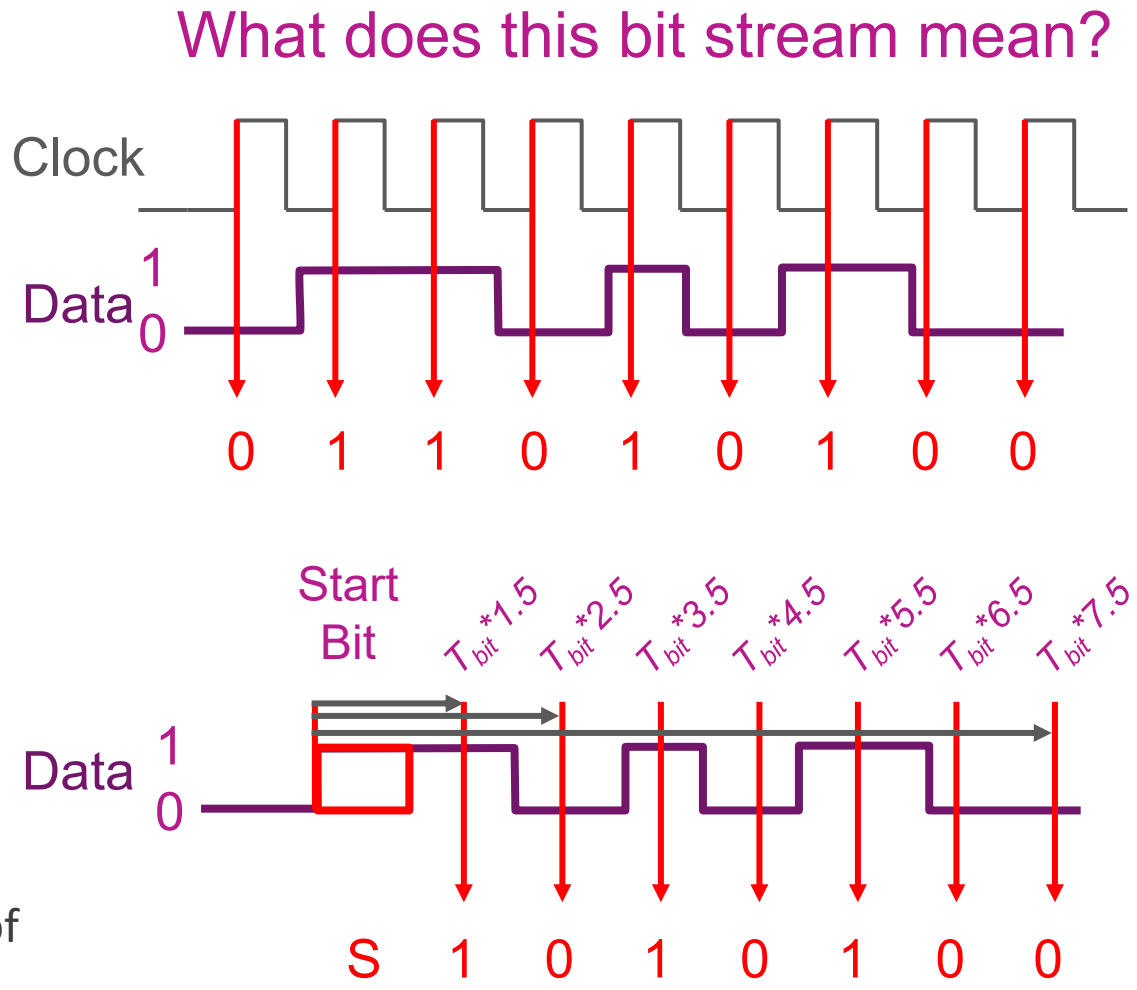




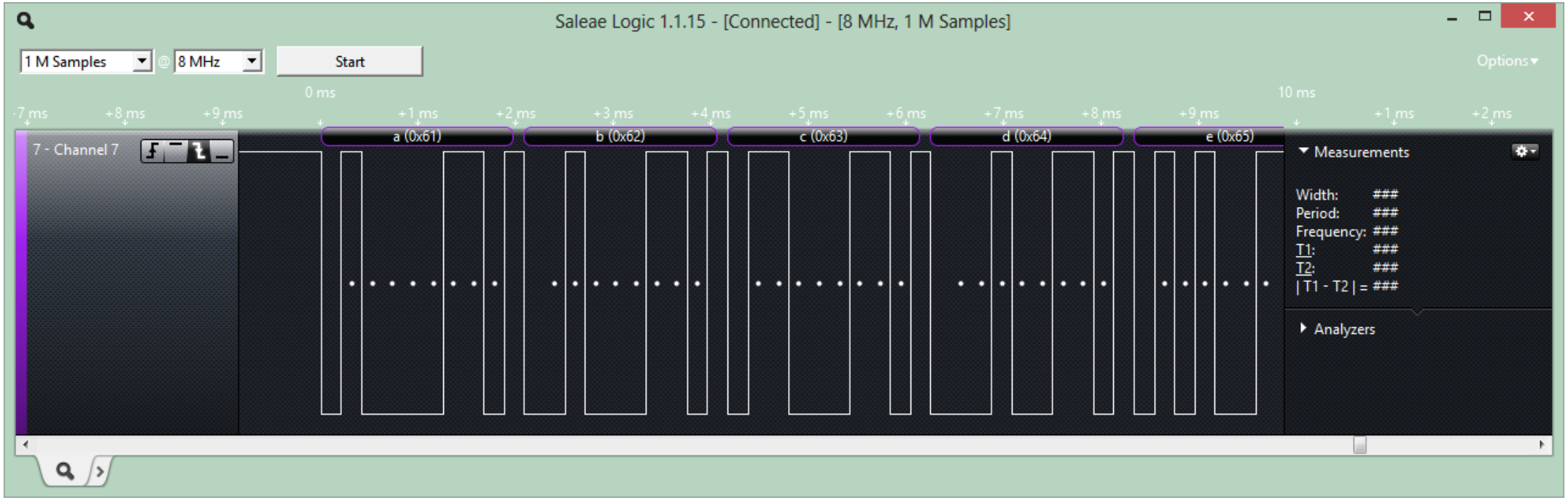
Sampling Data

- Synchronous
  - Use separate clock signal to define bit times
  - Example: Sample data on clock's rising edge

- Asynchronous
  - Infer bit times based on ***fixed delays*** from ***reference event***
  - Example
    - Reference event is leading edge of start bit (0 to 1 transition)
    - Sample input data at n+1/2 bit times after beginning of start bit



Tools for Serial Communications Development



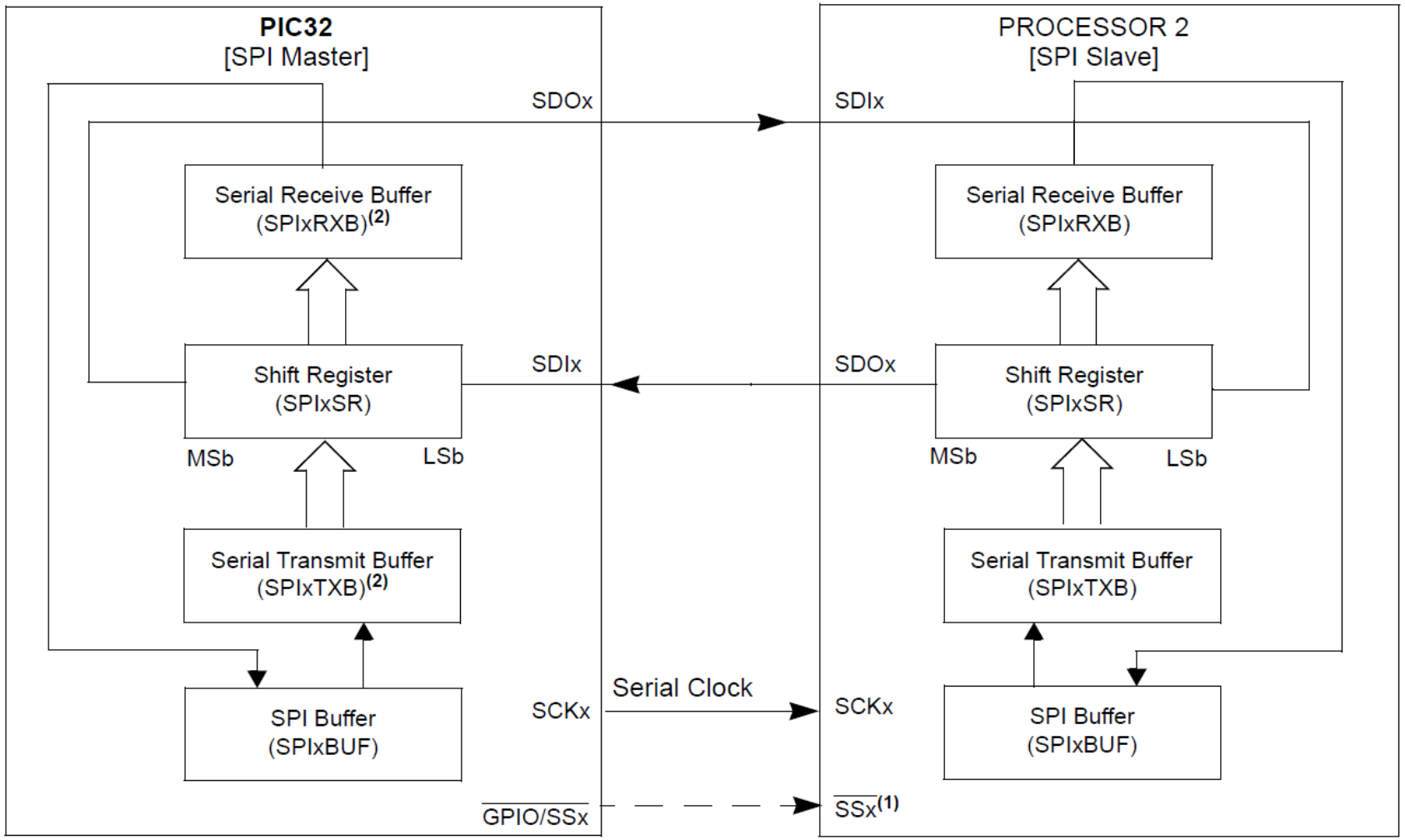
- Tedious and slow to debug serial protocols with just an oscilloscope
- Instead use a logic analyzer to decode bus traffic
- Worth its weight in gold!

- Saelae 8-Channel Logic Analyzer
  - $150 (www.saelae.com)
  - Plugs into PC's USB port
  - Decodes SPI, asynchronous serial, I^2C, 1-Wire, CAN, etc.

