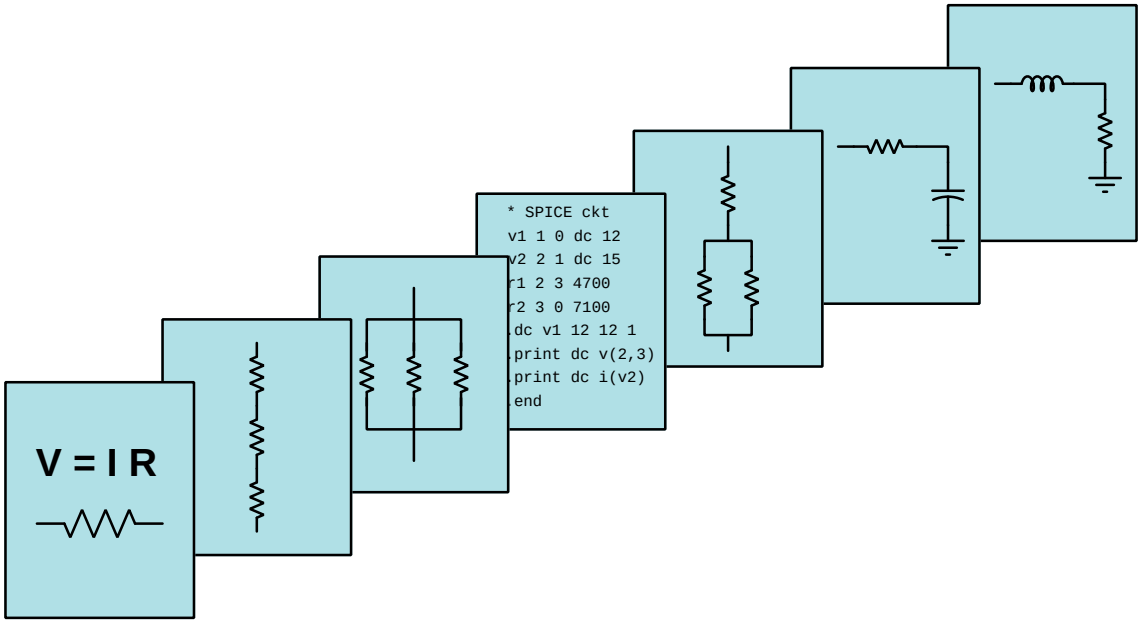
Modular Electronics Learning (ModEL) project



EIA/TIA-232, 422, and 485 Serial Networks

© 2020-2026 by Tony R. Kuphaldt – under the terms and conditions of the Creative Commons Attribution 4.0 International Public License

Last update = 14 March 2026

This is a copyrighted work, but licensed under the Creative Commons Attribution 4.0 International Public License. A copy of this license is found in the last Appendix of this document. Alternatively, you may visit `http://creativecommons.org/licenses/by/4.0/` or send a letter to Creative Commons: 171 Second Street, Suite 300, San Francisco, California, 94105, USA. The terms and conditions of this license allow for free copying, distribution, and/or modification of all licensed works by the general public.

Contents

1 Introduction **3**
- 1.1 Recommendations for students . . . 3
- 1.2 Challenging concepts related to EIA/TIA-232, 422, and 485 networks . . . 5
- 1.3 Recommendations for instructors . . . 6

2 Case Tutorial **7**
- 2.1 Example: UART data frames with and without parity . . . 8
- 2.2 Example: serial ASCII data shown on an oscilloscope . . . 11
- 2.3 Example: GPS receiver serial data over EIA/TIA-232 . . . 13

3 Tutorial **15**
- 3.1 Serial data communication and shift registers . . . 16
- 3.2 EIA/TIA-232 . . . 20
- 3.3 EIA/TIA-422 and EIA/TIA-485 . . . 25

4 Derivations and Technical References **33**
- 4.1 ASCII character codes . . . 34
- 4.2 The OSI Reference Model . . . 35
- 4.3 Using an oscilloscope on differential networks . . . 37

5 Questions **39**
- 5.1 Conceptual reasoning . . . 43
  - 5.1.1 Reading outline and reflections . . . 44
  - 5.1.2 Foundational concepts . . . 45
  - 5.1.3 Minimalist EIA/TIA-232 system . . . 47
  - 5.1.4 EIA/TIA-232 data frames of ASCII characters . . . 48
  - 5.1.5 Multipoint and multidrop communications . . . 54
  - 5.1.6 422 or 485? . . . 56
- 5.2 Quantitative reasoning . . . 57
  - 5.2.1 Miscellaneous physical constants . . . 58
  - 5.2.2 Introduction to spreadsheets . . . 59
  - 5.2.3 Terminating and bias resistors . . . 62
  - 5.2.4 EIA/TIA-232 data transmission time . . . 63
- 5.3 Diagnostic reasoning . . . 64

5.3.1 Testing a multi-pair data cable 65
5.3.2 Networked DAQ modules 66

A Problem-Solving Strategies **69**

B Instructional philosophy **71**
B.1 First principles of learning 72
B.2 Proven strategies for instructors 73
B.3 Proven strategies for students 75
B.4 Design of these learning modules 76

C Tools used **79**

D Creative Commons License **83**

E References **91**

F Version history **93**

Index **94**

Chapter 1

Introduction

1.1 Recommendations for students

Some of the simplest types of digital communication networks used to connect different computer devices together are defined by the EIA (Electronic Industry Alliance) and TIA (Telecommunications Industry Alliance) groups, under the numerical labels 232, 422, and 485. This module explores these three network types.

Important concepts related to these communication network standards include the **encoding** of information in digital form, **delimiting** of serial data, **NRZ** encoding, **bit rate**, **baud**, the packaging of information into **frames**, **start** and **stop** bits, **parity**, **full-duplex** versus **half-duplex**, **driver** versus **receiver** circuits, **flow control**, **single-ended** (unbalanced) versus **differential** (balanced) signals, **transmission lines**, **noise immunity**, **termination resistors**, **tri-state logic**, **DTE** versus **DCE** devices, and the **OSI model**.

Here are some good questions to ask of yourself while studying this subject:

- How might an experiment be designed and conducted to measure the noise margin within a serial data communication system? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- What minimum connections are necessary for two devices to communicate using EIA/TIA-232?
- What minimum connections are necessary for two devices to communicate using EIA/TIA-485?
- How does EIA/TIA-422 differ from EIA/TIA-485?
- Which of these networks is capable of the highest data rates for a given cable length?
- When is a *null modem* cable necessary?
- What is the purpose of a *loopback* jumper wire?
- How does differential signaling work to minimize received noise?

- What causes "reflected" signals within cables?
- What is "flow control" and why is it necessary in some applications but not in others?
- What are the different capabilities of 2-wire versus 4-wire EIA/TIA-485?
- What is *noise margin* and how is it calculated?
- Why are biasing resistors necessary for these networks?
- How could an oscilloscope be used to measure signals within an EIA/TIA-232 network?
- How could an oscilloscope be used to measure signals within an EIA/TIA-422 or EIA/TIA-485 network?
- How does the OSI Reference Model relate to different communication systems?

1.2 Challenging concepts related to EIA/TIA-232, 422, and 485 networks

The following list cites concepts related to this module's topic that are easily misunderstood, along with suggestions for properly understanding them:

- **Shift registers** – widely used in serial data communication, shift registers operate by "shifting" the locations in which bit-states are held by semiconductor flip-flops in their latching modes. "Parallel" mode refers to multiple bits being read into or read from a register circuit, while "serial" mode refers to one bit at a time being read into or read from a register.
- **Parity** – a common area of confusion for students is the subject of *parity bits* and error-checking in general. In order to understand these topics accurately and to avoid misconception, it is helpful to run some *thought experiments* whereby you imagine data becoming corrupted during serial transmission, and analyze how each of the error-checking techniques would be able to identify the presence of data corruption.
- **OSI reference model** – an easy way to get confused on this subject is to try to interpret it as a standard in and of itself. It is more accurately thought of as a taxonomy of communication instead: merely a way to label and describe aspects of communications standards. No single communications embodies all seven layers of the OSI model, and many relate to only one of those layers!
- **Single-ended versus Differential signals** – single-ended signals are also known as *ground-referenced* or *unbalanced* signals, since their voltage values are always measured with respect to ground. Differential signals which are also known as *balanced* signals, by contrast, are voltages measured between two non-grounded conductors. This is but one more example of how the inherently differential nature of voltage can be challenging to grasp, there being no such thing as voltage existing at any single point in any network.
- **Synchronous versus Asynchronous communication** – synchronous communication is when two or more digital devices communicate in lock-step with each other due to the simultaneous transmission of a clock signal. Asynchronous communication is when two or more digital devices use internal clocks to keep pace with each other, but rely on a "start" signal of some kind to synchronize themselves together at the start of each data frame. EIA-232 is an asynchronous communication standard.

The *Case Tutorial* chapter contains sections showing pulse waveforms for UART and EIA/TIA-232 which are helpful in learning to decipher serial data frames and their parity bits, as well as examples of real EIA/TIA-232 circuits using transceiver ICs.

1.3 Recommendations for instructors

This section lists realistic student learning outcomes supported by the content of the module as well as suggested means of assessing (measuring) student learning. The outcomes state what learners should be able to do, and the assessments are specific challenges to prove students have learned.

