PURDUE UNIVERSITY DEPARTMENT OF PHYSICS

Physics 53600

Electronics Techniques for Research

Now in PowerPoint!

Spring 2020 Semester

Prof. Matthew Jones

The usual ANNOUNCEMENT

- Obvious changes to the course:
  - No in-person lectures: you'll have to read the lecture notes yourself
  - No more labs: don't worry about it – your grade will be based on work done so far
  - Remaining assignments will try to cover topics that would have been explored in the lab
  - Second mid-term: simplest to cancel it
  - Final exam: I'm not sure what to do about this yet, but I'll figure something out.
- Changes to grading scheme:
  - Old scheme: Assignments (30%) exams (40%) lab (30%)
  - New scheme: Assignments (50%) exams (25%) lab (25%)

The usual ANNOUNCEMENT

- Because there won't be any in-person lectures, you will have to read the lecture notes yourself.
- To demonstrate that you have read them, you will be required to answer *one or two simple questions* before the next lecture is posted.
- The question will be somewhere (like maybe at the end?) and you just have to e-mail me the answer

  mjones@physics.purdue.edu
- To make this easy, please make your subject look like this:

  "PHYS53600 Lecture xx questions Your Name"
- These will be part of your assignment grade, maybe contributing 10% of your total grade.

More ANNOUNCEMENTS

- Feel free to send me questions about the lecture material if there is anything you don't understand. I'm happy to give more explanation (and I'm soooo bored.)
- Send me e-mail if you think it would be useful to arrange a time as a class to have a time where you can ask questions by video.

LECTURE 22 QUESTIONS

1. *Why did the original RS-232 standard use such large voltage swings on its signals?*
2. *What are some advantages of RS-485 compared with RS-232?*
3. *What are the main differences between RS-422 and RS-485?*

Serial Digital Data Transfer

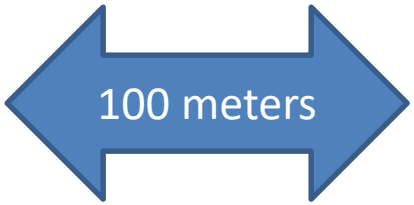
- So far we have considered various types of data transfer (both serial and parallel)
- All types we have considered use a clock signal (or strobe, or whatever you want to call it) to synchronize the sampling of a data signal
- This works, provided the data satisfies well-defined setup and hold times.
- Today we will investigate two other forms of serial communication that don't *explicitly* transmit clock signals

Serial Data Transfer

- We can first motivate the problem by considering transmitting serial data over a long distance (a few meters to a few hundred meters)
- Then, if we can make this work then it should (in principle) work even better (ie, faster) over shorter distances

Serial Interfaces

- With the emergence of mainframe computers in the 1950's-1980's, the human interfaces were usually located far away from the main computing hardware



Located in a carefully controlled secure area

Located in a publically accessible area

Signal Skew

- Obviously it will be easier and less expensive to have fewer signals (ie, wires) in an interface
- But, if we send a dedicated clock signal then we must guarantee that the setup and hold times are respected even when the cable is very long
- Without careful physical construction (which adds expense) it can be hard to guarantee that the electrical length of all conductors in a cable are the same.
- It might work, but maybe can only be guaranteed to work at low speeds.

Serial Data Transmission

- Let's consider a scheme where we ONLY transmit the data signal
  - This would only require two conductors
- The problem is that the receiver doesn't necessarily know when to sample the data
  - This is the basic problem with all serial communication and it is solved in various ways
- Both receiver and transmitter must agree on the format of the data and the receiver can use this to deduce a local sample clock signal

Universal Asynchronous Receiver/Transmitter

- This is the scheme used for several serial protocols (RS-232, RS-485, …)
  - Both receiver and transmitter have their own internal clocks
  - These clocks need to run at *approximately* the same frequency
  - There is no way to guarantee the phase difference
- These internal clocks determine the rate at which bits are sent and received
  - This is called the baud rate