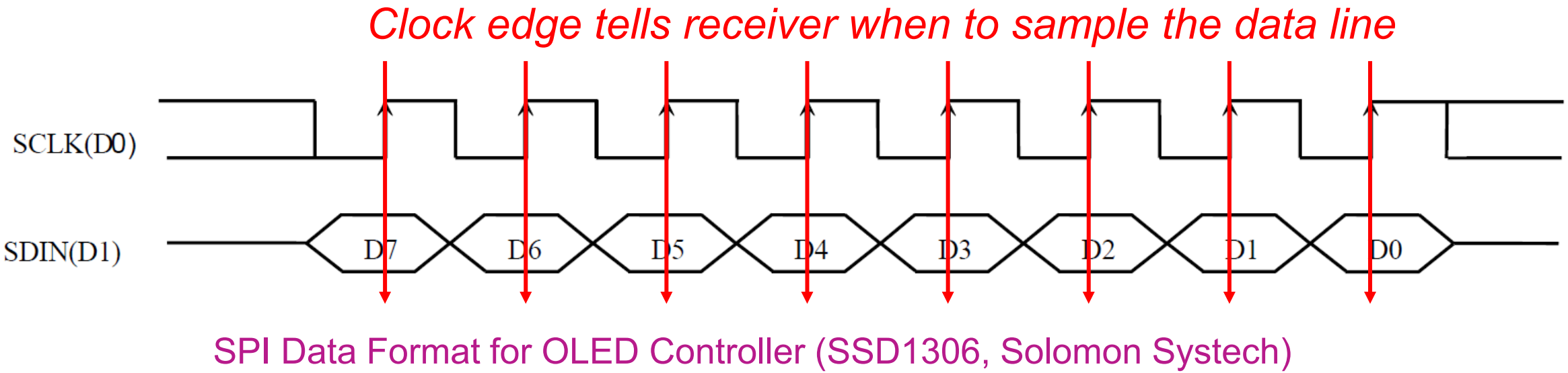
Section 2:
Serial Peripheral Interconnect (SPI)

SPI Basics

- "Ring of shift registers" which exchange data
- Master device generates clock signal (SCKx) which…
  - Shifts data from master to slave one bit at a time
  - Shifts data from slave to master one bit at a time
- Optional Slave Select signal (SSx)
  - Used to identify which slave is being accessed
- SPI defines parts of physical and data link layers



SPI Data Transmission



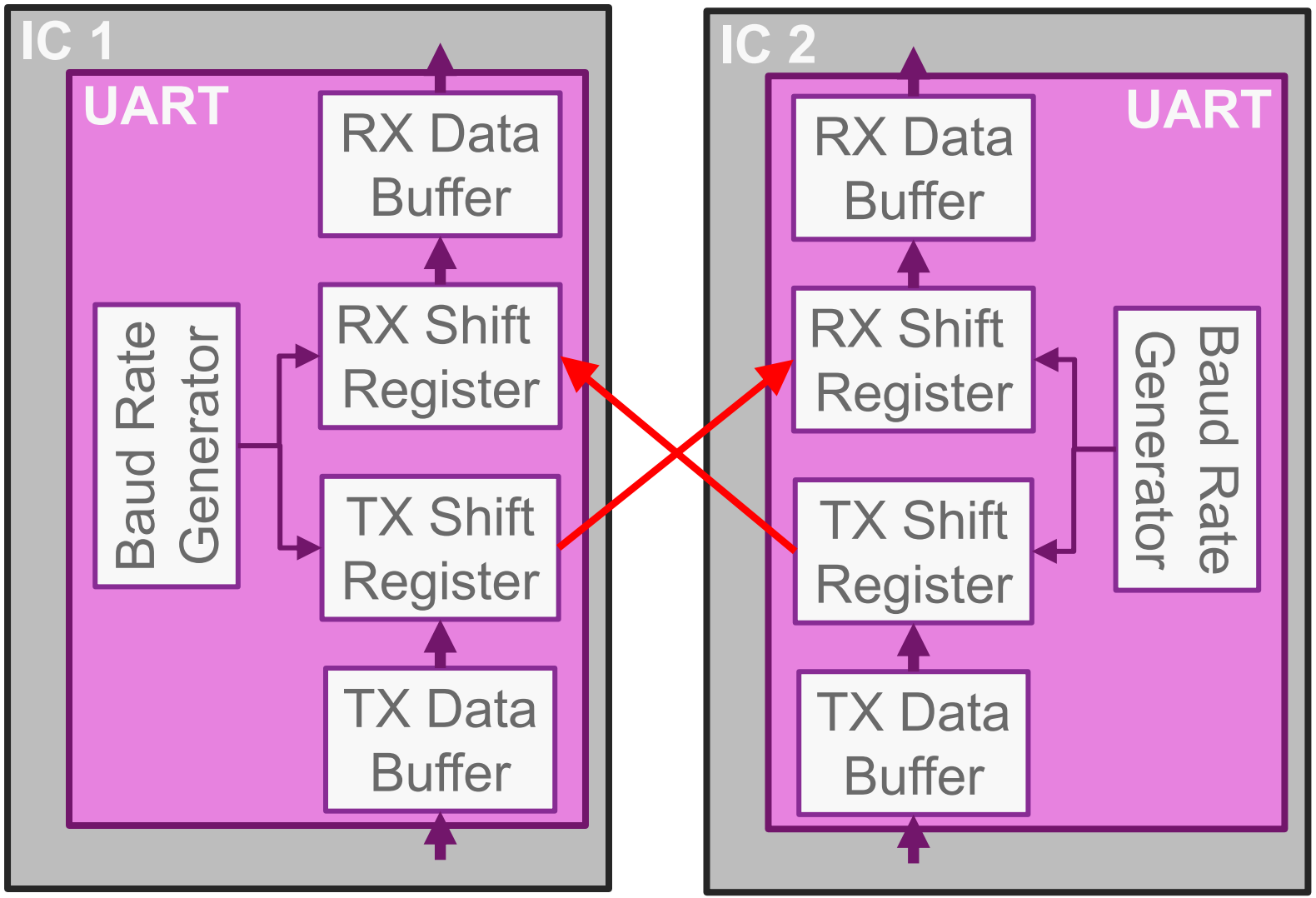
SPI Data Format for OLED Controller (SSD1306, Solomon Systech)

- Clock signal SCKx
  - Generated by master
  - Defines communication timing
  - SCKx generated by SPI module's baud rate generator by dividing down a reference clock
- Data signal
  - Generated by master on data output pin SDOx
  - D7 (most-significant bit) sent first
  - Sampled by slave when clock rises*
    - * Other versions of SPI use falling clock edge

Section 4:
Asynchronous Communication

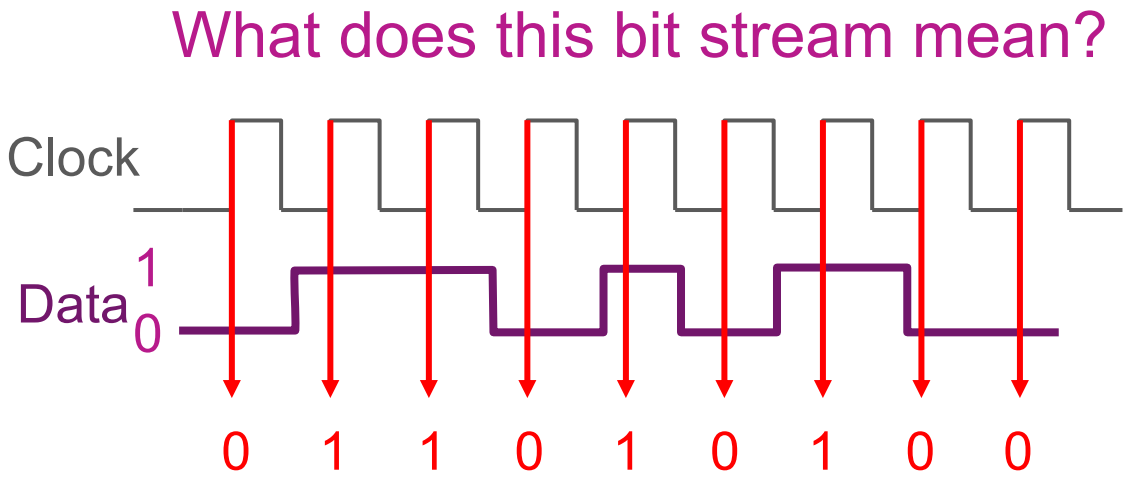
Asynchronous Serial Basics

- Similar to SPI, but no external clock signal used
- Peripheral is called a ***UART***
  - Universal = configurable
  - Asynchronous = no clock signal used for communication
  - Receiver/Transmitter = contains both receiver and transmitter
- Defines parts of physical and data link layers

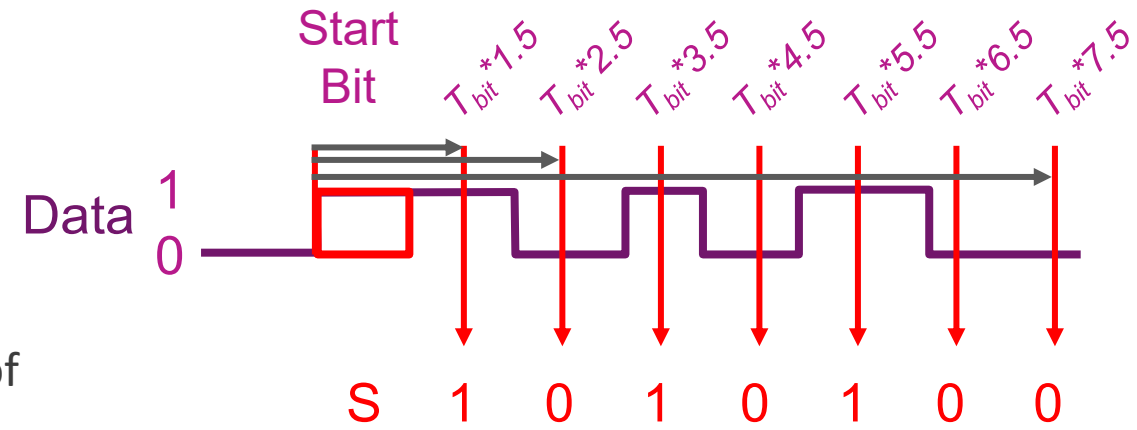


Reminder: Sampling Data

- Synchronous
  - Use separate clock signal to define bit times
  - Example: Sample data on clock's rising edge