- **Outcome** – Demonstrate effective technical reading and writing

  Assessment – Students present their outlines of this module's instructional chapters (e.g. Case Tutorial, Tutorial, Historical References, etc.) ideally as an entry to a larger Journal document chronicling their learning. These outlines should exhibit good-faith effort at summarizing major concepts explained in the text.

- **Outcome** – Interpret serial data frames from oscillographs

  Assessment – Correctly determine ASCII characters from oscillographs of serial data transmissions; e.g. pose problems in the form of the "EIA/TIA-232 data frames of ASCII characters" Conceptual Reasoning question.

- **Outcome** – Diagnose a faulted communication network

  Assessment – Determine the probability of various component faults in a serial communication circuit given symptoms and measured values; e.g. pose problems in the form of the "Networked DAQ modules" Diagnostic Reasoning question.

Chapter 2

Case Tutorial

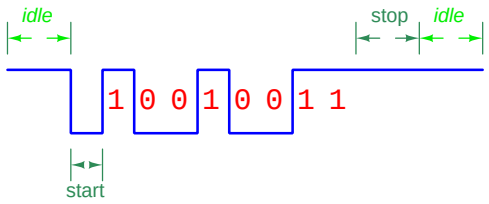
The idea behind a *Case Tutorial* is to explore new concepts by way of example. In this chapter you will read less presentation of theory compared to other Tutorial chapters, but by close observation and comparison of the given examples be able to discern patterns and principles much the same way as a scientific experimenter. Hopefully you will find these cases illuminating, and a good supplement to text-based tutorials.

These examples also serve well as challenges following your reading of the other Tutorial(s) in this module – can you explain *why* the circuits behave as they do?

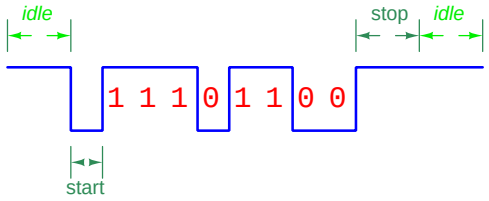
2.1 Example: UART data frames with and without parity

In the following illustrations we will show three examples of UART serial data signals representing eight-bit binary numbers, assuming the use of positive DC source voltage for "idle" and logical "1" states and zero voltage for logical "0" states[1], *no parity bit*, and two stop bits:

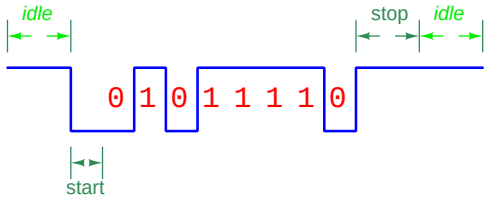
Binary number = 11001001



Binary number = 00110111



Binary number = 01111010

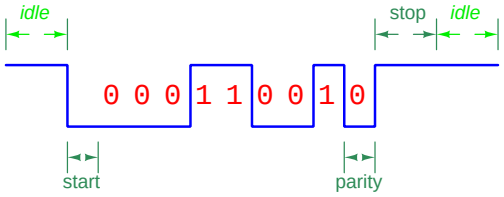


Note how UART data is always transmitted with the least-significant bit (LSB) sent first and the most-significant bit (MSB) last. This is why the pulse waveforms' high and low states seem to come in reverse order.

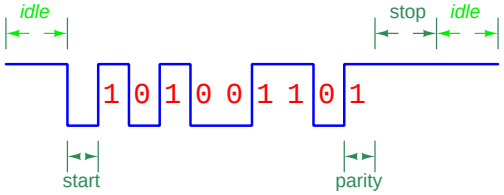
[1] Note how this "positive" logic is consistent with most digital logic devices such as logic gates, but is *opposite* of certain communication standards such as EIA/TIA-232 where a positive voltage is a 0 or "space" state and a negative voltage is a 1 or a "mark" state!

Next we will examine three more examples of UART signals using the same parameters as before (positive logic levels, 8 data bits, and 2 stop bits), but this time including a *parity bit* set for *odd parity*:

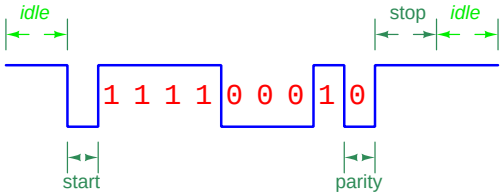
Binary number = 10011000



Binary number = 01100101



Binary number = 10001111



Note how each one of the bits' time duration is identical, the data bits each occupying the exact same amount of time on the horizontal axis as each start bit, each stop bit, and each parity bit. The duration of the "idle" time is arbitrary, depending only on how often binary numbers get communicated.

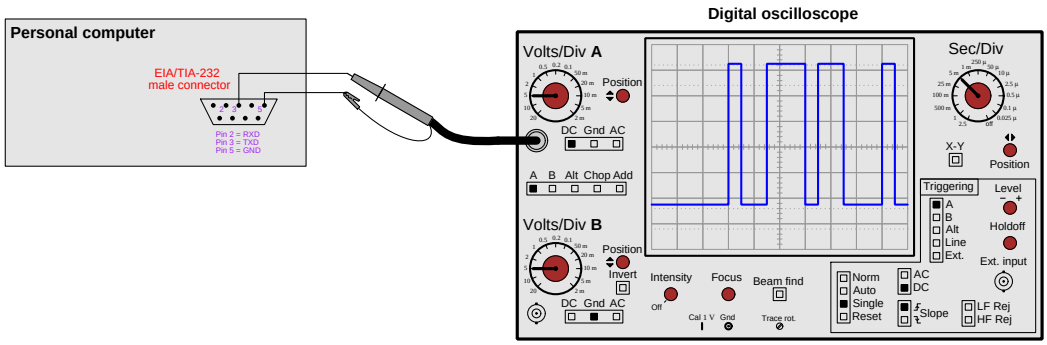
Parity bits exist for the purpose of *error detection* in serial data communication. Errors usually originate in communication systems due to excessive electrical *noise*, and so it is possible for noise to grow strong enough to corrupt one of the bits so as to be interpreted at the receiving end as the wrong logical state; i.e. a bit transmitted as a 0 being received as a 1, or vice-versa. If both transmitting and receiving devices are pre-configured to generate and interpret (respectively) a parity bit whose value is based on the number of "1" bits in the data payload, any single-bit corruption will be detectable as such when the receiver does its own parity calculation and determines a mis-match.

In the last set of UART signal examples shown, parity was configured for "odd". This meant that the transmitting device counted up the number of "1" bits in the eight-bit binary number and then made the parity bit either a "1" or a "0" as necessary to bring the total "1" bit count to an odd value. For example, the binary number `10011000` in the first parity example already had an odd number of "1" bits and so the transmitting device set the parity bit to zero because it didn't need any more "1" bits to make an odd count. However, the binary number `01100101` used in the

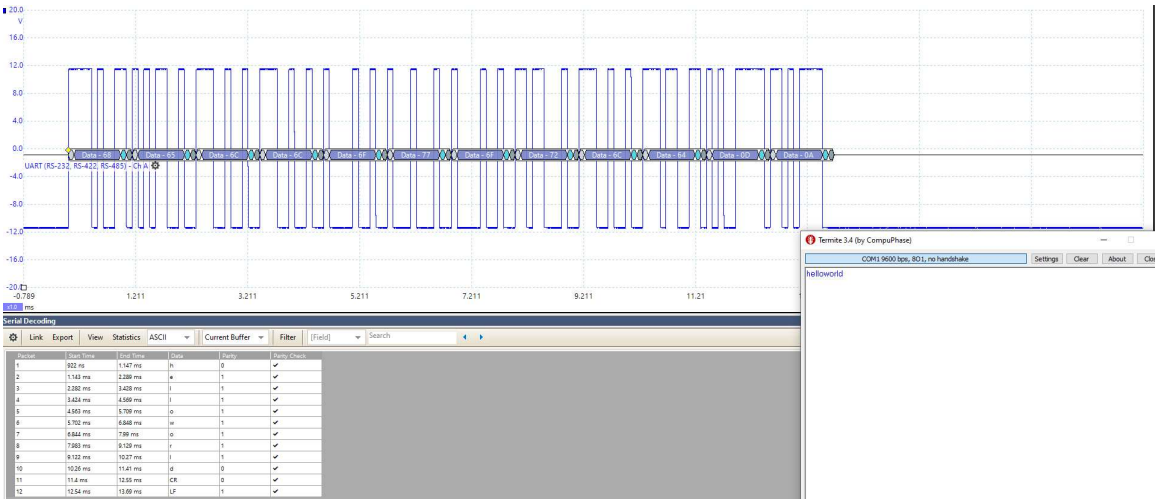
second parity example had an even number of “1” bits and so the transmitting device did need to set the parity but to one in order to being the total “1” bit count to an odd number. The receiving device simply has to count all the “1” bits it receives in total to see if it finds an odd number. If it happens to count an even number of “1” bits in total, it would knows something has gone wrong.