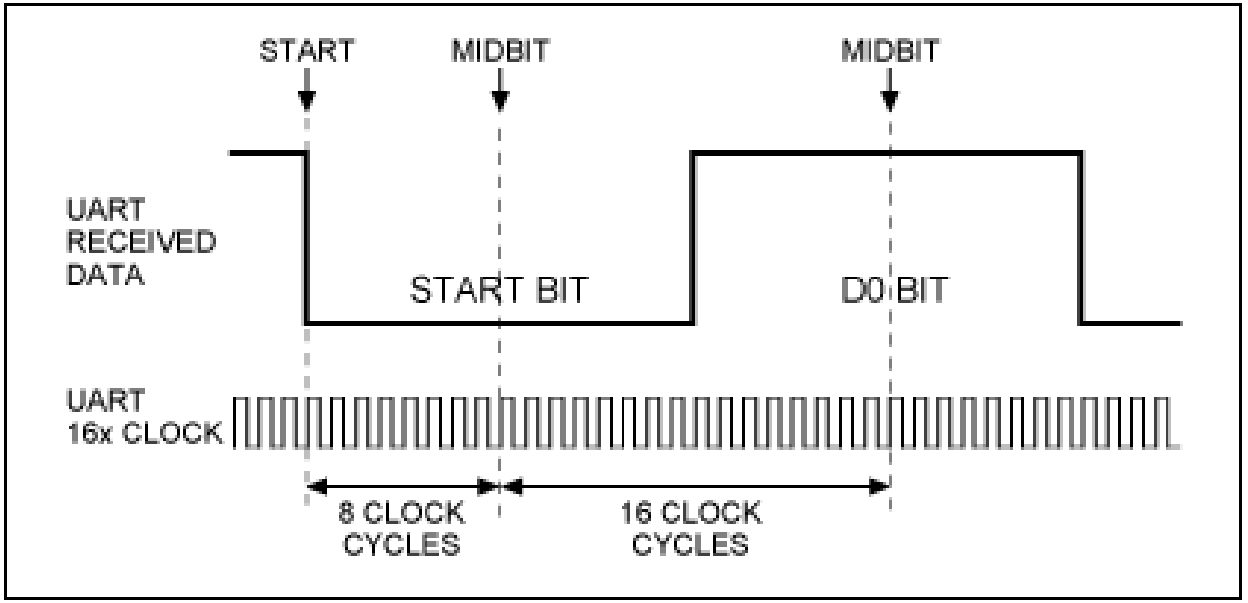
Baud Rate

- Both receiver and transmitter start with a local oscillator that runs at about 1.8432 MHz
- This is divided by integer factors to produce specific baud rates:

Baud rate	Divisor
300	6144
1200	1536
2400	768
9600	192
19,200	96
38,400	48
115,200	16

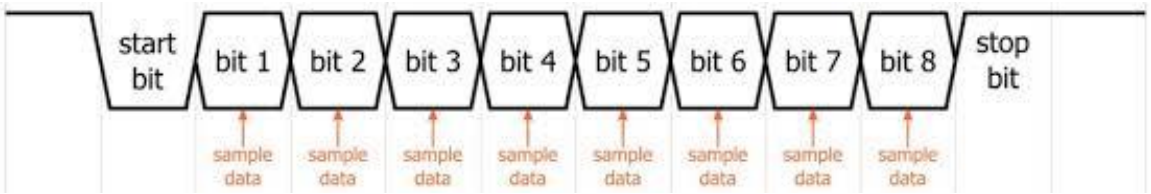
Start Bits

- The transmitter begins by sending a START bit
  - The receiver detects the start bit and knows (approximately) what the phase of bit boundaries are



Data Bits

- After the start bit is detected, the transmitter can send a number of data bits
  - Both transmitter and receiver need to agree on how many
- After the data has been transferred, the transmitter must get ready to send another start bit
  - This is like a STOP bit, and returns the signal to the opposite logic state of the START bit so that the next START edge can be detected