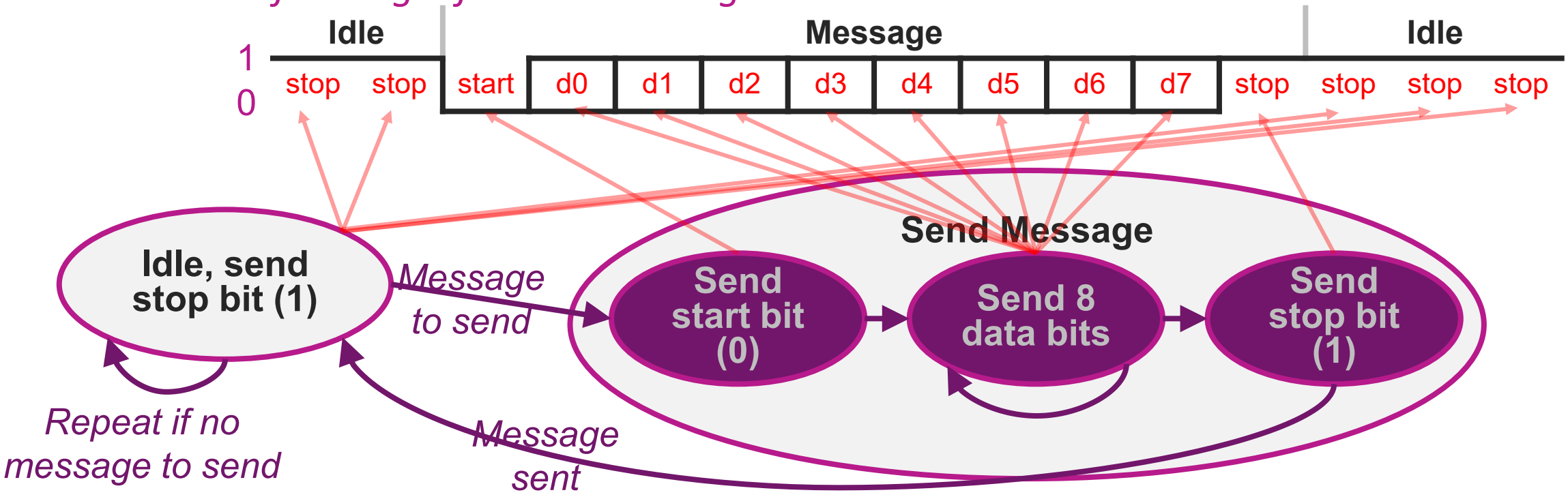


- Asynchronous
  - Infer bit times based on ***fixed delays*** from ***reference event***
  - Example
    - Reference event is leading edge of start bit (0 to 1 transition)
    - Sample input data at n+1/2 bit times after beginning of start bit



Framing

Transmitter inserts framing information to signal data start and end



- Transmitter in Idle state:
  - Send another stop bit (1)
- Transmitter in Send Message state
  - Send Start bit (0),
  - Send data bits, starting with LSB
  - Send Stop bit (1)

Major Asynchronous Communication Options

- May have 1 or 2 stop bits
- May have 7, 8 or 9 data bits

- Transmitter may add **parity bit** for error detection
  - Using Even Parity? Set parity bit to make total number of 1s **even**.
  - Using Odd Parity? Make total number of 1s **odd**.
    - Receiver calculates parity based on received data bits and parity bit (not start or stop bits)
    - If parity doesn't match specification (even or odd), then signal an error
  - Can detect an odd number of bit errors, but not an even number

Data		# of 1's	Parity bit for…	
Hex	Binary		Even Parity	Odd Parity
00	0000 0000	0	0	1
3f	0111 1111	7	1	0
a5	1010 0101	4	0	1
16	0001 0110	3	1	0

Section 7:
Advanced Communication Concepts and Example Protocols

Review: Protocol Stack Concepts

All nodes must follow the same or compatible rules

Application
Presentation
Session
Transport
Network
Data Link
Physical

- Helpful to group these rules into layers in a stack
- Example: ***Open System Interconnection (OSI) model***
  1. ***Physical layer:*** Defines how 1s and 0s are represented. Voltage, current, electromagnetic field, light. Amplitude, duration, etc.
  2. ***Data Link layer:*** Has two layers
     - ***Media Access Control layer:*** How nodes share the communication medium. When does a node get to talk on the wire?
     - ***Logical Link Control layer:*** How data is framed, how receiver is synchronized (when does the data start?), and how errors are detected
  3. ***Network layer:*** How to route data between nodes, including addressing, handling data too large to fit into one packet, congestion control, and error handling
  4. ***Transport layer:*** How to provide complete, correct data transfer between nodes (called hosts)
  5. ***Session layer:*** How to provide connections between application programs on different nodes
  6. ***Presentation layer:*** Translates data (e.g. encryption and decryption)
  7. ***Application layer:*** Consists of application programs
- OSI model defined for large networks of computing systems (e.g. Internet), not targeted to embedded systems
- Protocols for embedded systems often merge or omit layers/features if not needed

Key Concepts

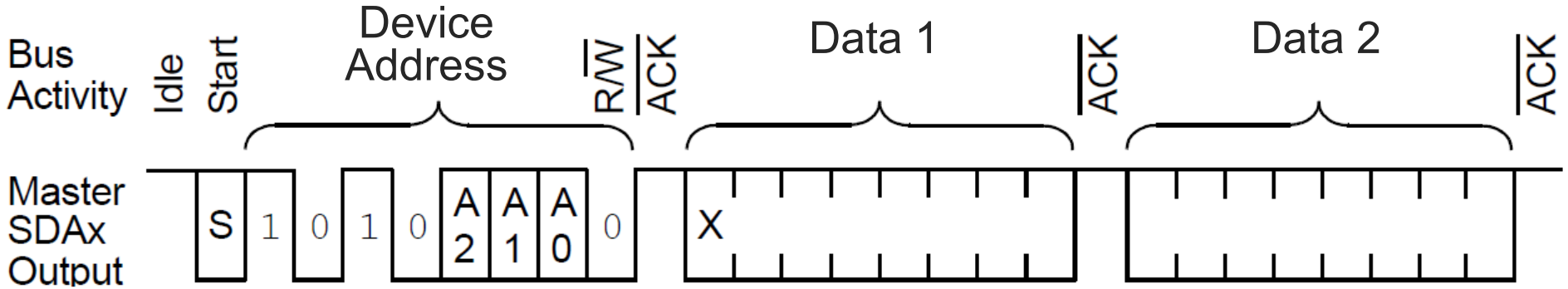
- How do we detect data transmission errors?
- What's in a message besides data?
- How can we make the communication system scale up to large sizes easily?
- How can we increase the communication speed?
- How can we transmit data wirelessly?

How Do We Detect Errors?

- Approach
  - Transmitter sends extra error-detection information along with data
  - Receiver ***recalculates*** the error-detection information based on received data,
  - Receiver ***compares*** recalculated version with received version
  - If these don't match, then the message was corrupted and should be discarded
- Examples:
  - Parity: There is an odd number of ones in this message
  - Checksum: If you add up all the bytes in this message, the sum ends in 0x38
  - CRC: If you process all the bytes in this message this way (e.g. by shifting and exclusive-oring them together), the result ends in 0x68
- Acknowledgements used?
  - Positive: Transmitter expects OK response, else it retransmits
  - Negative: Transmitter only retransmits if it gets an error response
- Multiple receivers, or even broadcast?
  - Common for distributed system to share system state (e.g. vehicle speed)
  - Want message to be received correctly by ALL nodes, or NONE. Not just some.
  - Need quick way for any receiver to be able to force retransmission by sending negative acknowledgement

What's in a Message Besides Data?

There's more than just data



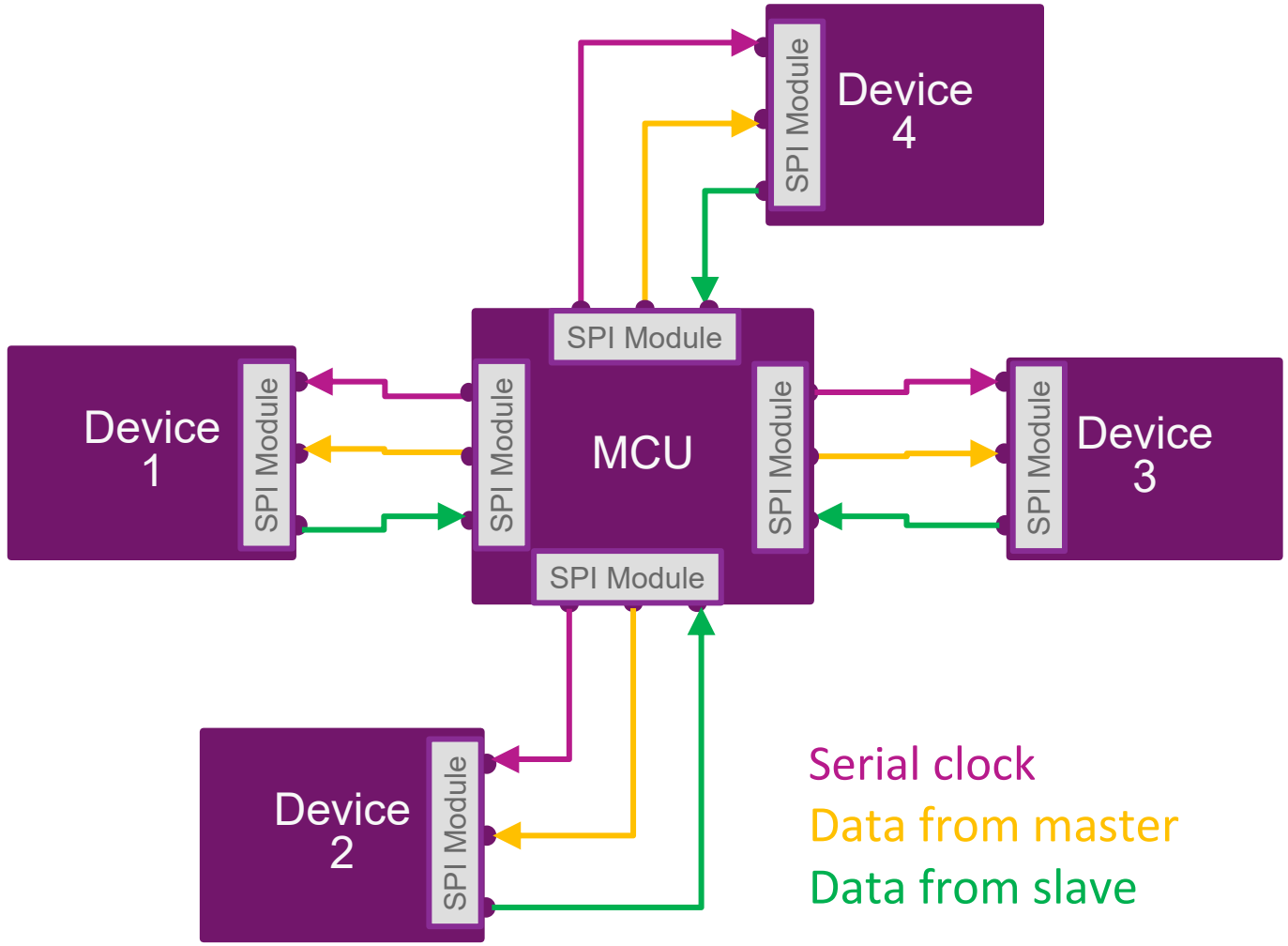
Example of I²C message

- Message holds data and other information
- Data: May multiple bytes per message
- Other information
  - Framing information: When message starts, stops
  - Error detection information: Parity or CRC
  - Acknowledgement: Received correctly?
  - More (discussed soon): Device address, operation (read, write), data length, etc.
- Example: I²C (Inter-integrated circuit bus)
  - Start condition
  - Device address
  - Read or write command
  - Acknowledgement(s) from slave
  - Multiple bytes of data

How Can we Make Scaling Up Easier?