2.2 Example: serial ASCII data shown on an oscilloscope

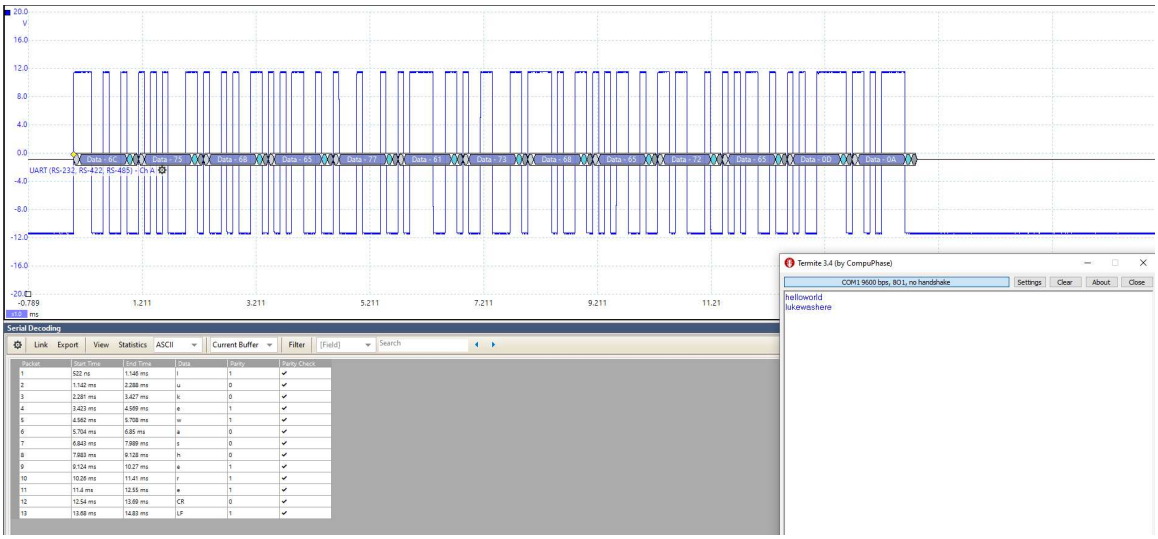
A digital oscilloscope configured for single-sweep mode may be used to capture the EIA/TIA-232 serial data stream transmitted by a personal computer if measuring voltage between the PC's "transmit" and "ground" pins (3 and 5, respectively):



One very simple way to prompt the computer to output serial data for the oscilloscope to capture is by running "terminal" software which converts keystrokes into ASCII-encoded data pulses. An example of this is shown below, using the terminal program `termite` to transmit character data and using a Picoscope model 2204A digital oscilloscope to capture the data pulses. Interestingly, this model of oscilloscope has the ability to decode EIA/TIA-232 serial data frames, as evidenced by the screenshot shown below where the oscillograph is annotated with hexadecimal values and color-coded "start" (light grey), "parity" (light blue), and "stop" (darker grey) bits:



Another screenshot appears below, again with the same personal computer displaying the Picoscope's annotated oscillograph and the `termite` terminal application simultaneously:

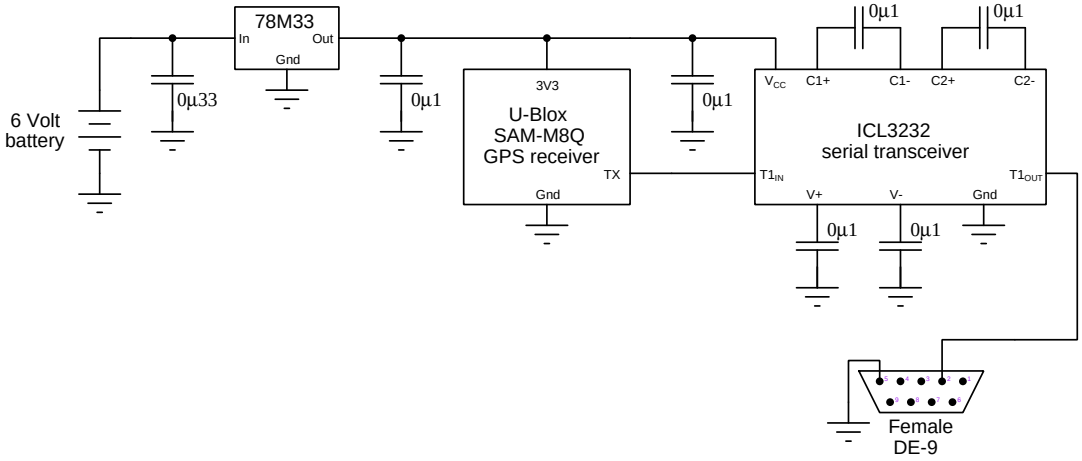


It is well worth your time to closely inspect the oscillograph to learn how to manually decode the "mark" and "space" states of the pulse waveform, checking to see if your interpretation matches the oscilloscope's.

Two characters appearing at the end of both captures are `CR` (carriage return) and `LF` (linefeed). These characters are customarily appended to any ASCII-encoded message whenever the user presses the computer's "Enter" key, signifying to the receiving device that the string of text represented by the former portion of the message should be concluded by returning the cursor to the left-hand margin of the screen (carriage return) and by stepping down to a new line on the display for the next text stream (linefeed). These ASCII "control characters" date back to the days of electromechanical *teletype* machines modeled after electric typewriters, where a mechanical "print head" required repositioning and the feed mechanism needed to increment the paper page up one line for the start of a new line of text.

2.3 Example: GPS receiver serial data over EIA/TIA-232

An interesting circuit utilizing serial data communication is shown here, where a U-Blox model SAM-M8Q Global Positioning System (GPS) radio receiver module interprets radio signals from satellites in geosynchronous orbit and generates a stream of positioning data sent out of its UART "transmit" port. The GPS receiver is powered by 3.3 Volts DC, which means the serial data signals at its `TX` pin consist of voltage pulses with an approximate range of 0 to 3.3 Volts. This is not a strong enough signal to meet the EIA/TIA receiver standard of ± 3 Volts minimum. In order to properly interface the GPS unit to the serial port of a personal computer, a *transceiver* IC must be used to shift the voltage levels. In this case I'm using a Renesas model ICL3232 which can run as fast as 250 kbps and uses capacitive charge-pump[2] circuitry to boost the +3.3 Volt supply voltage to ± 5.5 Volts:



The female DE-9 connector then attaches to the COM1 port of a personal computer. With suitable serial terminal software[3] running on the computer and configured with the proper serial parameters for the SAM-M8Q module (9600 bps 8N1) what you see on the computer's screen is a burst of ASCII characters at a rate of one burst per second, the characters following a GPS data standard called *NMEA-0183*.

[2] A *charge pump* is a type of DC-DC power converter circuit using capacitors that are switched back and forth between a DC source and a DC load, in such a way as to boost, buck, and/or isolate one voltage level from another. Like most IC charge pump circuits, the ICL3232 contains all the oscillator and transistor switching circuitry inside the IC but leaves the capacitors as external components.

[3] For example, `Kermit`, `Termite`, `minicom`, or other programs designed to receive ASCII characters via the RS-232 serial standard and display those alphanumeric characters on the computer's monitor.

A sample of the GPS positioning data obtained from this circuit is shown here:

```
$GNRMC,183420.00,A,4623.57504,N,11658.41126,W,0.286,,020821,,,A*7E
$GNVTG,,T,,M,0.286,N,0.531,K,A*36
$GNGGA,183420.00,4623.57504,N,11658.41126,W,1,06,2.04,437.7,M,-18.2,M,,*76
$GNGSA,A,3,25,02,12,15,23,13,,,,,,,4.39,2.04,3.89*17
$GNGSA,A,3,,,,,,,,,,,,,4.39,2.04,3.89*16
$GPGSV,4,1,14,02,17,074,09,05,66,092,,11,10,070,,12,08,173,25*76
$GPGSV,4,2,14,13,10,114,10,15,12,145,22,16,03,329,,18,40,264,*7A
$GPGSV,4,3,14,20,37,054,,23,10,207,23,25,30,197,20,26,25,309,*78
$GPGSV,4,4,14,29,85,355,,31,02,271,*7D
$GLGSV,1,1,04,85,51,062,,86,56,323,,87,01,284,,,,19*5B
$GNGLL,4623.57504,N,11658.41126,W,183420.00,A,A*62
```

These characters represent *codes* defined by the NMEA-0183 standard. The first line (beginning with `$GNRMC`) may be interpreted as follows, each of the data fields separated by comma delimiting characters:

- `183420.00` represents the Coordinated Universal Time, read as 18 hours, 34 minutes, 20.00 seconds (the time at Greenwich, London, England).
- `A` represents the data status, with "A" meaning valid and "V" meaning a receiver warning.
- `4623.57504,N,11658.41126,W` represents the latitude and longitude, respectively. In this case we read it as 46 degrees and 23.57504 minutes north, 116 degrees and 58.41126 minutes west.
- `0.286` represents the ground speed in nautical miles per hour (knots).
- `020821` represents the Coordinated Universal Time date field, read as 2 August 2021.
- `A` represents the mode indicator, with "A" meaning autonomous, "N" meaning data invalid, "D" meaning differential mode, and "E" meaning dead reckoning mode.
- `7E` represents the checksum for this block of data, for error-detection purposes.