Error Detection

- There are several ways to improve the signal-to-noise ratio on the electrical signals
  - Remember Shannon's channel capacity theorem?
- If errors are still possible, then there are some simple ways to detect them
  - At least you can try to detect some of them
- One simple scheme is to transmit a "parity" bit that ensures that the total number of 1-bits in the data+parity is either even or odd:
  - Even parity: Only an even number of 1 bits
  - Odd parity: Only an odd number of 1 bits

Error Detection

- The “bit-error-rate” (BER) is the rate at which bits are incorrectly received.
- Parity will detect the flips of any single bit in the data+parity word
- Example with 8-bit data words and one parity bit:

$$P(\text{no errors}) \approx 1\text{-}BER^9$$

$$P(\text{one error}) \approx 9 \times BER$$

$$P(\text{two errors}) \approx 9 \times 8 \times BER^2$$

- There is still a (small) chance that there will be two bit flips in one word
  - This will not be detected by the parity bit

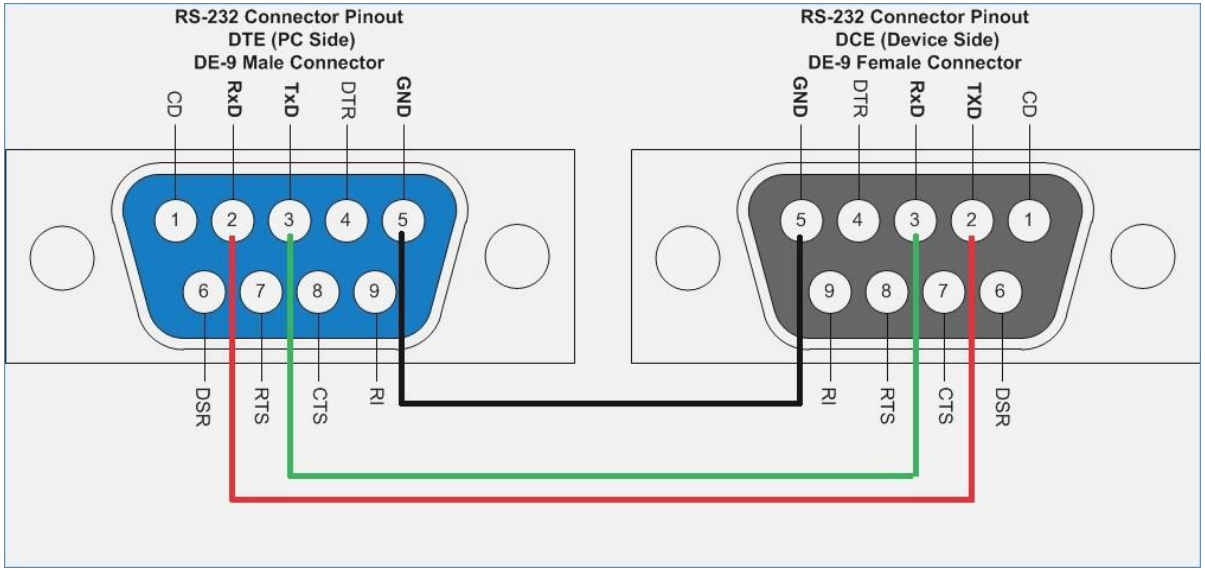
Physical Interface

- The protocol for transferring serial data is specified at the bit level (logic 0 or 1)
- Physically, the electrical connections can be implemented in various ways.
- It can also be convenient to provide other signals to help with data synchronization
- Physical connectors:
  - DB-9 (male and female)
  - DB-25 (male and female)
  - RJ-11
  - RJ-45