Supporting many communication devices

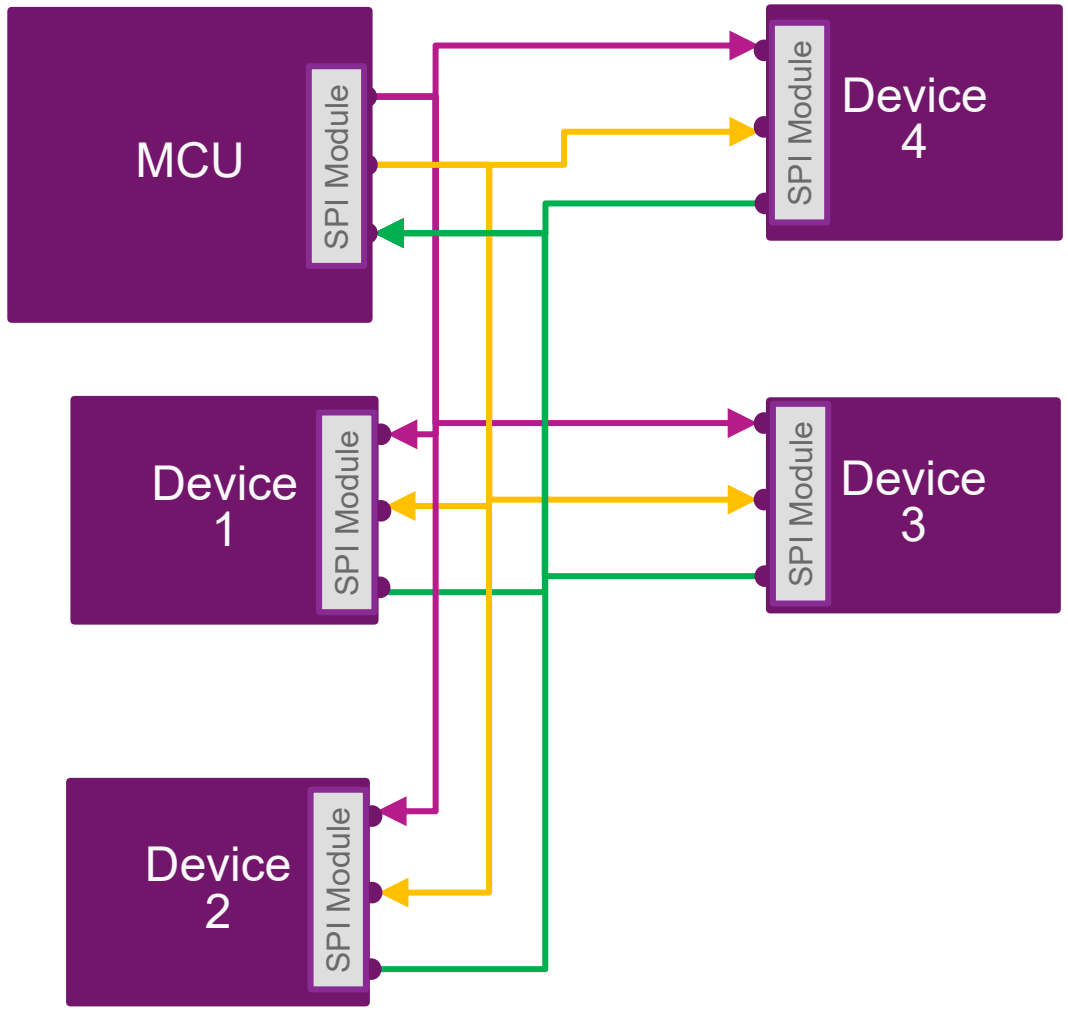
- Dedicated communication links
  - Each pair of communicating devices has a dedicated set of wires and a pair of communication modules
- Problem
  - We need many ports and sets of wires to talk with multiple devices
  - Doesn't work well for systems with many devices
- Instead, can multiple devices ***share*** same the communication signals and wires?



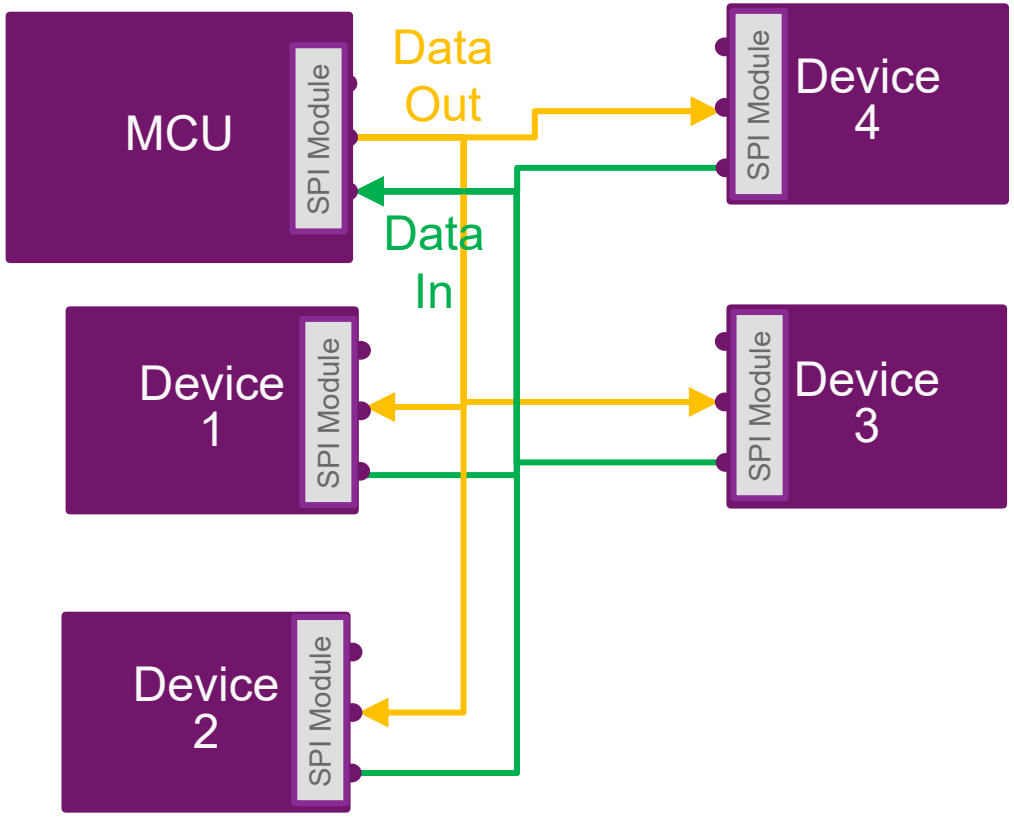
Sharing: Who Gets to Talk When?

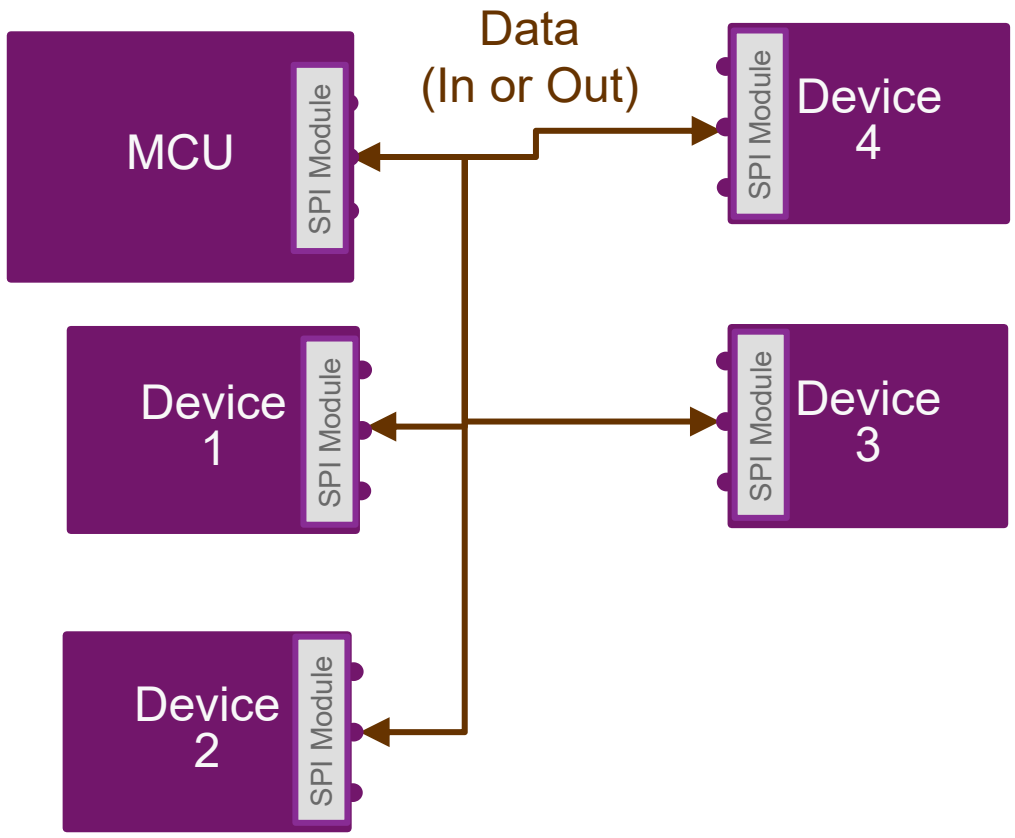
Data link layer and media access control

- Now each node needs ***just one*** communication interface
- All nodes share the bus (communication medium)
- Need a ***media access control*** method
  - Determines which node talks when
- Categories
  - Master/Slave: master tells each node when it can talk
  - Multiple access: no master needed, nodes decide on their own when to talk