Chapter 3

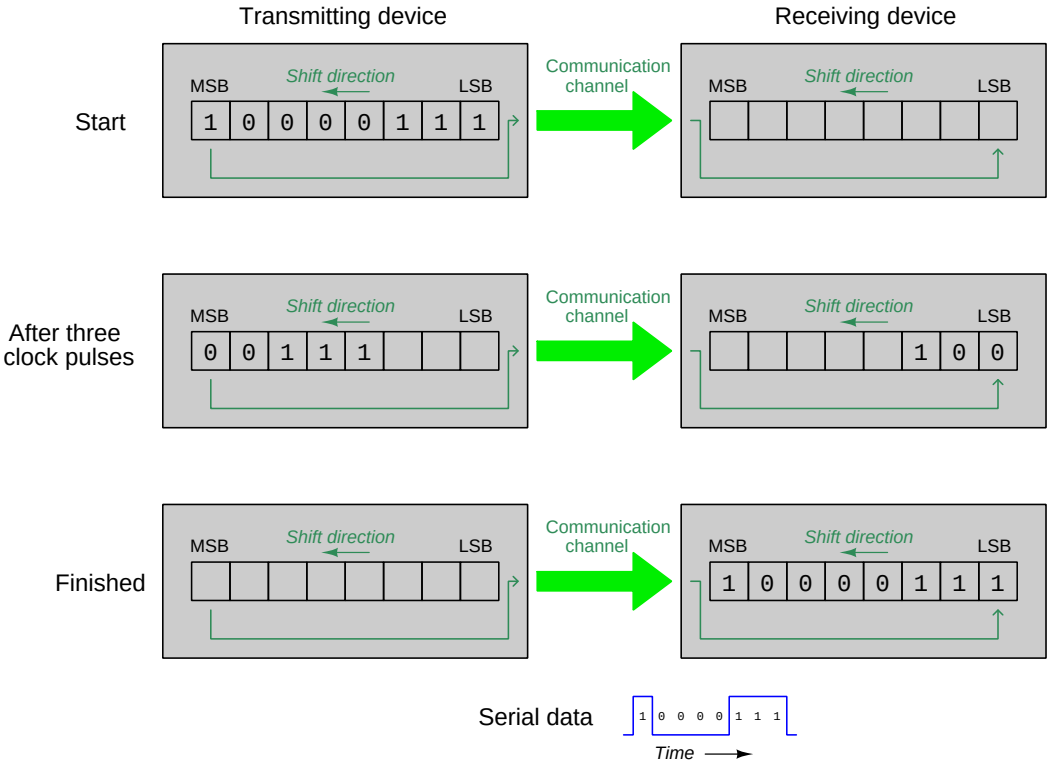
Tutorial

3.1 Serial data communication and shift registers

Digital data is comprised of *bits* of information, each bit being either a 1 or a 0, a high or a low, a true or a false. Collections of bits are generally known as *words*, although some specific numbers of them have unique names, such as a *byte* for an 8-bit word, or a *nybble* for a 4-bit word.

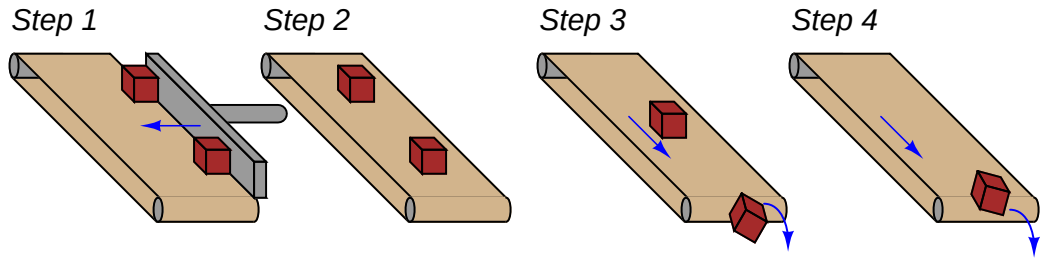
One way to communicate multi-bit digital words from one device to another is to do so one bit at a time over the same conductor(s), and this is known as *serial* communication because the bits are sent in a series over time rather than all at once using one conductor per bit (which is called *parallel* communication). Serial data communication is of course slower than parallel, but enjoys the decided advantage of using fewer conductors which is important in multiple contexts: for long-distance communication, fewer conductors means less expensive and less bulky cable; for communication between ICs on a circuit board, fewer conductors means fewer terminals (pins) necessary on the ICs to exchange that data which in turn means physically smaller devices are possible.

Multiple methods have been standardized for serial data communication, but fundamentally they all involve the use of digital devices known as *shift registers*. The following illustration shows a block diagram of a serial communication system where eight bits (one byte) are shifted out of the transmitting device's shift register, one bit at a time in sequence, and received at the receiving device's shift register to form a complete byte of data. This shifting of bits out of and in to the shift register circuits happens at the command of a signal called a *clock pulse*:

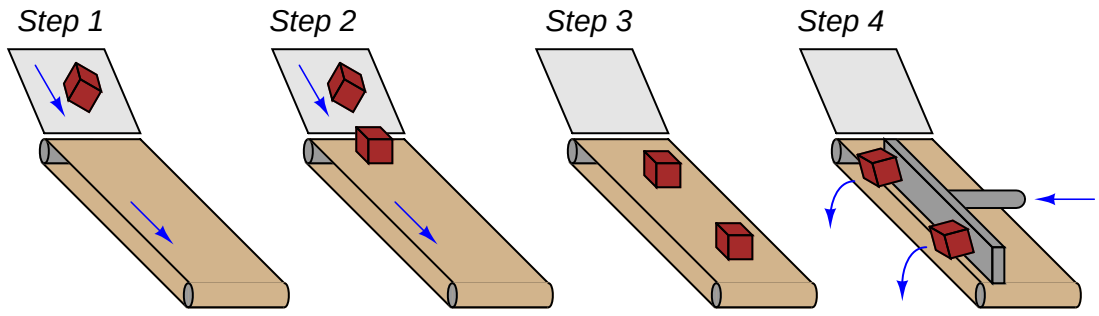


Shift registers are comprised of *flip-flops*, and are designed to move bits of data along from one flip-flop to another in sequence at the command of the clock pulse signal. The action of a shift register may be likened to a conveyor belt where packages move on to and off of the belt in various ways as shown below, the packages representing bits and the conveyor's motion representing the clock pulse:

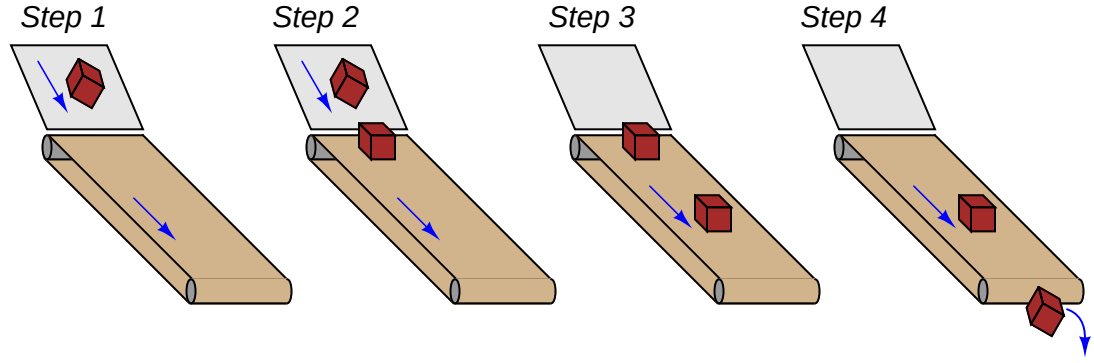
Parallel-in, serial-out (used by serial transmitting devices)



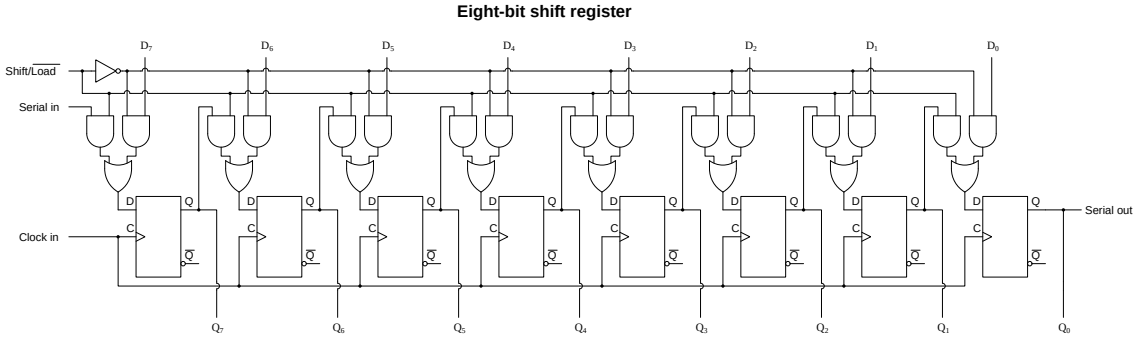
Serial-in, parallel-out (used by serial receiving devices)