No really good universal standard – often vendor dependent

DB-9 Connectors

- This is usually the bare minimum number of signals that are used

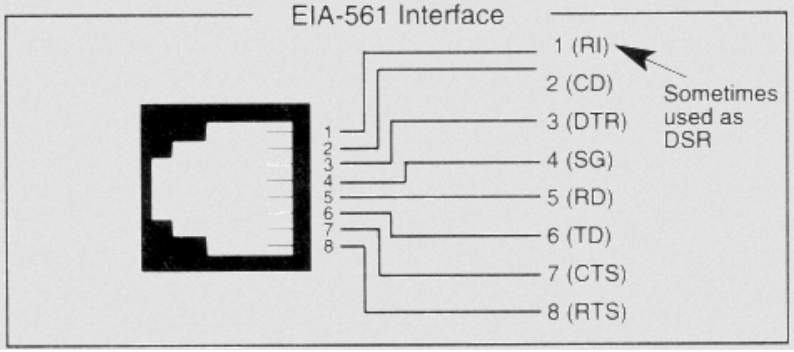


CTS and RTS signals are used for flow-control.

RJ-45 Connectors

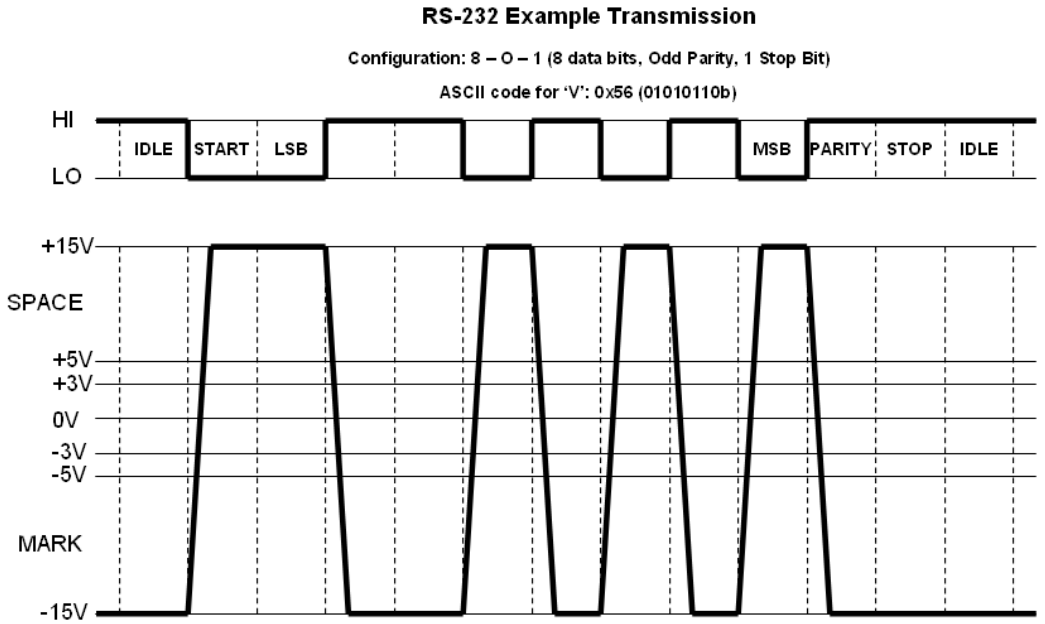
- RJ-45 connectors are convenient because they can use regular CAT-4/5 Ethernet cables with inexpensive connectors at both ends
- Sometimes you have to make adapters to connect up your own weird equipment

EIA-561 defines RS-232 on a modular connector. (For nonsynchronous applications only, since it does not provide for the synchronous clocking signals.)



Electrical Interface

- The original RS-232 specification used large voltage swings to increase the signal-to-noise ratio
- The voltage levels were called “mark” and “space”:



Electrical Interface

- In practice, it can be quite inconvenient to work with ±15 volt signals in digital circuits.
- There are driver IC's that will translate standard 3.3V or 5V logic levels to these higher voltages
- For example:

MAX3232-EP

www.ti.com

SGLS337A–APRIL 2006–REVISED MARCH 2009

3-V TO 5.5-V MULTICHANNEL RS-232 LINE DRIVER/RECEIVER
WITH ±15-kV ESD PROTECTION

Disadvantages of RS-232

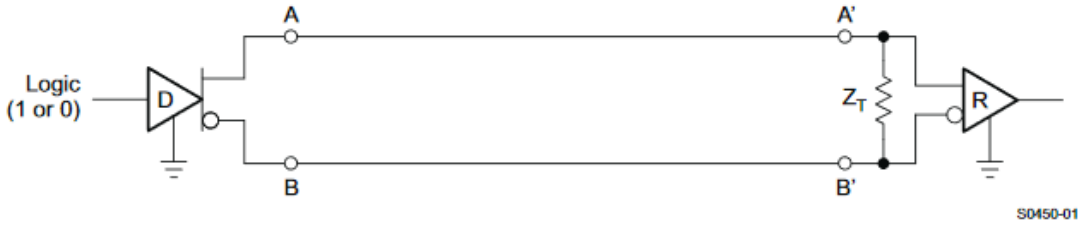
- The signals are single-ended
  - Noise is added linearly to the voltage being driven
- The voltage swings are potentially large
  - They can radiate noise that might interfere with other circuits
- The signals are ground-referenced
  - Both pieces of equipment have to be connected to the same electrical ground or else large currents might flow in the cable
- The signals are point-to-point
  - You can’t share the signals with multiple devices on the same cable

RS-422/RS-485 Electrical Specification

- The RS-422/RS-485 specification attempts to address many of these deficiencies.
- The data that is transferred is still sent/received using the UART protocol
- Additional conventions are required to allow multiple devices to share the same bus

RS-422/RS-485 Electrical Specification

- Both use a single pair of wires to transfer differential signals:



A D = driver (or generator)
B R = receiver
C Z_T = termination impedance

Figure 1. RS-422 Balanced-Voltage Digital-Interface Circuit

- There is no common ground connection
- The wires act as a transmission line, so Z_T is needed to prevent reflections at the receiving end.

RS-422 Specification

- With the RS-422 specification, one transmitter (the master) can send data to many (up to 10) receivers (slaves)
- The master can also receive data (on a separate pair of signals) from many slaves that share the bus
- The slaves have to decide who will transmit data at any given time.
- Cables can be long! Up to 1 km with data rates less than 100 kbps (kilo-bits per second)

RS-485 Specification

- The RS-485 specification is built on the RS-422 specification
  - They use the same differential signals
- But the pair of signals is truly bidirectional
  - Not only can there be multiple slaves, but there can also be multiple masters
  - Up to 32 drivers and 32 receivers on a single pair
- Now all the devices need to coordinate who will be driving the pair at any given time.

Example RS-485 Communications

- A standard communication protocol is needed to make this work
- One example is the MODBUS protocol
- Each device is assigned an 8-bit address
- Data is organized as a set of 16-bit registers
- The master sends data in the following format:

Slave Address	Function Code	Start Address (Hi)	Start Address (Lo)	Number of Points (Hi)	Number of Points (Lo)	Error Check (Lo)	Error Check (Hi)

Example RS-485 Communications

- The master then releases the bus (goes to a high impedance state and starts listening)
- The slave then transmits a response:

Slave Address	Function Code	Byte Count	Data (Hi)	Data (Lo)	Error Check (Lo)	Error Check (Hi)

- Error checking is performed by constructing a 16-bit word from the data
- If the word calculated from the received data does not match the transmitted data, then an error must have occurred.

Example of MODBUS Error Checking

```
BEGIN
    ErrorWord = Hex (FFFF)
        FOR Each byte in message
            ErrorWord = ErrorWord XOR byte in message
                FOR Each bit in byte
                    LSB = ErrorWord AND Hex (0001)
                    IF LSB = 1 THEN ErrorWord  = ErrorWord – 1
                    ErrorWord = ErrorWord / 2
            IF LSB = 1 THEN ErrorWord  = ErrorWord  XOR Hex (A001)
                NEXT bit in byte
        NEXT Byte in message
END
```

Examples of RS-485 Equipment

- Lots of industrial equipment uses RS-485 for communications.
- Most computers do not have a specific RS-485 interface.
- You can buy RS-485 to USB converters:

Examples of RS-485 Equipment