Duplexing

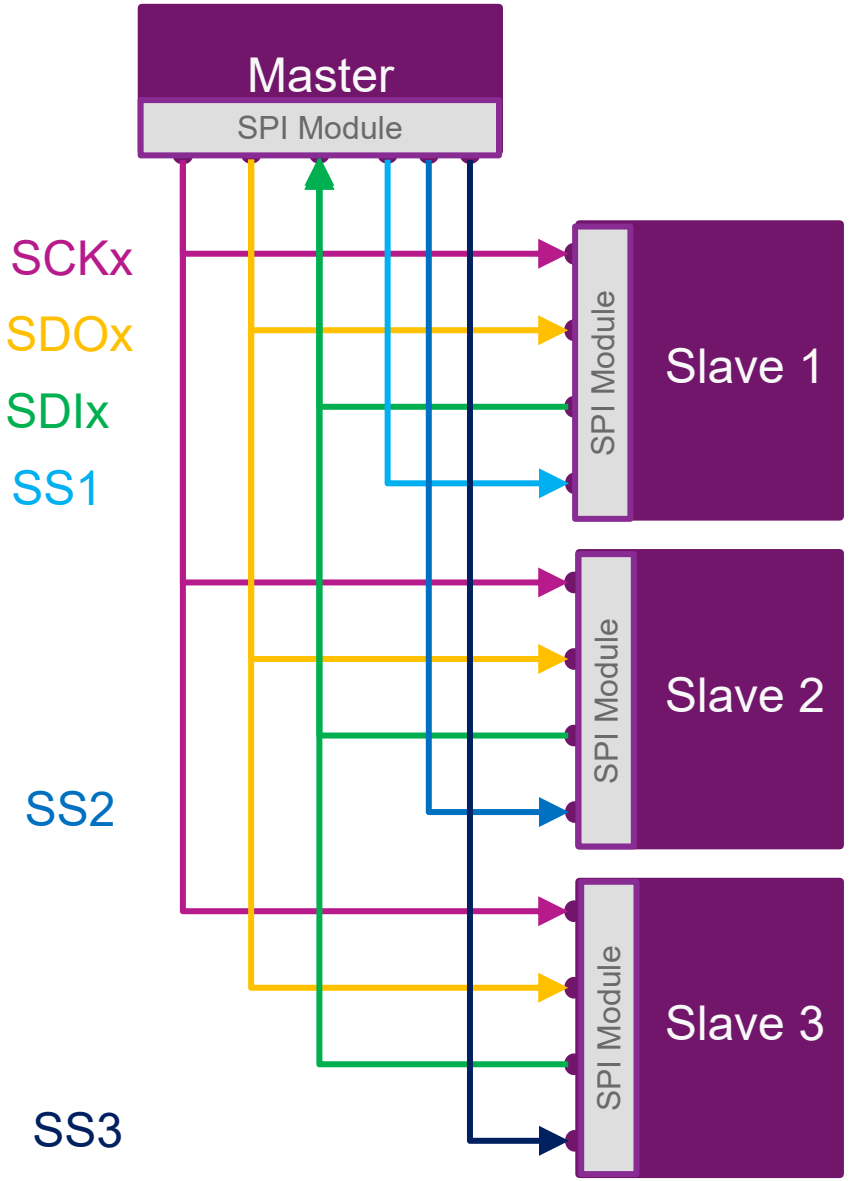




- Full-Duplex
  - Node can transmit and receive simultaneously
- Half-Duplex
  - Node cannot transmit and receive simultaneously

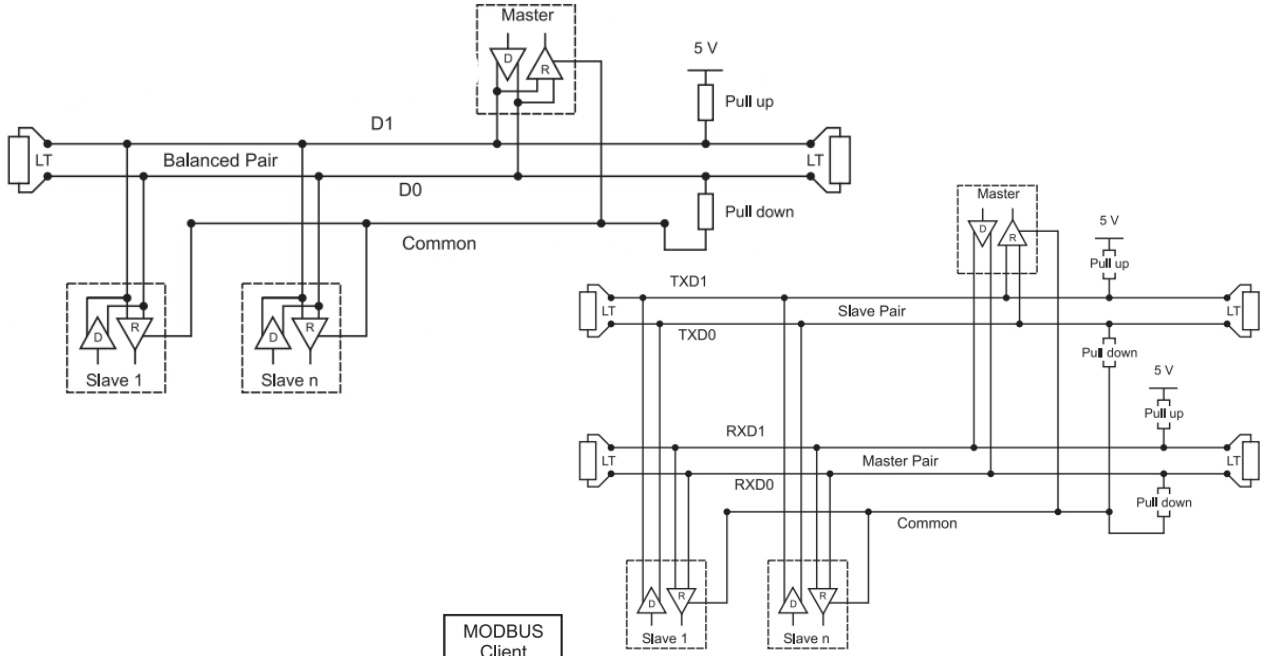
Master/Slave with Select Signals

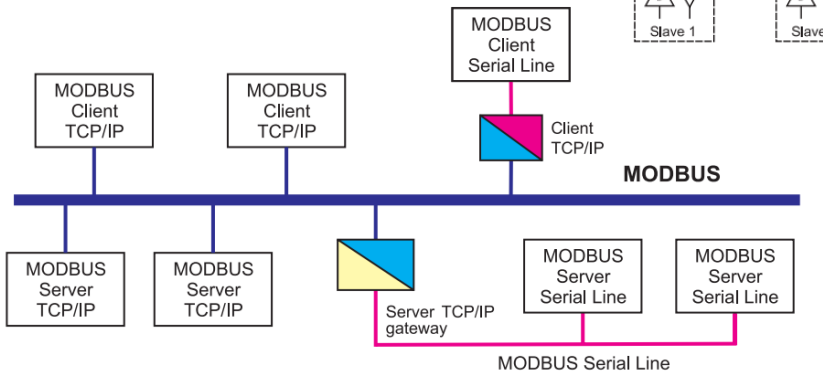
- Multiple SPI slave devices can share clock and data lines
- Master only needs one SPI module
- Select slave by asserting its slave select line (SS1, SS2, SS3)
- Only one slave select line will be active at a time



MODBUS: Master/Slave

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 bytes(s)	2 bytes CRC Low | CRC Hi

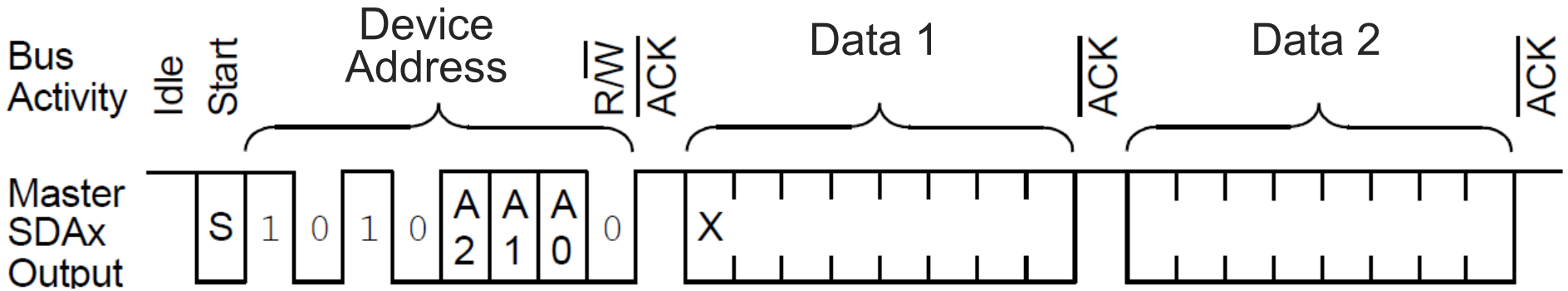




- Master periodically requests data from slave
- Message
  - Slave device address (0: master broadcast, no response)
  - Function Code
    - [Read or write] [one or multiple data items] [single bit or 16-bit word]. Plus others, and user-defined
  - CRC/LRC used to detect errors
- Data model: “registers”
  - Coils: Binary outputs
  - Input bits: Binary inputs (read-only)
  - Input registers: analog inputs (read-only)
  - Holding registers: modifiable analog parameters
- Versions and variations
  - Serial
    - ASCII (text data) vs. RTU (binary data)
    - Point-to-Point (EIA-232C) vs. Multipoint (EIA-485)
    - Half-duplex vs. full-duplex
  - TCP/IP (on Ethernet)

Source: Contemporary Controls, https://www.ccontrols.com/pdf/Extv9n5.pdf

I²C: Master/Slave with Address in Message



Example of I²C message

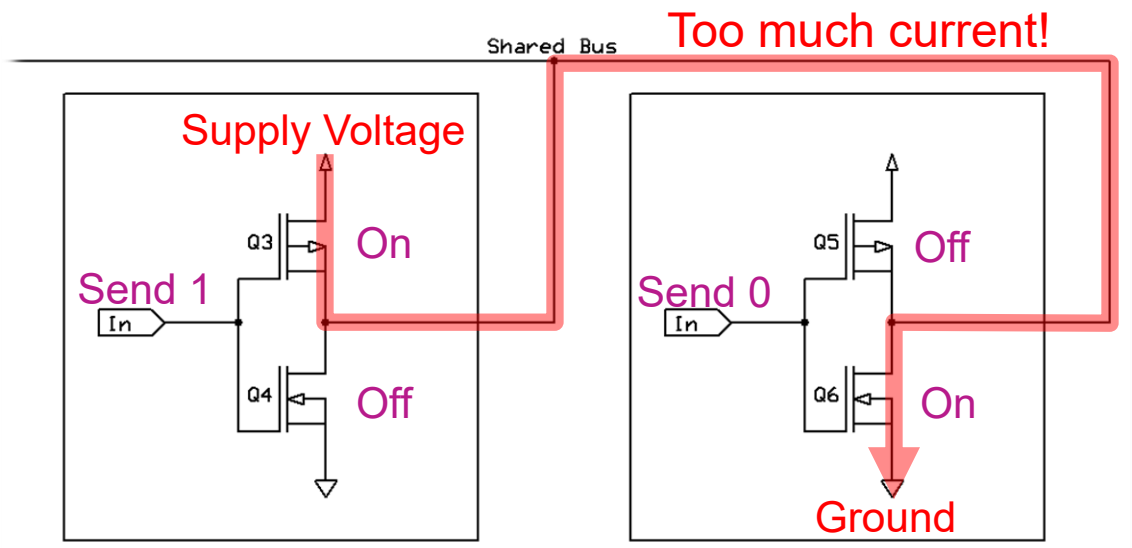
- Use data line for both address and data
- Protocol *explicitly defines* address location in message
- Master sends slave device address
- Data is sent by master (if write operation) or slave (if read operation)
- Slave device only processes messages with its own address, ignores other messages
- Typically also have broadcast address to send data to all slaves

Protocols Based on Asynchronous Serial

	RS232	RS422	RS485
Signals	Single-ended	Differential	Differential
Data Direction	One-way	One-way	Two-way
Multidrop	No	Yes	Yes
Maximum Transmitters	1	1	32
Maximum Receivers	1	10	32
Maximum Data Rate over 50 ft.	20 kpbs	10 Mbps	10 Mbps
Maximum Cable Length	50 ft	4000 ft	4000 ft

Getting Rid of the Master

- For some systems, not practical to have one node coordinate all communications
  - Slaves usually wait longer than necessary to send data
  - Throughput limited by overhead of coordinating communications
  - Hard to write and maintain software for master to control everything
- Can we make a network with ***no master***?
  - When does a node get to transmit, if there is no master giving permission?