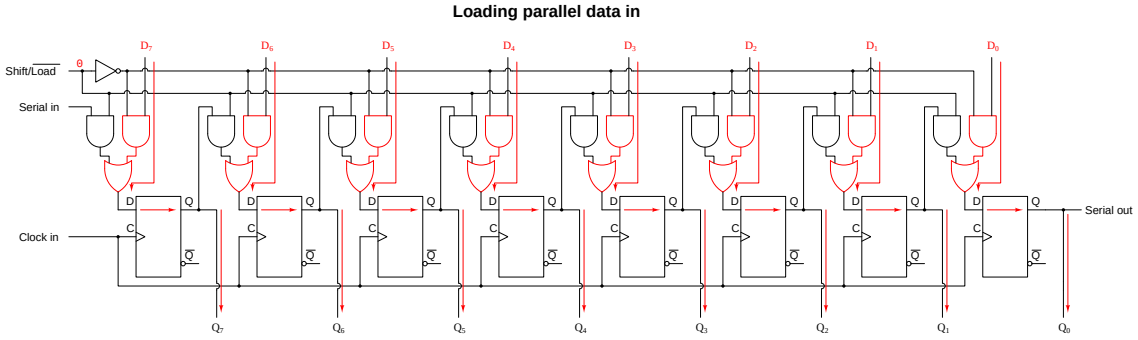
Serial-in, serial-out



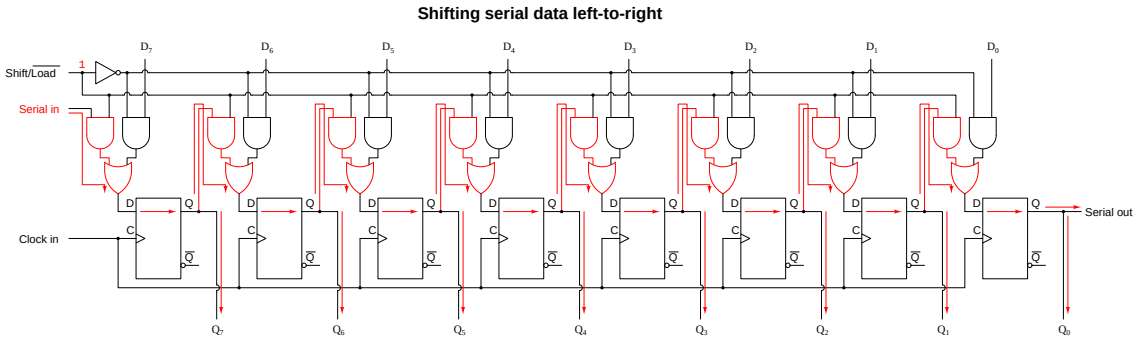
Shown below is a schematic diagram for an eight-bit (one-byte) shift register capable of all three modes of operation illustrated on the previous page. Lines D_0 through D_7 are for parallel data input, while lines Q_0 through Q_7 are for parallel data output. Serial data enters the shift register one bit at a time through the single "Serial in" line on the left, and exits the shift register one bit at a time through the single "Serial out" line on the right. A single "Clock" input receives a pulse signal causing parallel data to be fed into all the D-type flip-flops when the control line is in the Load state (i.e. "low"), and causing serial data to shift from left to right through all the D-type flip-flops when the control line is in the Shift state (i.e. "high"):



The AND/OR gate networks *steer* digital data to those flip-flops from different sources. In the "Load" mode of operation these steering gates route data from the parallel input terminals to the eight flip-flops as shown in red below:



In the "Shift" mode of operation these steering gates route data from one flip-flop stage to the next as shown in red below:



While all serial data communication systems employ shift registers at some level to convert between parallel and serial data formats, a range of different options distinguishes one standard from another. Some of this options include:

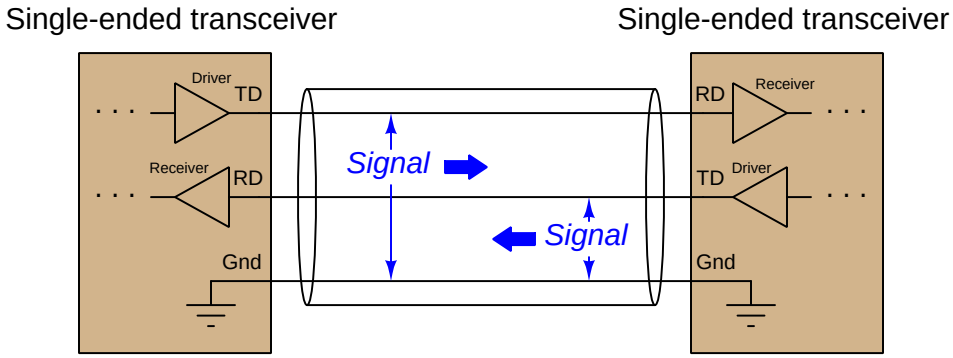
- The rate(s) at which the serial data may be communicated
- The manner in which clock pulses are synchronized between transmitter and receiver
- The exact voltage levels or other electrical characteristics of the serial data bits
- Properties of the communication channel connecting transmitter and receiver(s) together, for example the number of wires used and whether the signals are *single-ended* or *differential*

For example, a serial communication standard may specify certain allowable bit rates (i.e. bits per second), or let this be up to the end-user to decide. A serial communication standard may utilize a common clock pulse signal to drive both transmitter and receiver (called a *synchronous* network), or may use separate clock pulse generators at the transmitting and receiving ends which must be synchronized periodically to stay in-step with each other (called an *asynchronous* network). A serial communication standard may utilize common digital logic voltage levels to represent the 1 and 0 logical states of each bit, or it may use a much different set of voltage levels, or may even use some form of signal other than "high" and "low" voltage states to represent bits (e.g. using different audio-frequency tones). Many combinations exist on which to formulate a serial communications standard, which is why we have so many to choose from: EIA/TIA-232, EIA/TIA-485, SPI, I2C, 1-Wire, and Ethernet to name a few.

3.2 EIA/TIA-232

The EIA/TIA-232C standard, formerly[1] known as *RS-232*, is a standard defining details found at layer 1 of the OSI Reference Model (voltage signaling, connector types) and some details found at layer 2 of the OSI model (asynchronous transfer, "flow control" or "handshaking" signals between transmitting and receiving devices). In the early days of personal computers, almost every PC had either a 9-pin or a 25-pin connector (and sometimes multiple of each!) dedicated to this form of digital communication. For a while, it was *the* way peripheral devices such as keyboards, printers, modems, and mice connected to personal computers. USB (Universal Serial Bus) has now all but replaced EIA/TIA-232 for personal computers, but it still lives on in the world of industrial devices.

EIA/TIA-232 networks are point-to-point, intended to connect only two devices[2]. The signaling is *single-ended* (also known as *unbalanced*), which means the respective voltage pulses are referenced to a common "ground" conductor, a single conductor used to transfer data in each direction:



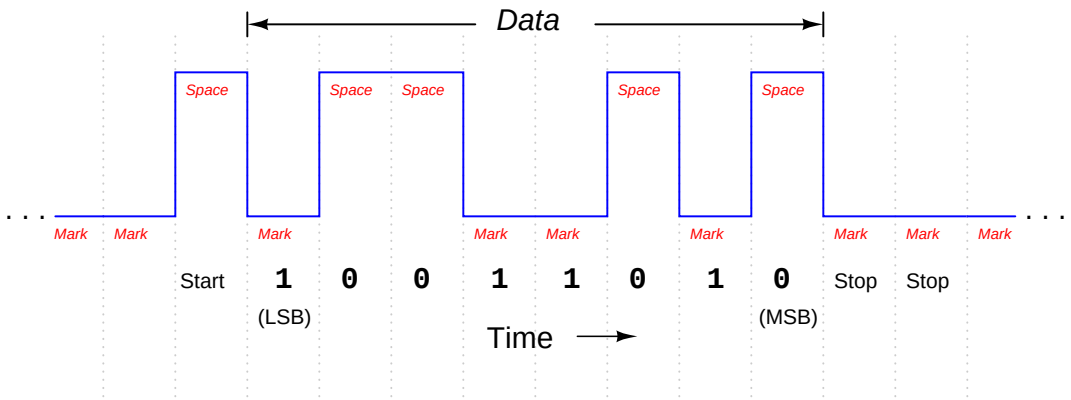
EIA/TIA-232 specifies positive and negative voltages (with respect to the common ground conductor) for its NRZ signaling: any signal more negative than −3 Volts detected at the receiver is considered a "mark" (1) and any signal more positive than +3 Volts detected at the receiver is considered a "space" (0). EIA/TIA-232 transmitters are supposed to generate −5 and +5 Volt signals (minimum amplitude) to ensure at least 2 Volts of noise margin between transmitter and receiver. The voltage limits and NRZ encoding of EIA/TIA-232 comprise the OSI layer 1 elements of the standard.

This is why *driver* and *receiver* devices are necessary to interface standard digital logic circuits to EIA/TIA-232 communication lines: the voltage levels are incompatible. A traditional TTL logic circuit operating with "high" 5-Volt and "low" 0-Volt signal levels would not meet the EIA/TIA-232 standard for transmission (± 5 Volts) nor would it necessarily tolerate the EIA/TIA-232 standard for received signals (± 3 Volts minimum). Such driver/receiver devices are commonly packaged together in *transceiver* integrated circuits.

[1] The designation of "RS-232" has been used for so many years that it still persists in modern writing and manufacturers' documentation, despite the official status of the EIA/TIA label. The same is true for EIA/TIA-422 and EIA/TIA-485, which were formerly known as RS-422 and RS-485, respectively.

[2] "Daisy-chain" networks formed of more than two devices communicating via EIA/TIA-232 signals have been built, but they are rarely encountered, especially in industrial control applications.

A simple example of an EIA/TIA-232 data frame is shown below, each "Mark" state being a negative potential and each "Space" state being a positive potential relative to ground. The minimum received signal strength according to the EIA/TIA-232 standard is ± 3 Volts, but EIA/TIA-232 transmitting devices output much stronger amplitudes than these in order to ensure some immunity to noise:

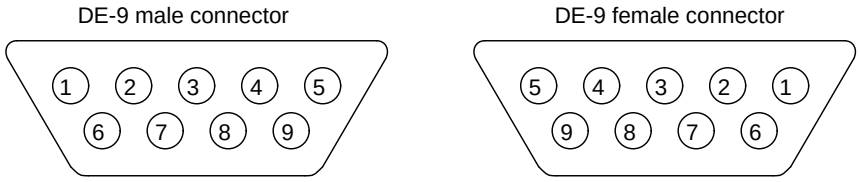


Serial bitstream for the digital byte 01011001, where the least-significant bit (LSB) is sent first

Note how the "idle" state of an EIA/TIA-232 network is the "Mark" voltage (i.e. negative with respect to ground). This makes every Start bit a "Space" in order to differentiate it from the idle "Mark" state. Note also how the Stop bit is a "Mark" state as well, preceding the next idle condition. This example frame happens to contain no parity bit, but if there was one it would lie between the MSB and the first Stop bit.