Short circuit – Transistors Q3 and Q6 may overheat and fail

- If multiple nodes transmit at same time, messages will ***collide*** on bus
  - One or both messages will be corrupted
  - Circuits may also be damaged if not designed correctly to handle conflicts
    - "1!" + "No, 0!" = short circuit and burned out transistors

Lossy Arbitration

Try and hope for the best

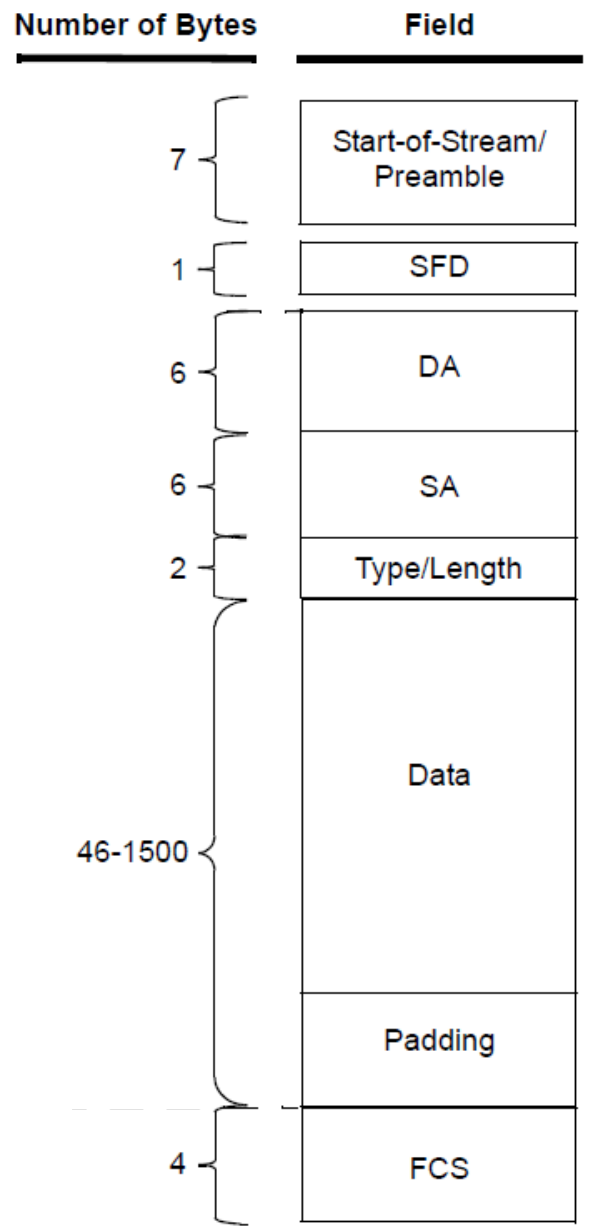
- Basic approach for node with data to send
  - Wait for bus to be idle
  - Send data
- Called CSMA (carrier sense multiple access):
- Works well with light traffic, but doesn't scale up well due to collisions
  - Collision → retransmission attempts → more collisions → more retransmission attempts
  - Available throughput is fraction of bit rate

- Improve by letting transmitters detect collisions
  - Monitor bus signal with *Collision Detection* circuit to compare bus against what was expected (CSMA/CD)
  - Give up transmission as soon as collision is detected
  - After a collision, wait a random time before trying to transmit again
  - If another collision occurs, repeat but double the maximum random time (to reduce odds of another collision)

Example Protocol: Ethernet

IEEE 802.3

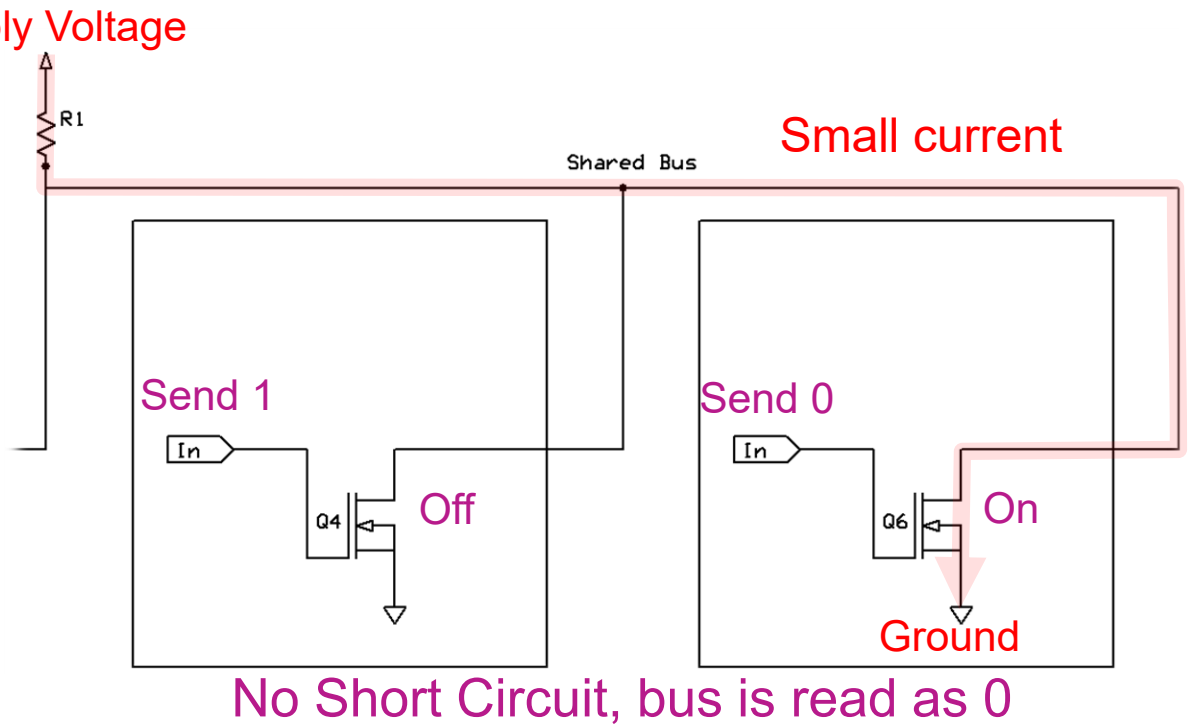
- Key features
  - 10 Mbps/100 Mbps/1 Gbps/10 Gbps data rates
  - 48 bit (6 byte) addresses for source, destination
  - Broadcast, multicast, and unicast message addressing
  - 46 to 1500 bytes of data per message
  - 32-bit CRC (frame check sequence) for error detection
- Improvements
  - Standard Ethernet: Devices share the media (wires) by connecting to a hub.
  - Switched Ethernet: Devices don't share the media. Switch (not hub) has an Ethernet controller and dedicated media for each device.
    - Raises throughput and eliminates collisions



Use Bit Dominance to get Lossless Arbitration

Messages never collide

- Want a system where messages never collide
  - Never lose messages (even with heavy traffic)
- Start with circuit providing ***bit dominance***
  - Example: 0 dominates 1
  - If any node is transmitting a 0, that overrides any nodes transmitting a 1.
    - All nodes on the bus see the 0 (even the nodes trying to transmit a 1)
  - Simple circuit (open drain with pull-up)
- Can now design protocol to prioritize access using this feature



Binary Countdown

Uses bit dominance

Transmitter loses arbitration: tries to send 1, but sees 0. Gives up.

Node B wins arbitration, keeps transmitting

	Priority				Data			
	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 8
Node A: Priority 4 (1000)	1	*0*	*0*	*0*	*1*	*1*	*1*	*0*
Node B: Priority 1 (0011)	0	0	1	1	0	1	0	1
Node C: Priority 2 (0100)	0	1	*0*	*0*	*1*	*1*	*0*	*0*
Bus	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**1**

- Each message starts with a priority field
  - Used to arbitrate medium access
  - Can be transmitter address, receiver address, message type, etc.
- Basic approach for node to transmit a message
  - Wait for idle bus
  - Start transmitting message
    - If bus value doesn't match what is being sent, then another node is sending a higher priority message, so stop transmitting immediately. Go to start again (wait for idle bus)
  - If able to transmit priority field without any bus mismatches, then have won arbitration and can continue to send rest of message

Binary Countdown

Uses bit dominance

Bus doesn't match node's data, so have lost arbitration. Stop transmitting, retry later.

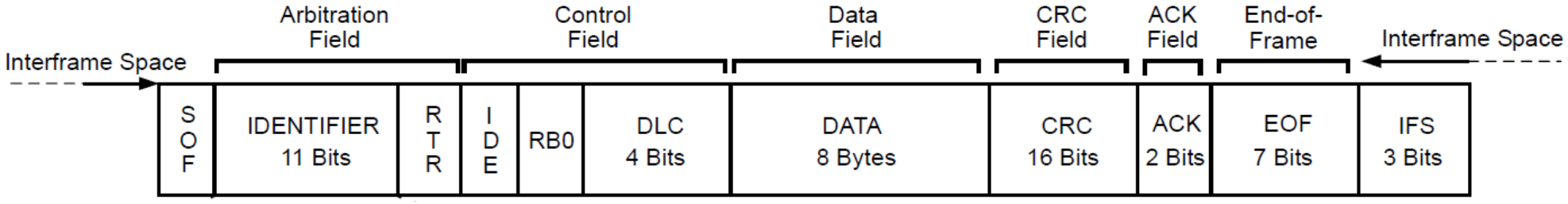
	Priority			Data			
	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
Node A: Priority 4 (100)	1	*0*	*0*	*1*	*1*	*1*	*0*
Node B: Priority 1 (001)	0	0	1	0	1	0	1
Node C: Priority 2 (010)	0	1	*0*	*1*	*1*	*0*	*0*
Bus	**0**	**0**	**1**	**0**	**1**	**0**	**1**

Node B wins arbitration, can transmit rest of its message.