Since we see Stop bits being the exact same "Mark" state as the network's idle condition, one might wonder what the purpose of a Stop bit is at all? The real purpose of specifying the number of Stop bits for any EIA/TIA-232 message is simply to provide a guaranteed period of idleness on the network before the *next* message is transmitted. This is why Stop bits are the same state as idle, in order to actually be that guaranteed idle period between successive frames.

Cable connectors are also specified in the EIA/TIA-232 standard, the most common being the DE-9[3] (nine-pin) connector. The "pinout" of a DE-9 connector for any *DTE* (Data Terminal Equipment) device at the end of an EIA/TIA-232 cable is shown here:



Pin number	DTE assignment	Abbreviation
1	Carrier Detect	CD
2	**Received Data**	**RD**
3	**Transmitted Data**	**TD**
4	Data Terminal Ready	DTR
5	**Signal Ground**	**Gnd**
6	Data Set Ready	DSR
7	Request To Send	RTS
8	Clear To Send	CTS
9	Ring Indicator	RI

Those terminals highlighted in **bold** font represent those connections absolutely essential for any EIA/TIA-232 link to function. The other terminals carry optional "handshaking" ("flow control") signals[4] specified for the purpose of coordinating data transactions (these are the OSI layer 2 elements of the EIA/TIA-232 standard).

For *DCE* (Data Communications Equipment[5]) devices such as modems, which extend the EIA/TIA-232 signal path onward to other devices, the assignments of all transmitting and receiving pins are swapped. For example, pin 2 is the Transmitted Data (TD) output while pin 3 is the Received Data (RD) input on a DCE device; similarly pin 7 is CTS and pin 8 is RTS on a DCE device, while pin 4 is DSR and pin 6 DTR. Pins 1, 5, and 9 are identically assigned for DTE and

[3] Often (incorrectly) called a "DB-9" connector. The second letter of the "D" connector designation refers to the physical dimensions of the metal "shell" outlining the pins. A "DB" shell is significantly wider than a "DE" shell. DB-25 connectors were initially the connector of choice for serial communication on personal computers, and when the smaller 9-pin connectors became more popular people started calling them "DB-9" assuming the numerical value would be the only change in the designator.

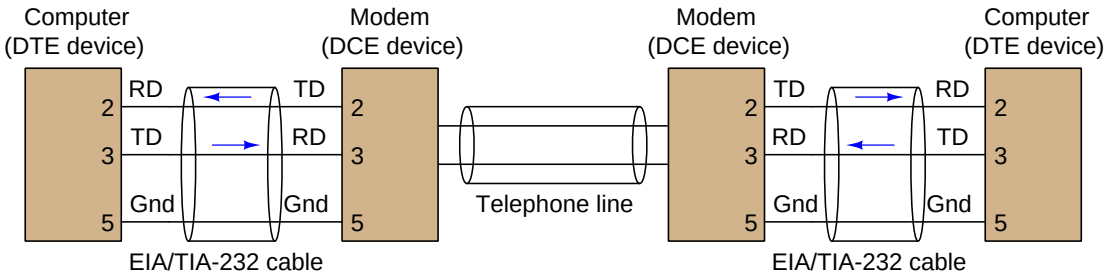
[4] The way hardware-based flow control works in the EIA/TIA-232 standard involves two lines labeled RTS ("Request To Send") and CTS ("Clear To Send") connecting the two devices together on a point-to-point serial network in addition to the TD ("Transmitted Data") and RD ("Received Data") and signal ground lines. Like the TD and RD terminals which must be "crossed over" between devices such that the TD of one device connects to the RD of the other device and vice-versa, the RTS and CTS terminals of the two devices must be similarly crossed. The RTS is an output line while the CTS is an input, on both devices. When a device is able to receive data, it activates its RTS output line to *request* data. A device is not permitted to transmit data on its TD line until it is *cleared* to send data by an active state on its CTS input line.

[5] Also known by the unwieldy acronym *DCTE* (Data Circuit Terminating Equipment). Just think of "DTE" devices as being at the very end ("terminal") of the line, whereas "DCE" devices are somewhere in the middle, helping to exchange serial data between DTE devices.

DCE devices. This allows straight pin-to-pin cable connections between the DTE and DCE devices, so the transmit pin of the DTE device connects to the receive pin of the DCE, and vice-versa.

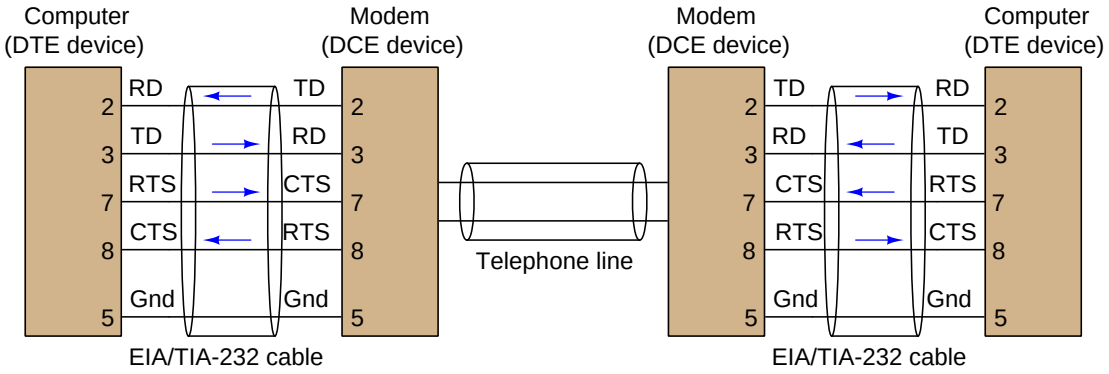
The following diagram shows the minimum cable requirements for an EIA/TIA-232 serial communication link consisting of a pair of DTEs connecting through a pair of DCEs. This minimal point-to-point network assumes the devices are configured for either *software* flow control (i.e. digital codes send over the TD/RD lines requesting the transmitting device to halt and resume) or no flow control at all:

EIA/TIA-232 communication using software flow control



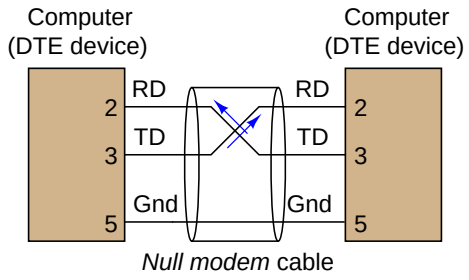
In order to utilize hardware flow control, the EIA/TIA-232 cable(s) must include two additional conductors connecting the RTS and CTS pins between devices to enable them to signal each other with voltage-level states declaring when they are ready to receive more data:

EIA/TIA-232 communication using hardware flow control



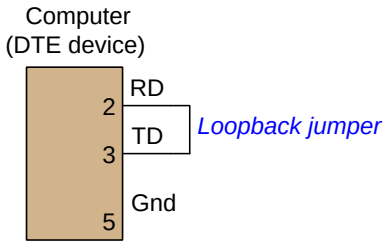
Improper flow control configuration is a common commissioning problem in new serial networks. If the devices are configured for hardware handshaking (i.e. necessitating RTS and CTS lines in the connecting cables) or those devices simply default to hardware handshaking as new, but cables lacking RTS-to-CTS lines are used between these devices, the devices will never communicate because their respective (floating) CTS inputs will remain in the idle state and therefore the devices "think" they do not have permission to send data. *Hardware flow control necessitates serial cables with at least five conductors, while software flow control needs only three* (TD, RD, and Ground).

If one desires to directly connect two DTE devices together using EIA/TIA-232, a special cable called a *null modem* must be used, which swaps the connections between pins 2 and 3 of each device. A "null modem" connection is necessary for the transmit pin of each DTE device to connect to the receive pin of the other DTE device:



The concept of a "null modem" is not unique to EIA/TIA-232 circuits[6]. Any communications standard where the devices have separate "transmit" and "receive" channels will require a "null modem" connection with transmit and receive channels swapped to be able to communicate directly without the benefit of interconnecting DCE devices. Four-wire EIA/TIA-485 and Ethernet over twisted-pair wiring are two other examples of digital communication standards where a "null" style cable is required for two DTE devices to directly connect.

Similarly, if we wish to have an EIA/TIA-232 communicate with itself, we may connect a *loopback* jumper wire from its TD pin to its RD pin as such:



For example, if the device in question is a terminal with keyboard and viewing monitor, the loopback connection will ensure that every keystroke will be instantly "echoed" to the monitor. This is useful for diagnostic testing purposes.

EIA/TIA-232 networks may be simple, but they tend to be rather limited both in data bit rate and distance, those two parameters being inversely related. References to the EIA/TIA-232 standard repeatedly cite a maximum data rate of 19.2 kbps at 50 feet cable rate. Experimental tests[7] suggest

[6] In fact, the concept is not unique to digital systems at all. Try talking to someone using a telephone handset held upside-down, with the speaker near your mouth and the microphone hear your ear, and you will immediately understand the necessity of having "transmit" and "receive" channels swapped from one end of a network to the other!