- Each message starts with a priority field
  - Used to arbitrate medium access
  - Can be transmitter address, receiver address, message type, etc.
- Basic approach for node to transmit a message
  - Wait for idle bus
  - Start transmitting message
    - If bus value doesn't match what is being sent, then another node is sending a higher priority message, so stop transmitting immediately. Go to start again (wait for idle bus)
  - If able to transmit priority field without any bus mismatches, then have won arbitration and can continue to send rest of message

Example Protocol: CAN

Controller Area Network

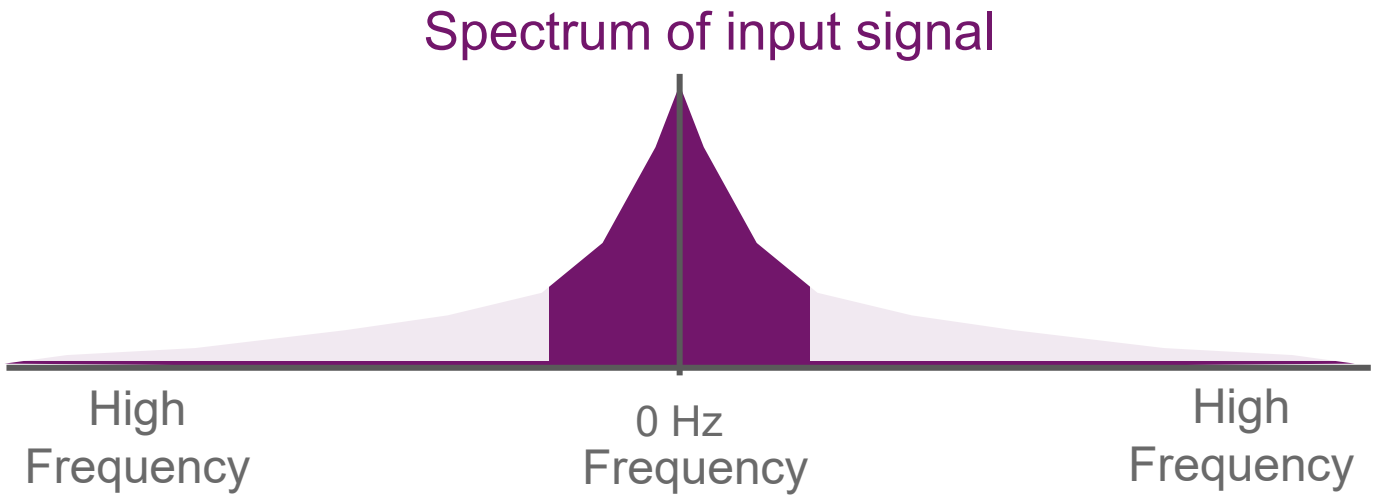


- Key features
  - Up to (and beyond) 1 Mbps bit rate. Bit time is limited by signal propagation delay.
  - Differential bus used for noise immunity
  - 11 or 29 bit message identifiers (message-based addressing) for **lossless arbitration**
  - Up to 8 bytes of data per message
  - Bit-stuffing used for clock synchronization, error detection/signaling
  - 15-bit CRC for error detection
  - ACK field for receiver acknowledgements
  - Remote transmit (read) request
- Easy to analyze responsiveness
  - Highest priority message always wins arbitration
  - Worst case: have to wait for longest other message before sending highest priority message
  - Given set of all possible messages, can calculate maximum time for any message to get through
    - Just like a prioritized, non-preemptive task scheduler
- Widely used in automotive and industrial control networks

Modulation Motivation

Goal: Use available spectrum better

- Input data spectrum
  - Most of energy concentrated in low frequencies and DC
  - Called the “baseband signal”
- Higher frequencies
  - From sharp edges on signals
  - Not necessary to convey information
  - Could remove these and still communicate the data correctly

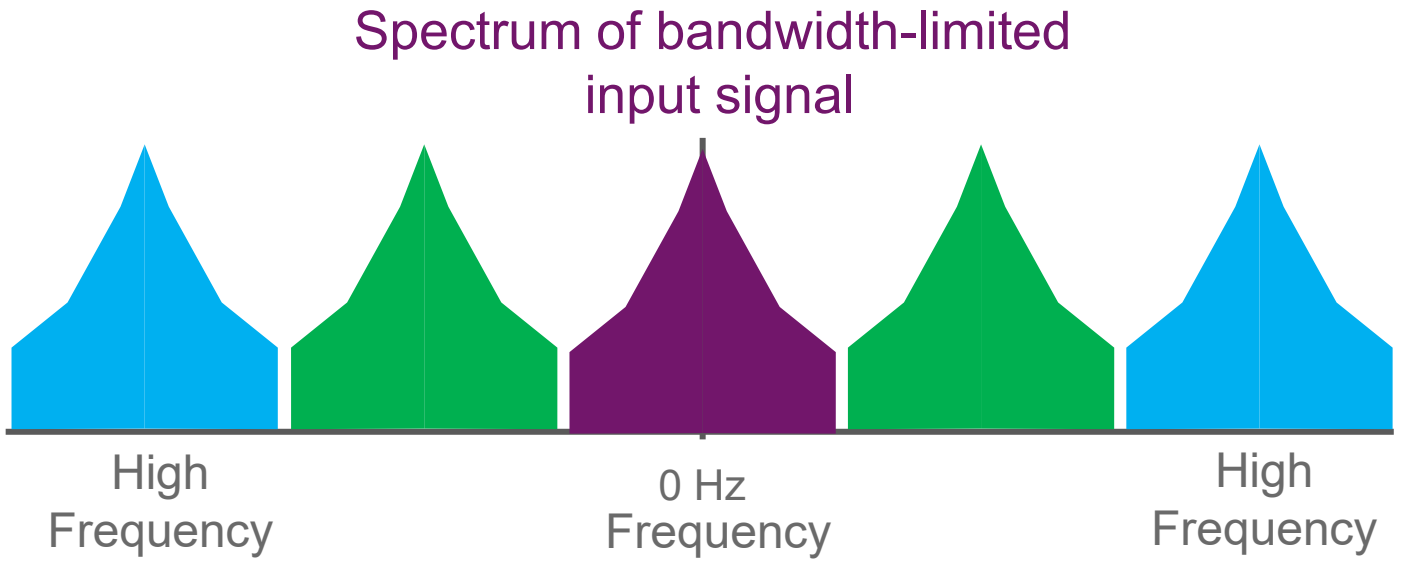


Filter out the signal’s high-frequency components

Modulation

Goal: Use available spectrum better

- We've freed up space in the spectrum
  - Just using the baseband portion
  - Multiple devices can use different parts of the spectrum
- Share spectrum by moving (modulating) baseband portions of signals up to free parts of the spectrum
- Used for cable TV and other wired communications

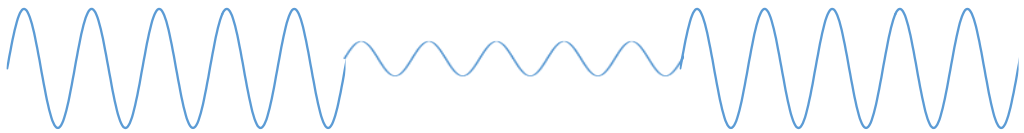


Can now support three communication channels on same wire

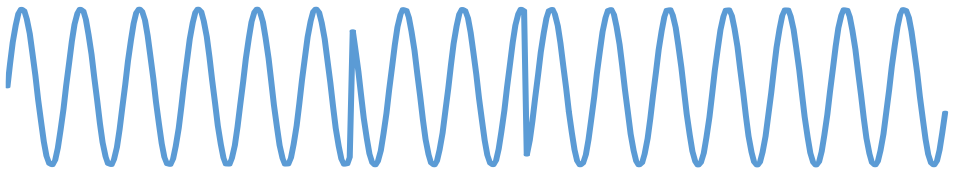
Common Types of Modulation

Many others exist

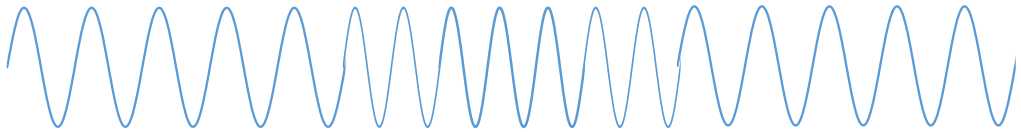
- Amplitude



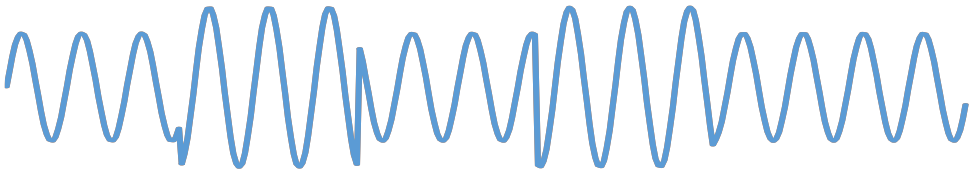
- Phase



- Frequency



- Quadrature Amplitude (phase and amplitude)



Radio Communications

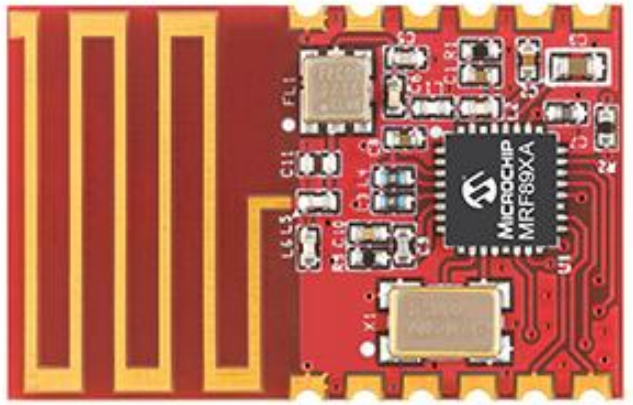
"Wireless"

- Use electromagnetic signals to send information without a physical connection
- Requires modulation to be practical
  - Low frequency radio signals require enormous antennas to work!
- **Transmitter** modulates a radio-frequency (RF) carrier signal with the data
  - RF carrier signal is high frequency sine wave, uses small antenna
  - Translate data bits into RF symbols (modulate the carrier signal), then send those RF symbols
- **Receiver** demodulates signal and extracts data

- MAC often uses Carrier Sense
  - Wait for media to be idle before transmitting
- Issue: Transmitter can't detect a collision!
  - Transmitter must wait for receiver to acknowledge receiving the message correctly