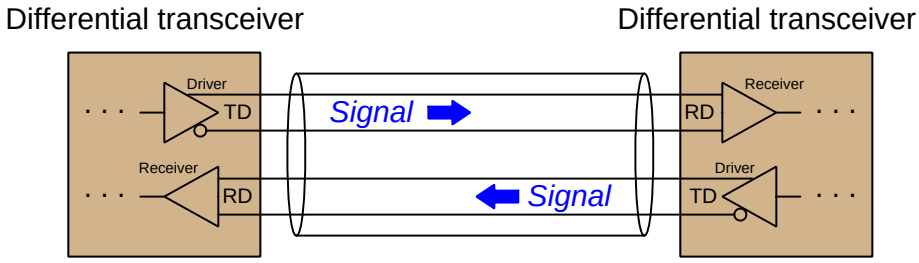
[7] Once I experimented with the fastest data rate I could "push" an EIA/TIA-232 network to, using a "flat" (untwisted, unshielded pair) cable less than ten feet long, and it was 192 kbps with occasional data corruptions. Park, Mackay, and Wright, in their book *Practical Data Communications for Instrumentation and Control* document cable lengths as long as 20 meters at 115 kbps for EIA/TIA-232, and 50 *meters* (over 150 feet!) at 19.2 kbps: over three

greater rate/distance combinations may be possible in optimum conditions (low cable capacitance, minimum noise, good grounding). Since EIA/TIA-232 was developed to connect peripheral devices to computers (typically within the physical span of one room), and at modest speeds, neither of these limitations were significant to its intended application.

3.3 EIA/TIA-422 and EIA/TIA-485

The next two network standards[8] are less comprehensive than EIA/TIA-232, specifying only the electrical characteristics of signaling without any regard for connector types or any layer 2 (handshaking) considerations. Within these domains, the 422 and 485 standards differ significantly from 232, their designs intended to optimize both maximum cable length and maximum data rate.

To begin with, the electrical signaling used for both EIA/TIA-422 and EIA/TIA-485 is *differential* rather than single-ended (*balanced* rather than unbalanced). This means a dedicated *pair* of wires is used for each communications channel rather than a single wire whose voltage is referenced to a common ground point as is the case with EIA/TIA-232. It also requires *driver* and *receiver* interface circuits to translate ground-referenced digital logic signals to the differential signals used for communication:

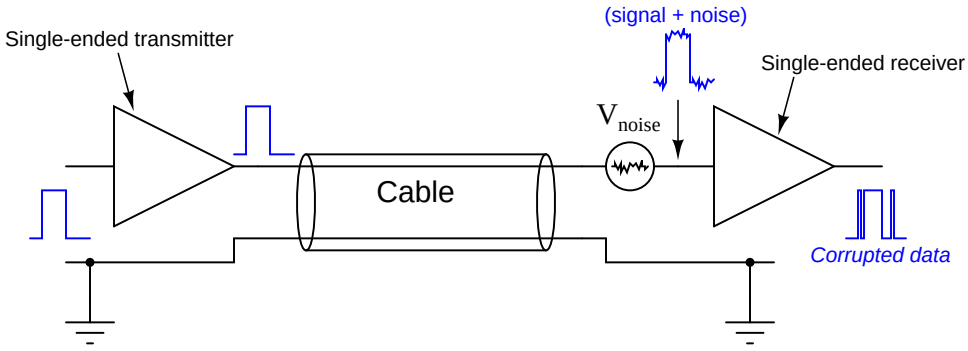


Using dedicated wire pairs instead of single conductors sharing a common ground means that EIA/TIA-422 and EIA/TIA-485 networks enjoy much greater immunity to induced noise than EIA/TIA-232. Noise induced via electrostatic coupling along the length of a network cable tends to be fairly equal (i.e. common-mode) on all non-grounded conductors of that cable, but since the receivers in EIA/TIA-422 and EIA/TIA-485 networks response only to differential voltages (not common-mode voltages), induced noise is ignored.

times better than the advertised EIA/TIA-232 standard.

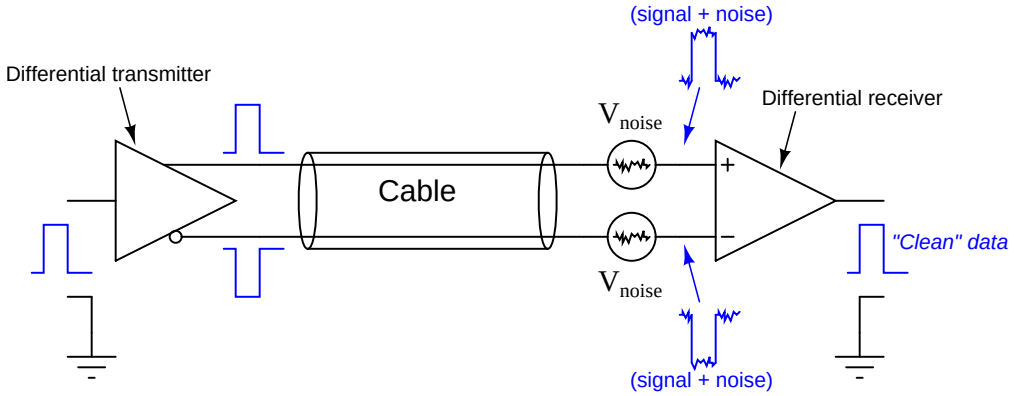
[8] Former labels for EIA/TIA-422 and EIA/TIA-485 were RS-422 and RS-485, respectively. These older labels persist even today, to the extent that some people will not recognize what you are referring to if you say "EIA/TIA-422" or "EIA/TIA-485."

The advantage differential signaling enjoys over single-ended signaling may be understood by graphical comparison. The first illustration shows how electrical noise imposed on the ungrounded conductor of a simplex communications cable becomes superimposed on the digital data signal, detected at the receiving end. Noise is modeled here as a voltage source in series along the ungrounded conductor, near the receiving end. In reality, it is more likely to be distributed along the bulk of the cable length:



If the superimposed noise voltage detected at the receiver has sufficient peak-to-peak amplitude to push the signal voltage above or below critical threshold levels, the receiver will interpret this as a change of digital state and cause corruptions in the data stream.

By contrast, any noise superimposed on ungrounded conductors in a differential signaling circuit cancel at the receiver, because the close proximity of those two conductors ensures any induced noise will be the same. Since the receiver responds only to *differential* voltage between its two inputs, this common-mode noise cancels, revealing a "clean" data signal at the end:



Both EIA/TIA-422 and EIA/TIA-485 systems use differential signaling, allowing them to operate over much longer cable lengths at much greater cable speeds than EIA/TIA-232 which is single-ended. Other high-speed network standards including Ethernet and USB (Universal Serial Bus) use differential signaling as well.

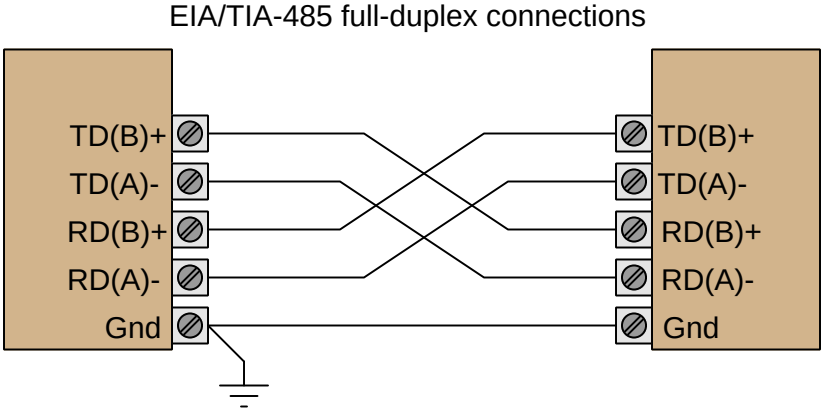
EIA/TIA-422 is a *simplex* (one-way) communications standard, whereas EIA/TIA-485 is a *duplex* (two-way) standard. Both support more than two devices on a network segment. With EIA/TIA-422, this means one transmitter and multiple receivers. With EIA/TIA-485, this may include multiple *transceivers* (devices capable of both transmitting and receiving at different times: half-duplex). Four wires are necessary to connect two such devices when full-duplex (simultaneous two-way communication) is required, and full-duplex is only practical between two devices.

EIA/TIA-422 and EIA/TIA-485 specify positive and negative voltage differences (measured between each dedicated wire pair) for its signaling, both for transmitting devices as well as receiving devices. A receiver must recognize any signal more negative than −200 milliVolts as a "mark" (1) and any signal more positive than +200 milliVolts as a "space" (0). These voltage thresholds are much lower than those specified for EIA/TIA-232 (± 3 Volts) due to the relative lack of noise on differential signal lines compared to ground-referenced signal lines. Simply put, less noise voltage on the lines means the signal doesn't have to be as strong to "swamp" that noise and be reliably detected at the receiver. EIA/TIA-422 transmitters ("drivers") are supposed to generate −2 and +2 Volt signals (minimum amplitude) to ensure at least 1.8 Volts of noise margin between transmitter and receiver. EIA/TIA-485 drivers are allowed a smaller noise margin, with the minimum signal levels being −1.5 Volts and +1.5 Volts.