Example Protocol: IEEE 802.15.4

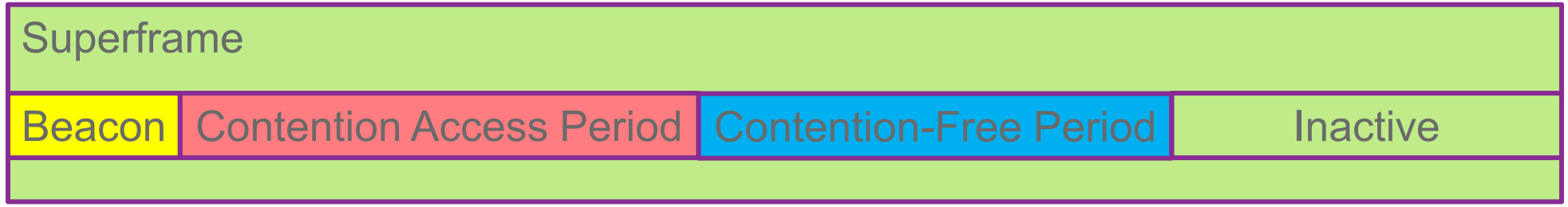
- Targets low-rate wireless personal area networks (LR-WPANs)
- Designed for low-cost, low-speed, short-range communication for low-power systems
  - 10 meter range, ≤ 250 kbps data rate
  - High carrier frequencies (868 MHz to 2.4 GHz) allow use of small antennas, can even fit onto PCB)
  - Beacon mode allows scheduled communications, so nodes save power by sleeping when possible, but will still get messages through reliably
  - Configurable time delays built into protocol to support sleeping or slower processors
- Protocol
  - Defines physical layer and MAC
  - Used in other protocols which define upper layers



ZigBee, MiWi, WirelessHART, 6LoWPAN, ISA100.11a, etc.	7. Application	
	6. Presentation	
	5. Session	
	4. Transport	
	3. Network	
	2. Data Link	LLC
802.15.4		MAC
	1. Physical	

IEEE 802.15.4 MAC

Offers two modes: beacon and non-beacon

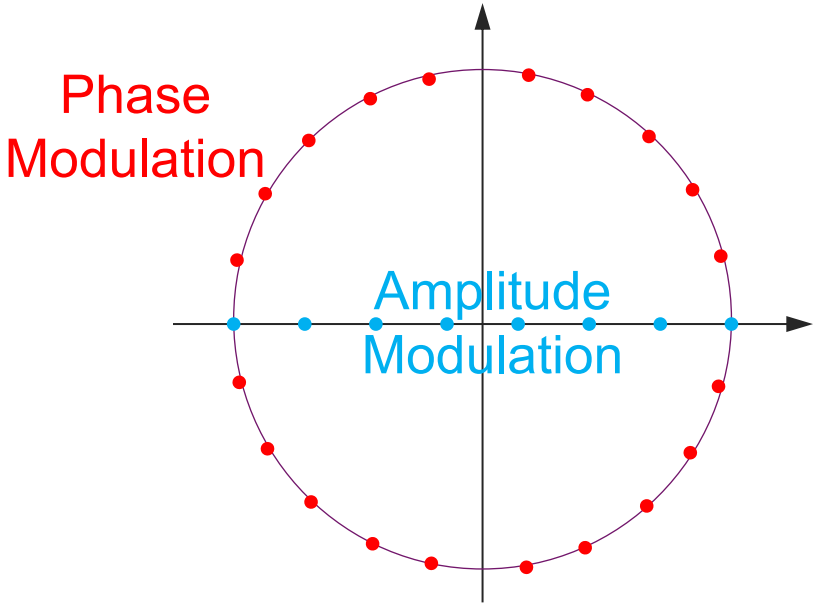


- Beacon mode
  - Requires node to be coordinator (master)
  - Allows tailoring of protocol to reduce power consumption and improve timeliness
  - Coordinator defines superframes by sending out beacon messages
  - Contention access period (CAP) follows beacon
    - Nodes send data using CSMA/CA
  - Contention-free period (CFP) follows CAP
    - Nodes can request guaranteed time slots (GTSs) from coordinator
  - Inactive portion – no communication occurs
    - Nodes may turn off radios, sleep here
- Non-beacon mode
  - No superframe, no beacon
  - Nodes send data using CSMA/CA, backing off after collisions

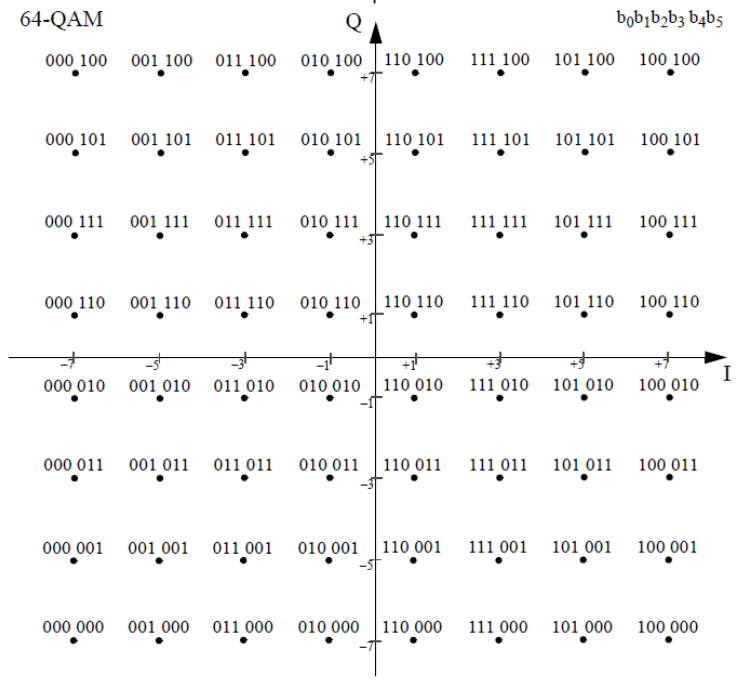
Increasing Data Rates

Several ways

- Reduce protocol overheads
  - Acknowledge faster, etc.
- Send symbols ***faster***
  - Also need to use more bandwidth and higher frequencies
  - Eventually gets very hard to go faster
- Send ***more data per symbol*** by changing phase, amplitude or frequency of carrier. Example: QAM (quadrature amplitude modulation)
  - Constellation plots show in-phase (sine) and quadrature (cosine) components of modulated signal
  - More points = higher data throughput
  - Closer points = harder for receiver to distinguish
  - QAM packs in more data points than just modulating only phase or amplitude



Quadrature Amplitude Modulation



Example Protocol: IEEE 802.11 (Wi-Fi)

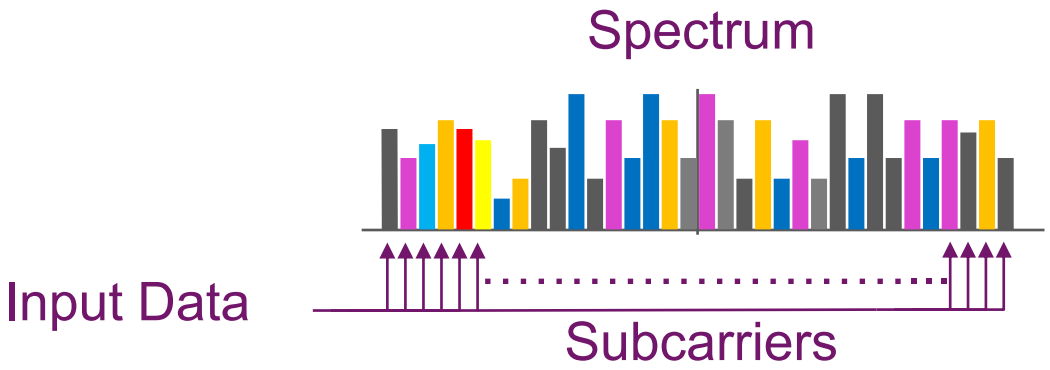
Overview

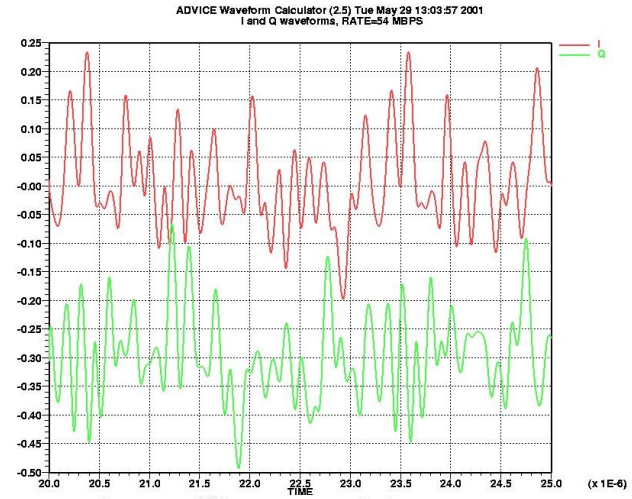
- Very fast local area network
  - Up to 1.7 Gbps throughput
  - Range up to 300 ft with standard antennas, much farther with directional antennas
- Very popular: used for computers, smartphones, printers, thermostats, cameras, smart televisions, security systems, IoT devices, etc.
- CSMA/CA used for media access control
- Physical layer is the key to its speed:
  - Each symbol transmits multiple data bits simultaneously

Year	Standard	Frequency Band	Bandwidth	Maximum Data Rate
1997	802.11	2.4 GHz	20 MHz	2 Mb/s
1999	802.11b	2.4 GHz	20 MHz	11 Mb/s
2000	802.11a	5 GHz	20 MHz	54 Mb/s
2003	802.11g	2.4 GHz	20 MHz	54 Mb/s
2009	802.11n	2.4 GHz, 5 GHz	20 MHz, 40 MHz	600 Mb/s
2014	802.11ac	5 GHz	20 – 160 MHz	1733 Mb/s

Example Protocol: IEEE 802.11 (Wi-Fi)

802.11a/g/n OFDM Physical Layer





- Physical layer optimized to tolerate reflected signals, moving devices (Doppler shift), dead frequencies, bursts of noise, clock drift, etc.
- Transmitter sends packet made of multiple symbols
  - **Preamble** symbols synchronize and help train receiver (delays, weak frequencies), and define protocol options used
  - **Data** symbols carry actual data
  - Each symbol lasts 4 µs
- Creating a data symbol
  - Transmitter groups input data bits to define the desired output spectrum for each symbol
  - Spectrum has up to 128 sub-carriers
  - Subcarriers will be modulated based on input data bits or protocol support (e.g. pilot tones)
  - Spectrum (frequency-domain) is converted to time-domain signal with Inverse Discrete Fourier Transform (IDFT)
- Resulting signal is transmitted out antenna