The maximum recommended cable length for both EIA/TIA-422 and EIA/TIA-485 networks is 1200 meters, which is greater than half a mile[9]. The maximum data rate is inversely dependent on cable length (just as it is for EIA/TIA-232), but substantially greater owing to the noise immunity of differential signaling. With the long cable lengths and higher data rates made possible by differential signaling, some applications may require *terminating resistors* to eliminate reflected signals. Experiments conducted by Texas Instruments demonstrate acceptable signal integrity at 200 kbps over a cable 100 feet long with no termination resistors. With a termination resistor at the receiver input (for simplex data transmission) in place on the same 100 foot cable, a data rate of 1 Mbps was achieved.

[9] 1200 meters is the figure commonly cited in technical literature. However, Park, Mackay, and Wright, in their book *Practical Data Communications for Instrumentation and Control* document EIA/TIA-422 and EIA/TIA-485 networks operating with cable lengths up to 5 km (over 16000 feet!) at data rates of 1200 bps. Undoubtedly, such systems were installed with care, using high-quality cable and good wiring practices to minimize cable capacitance and noise.

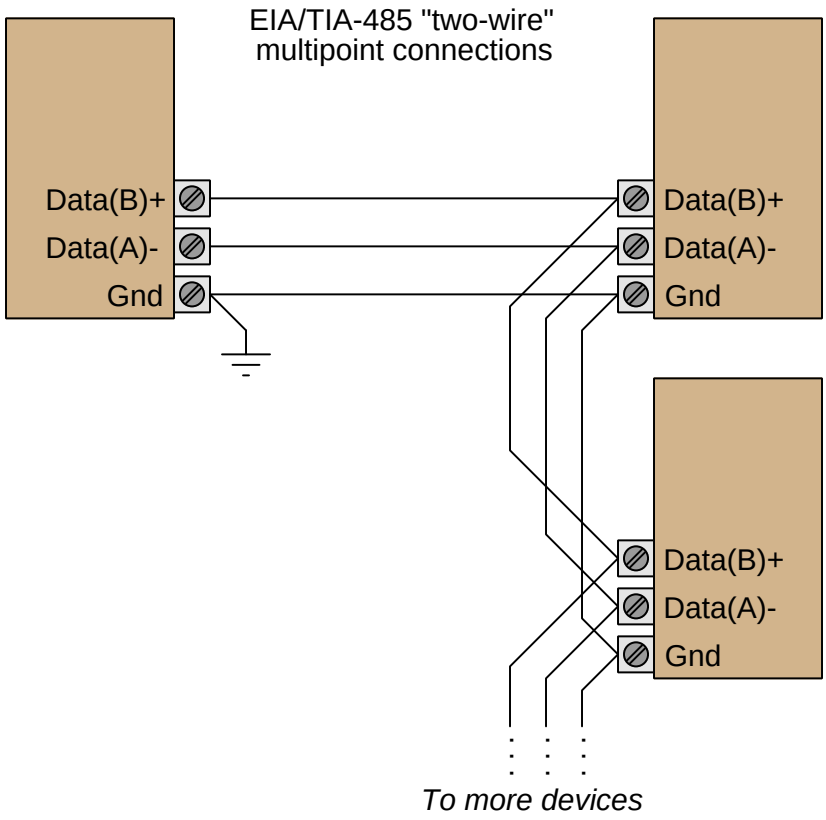
Due to the lack of standardization for cable connectors in EIA/TIA-422 and EIA/TIA-485 networks, there are no established pin numbers or labels for the differential transmit and receive conductors. A common convention seen in industrial devices, though, are the labels "A" and "B", alternatively labeled "−" and "+" or "A−" and "B+" in honor of their idle-state polarities (the "mark" or "1" state). In a 4-wire EIA/TIA-485 network, where full-duplex operation is possible, the terminals and connections will look something like this:



The good news with regard to 422/485 terminal labeling is that you will not harm the electronics by accidently connecting the wires with incorrect polarity. If, for example, you cannot get a 422/485 receiver to acknowledge data sent by a 422/485 transmitter, you may try swapping the polarity (A/B, or +/− connections) without risking damage to either device and see if that fixes the problem.

Note the use of a ground conductor connecting both devices together. Even though the data signaling is differential and therefore does not theoretically require a common ground connection (since common-mode voltage is ignored), a ground connection helps ensure the common-mode voltage does not become excessive, since *real* receiver circuits have practical limits on the amount of common-mode voltage they can tolerate.

A popular connection scheme for EIA/TIA-485 half-duplex operation is where the Transmitted Data (TD) and Received Data (RD) terminal pairs are combined, so that two-way communication may occur over one pair of wires. With such devices, it is customary to label the terminals simply as "Data" (A− and B+):

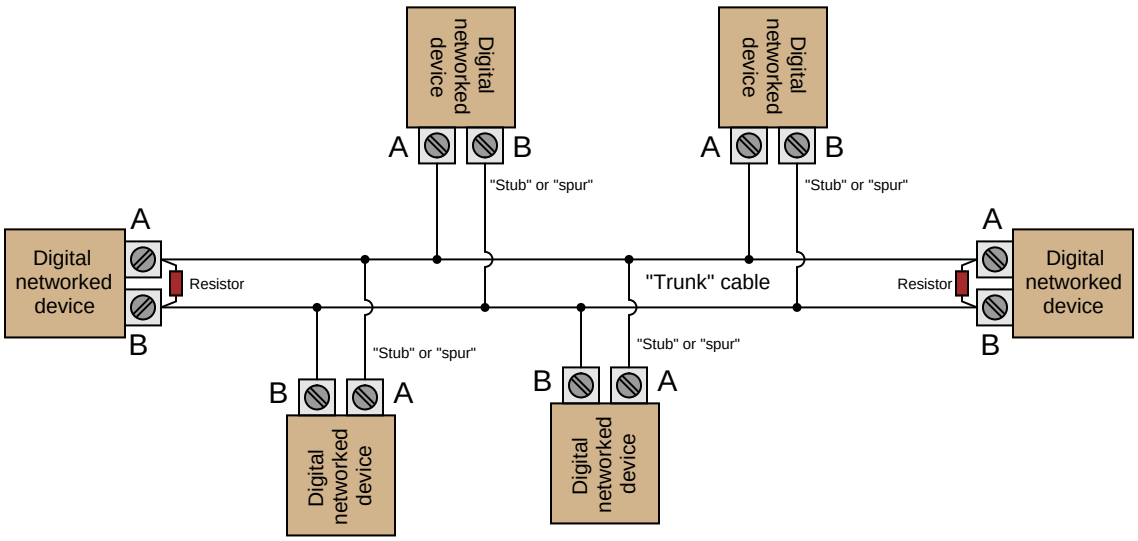


The possibility of half-duplex operation raises the question of channel arbitration and device addressing, but since the EIA/TIA-485 standard does not specify anything outside of layer 1 concerns, these matters are left to other networking standards to fulfill. In other words, EIA/TIA-485 is not a complete data communications standard, but merely serves as the layer 1 component of other standards such as Allen-Bradley's *Data Highway* (DH), Opto 22's *Optomux*, and others.

Given the potential for high-speed communication along lengthy runs of cable using EIA/TIA-422 or EIA/TIA-485, the potential necessity of terminating resistors to prevent signal "reflection" is very real. Networks operating with short cables, and/or slow data rates, may work just fine without termination resistors[10]. However, the effects of reflected signals grows more pronounced as the reflection time (time-of-flight for the signal to travel "round-trip" from one end of the cable to the other and back) approaches a substantial fraction of the bit time.

[10] In fact, a great many EIA/TIA-485 networks in industry operate "unterminated" with no problems at all.

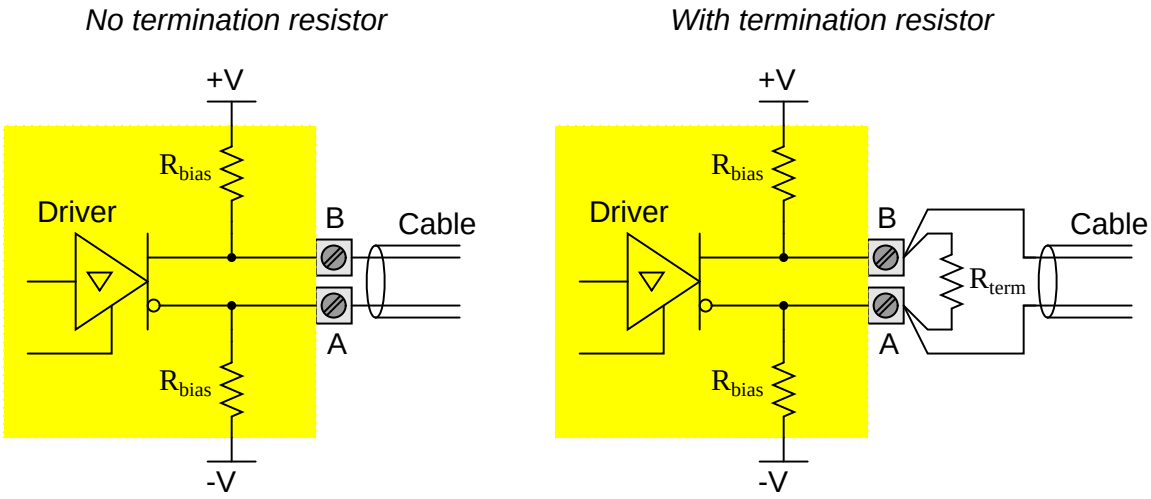
No network should have more than two termination resistors, one at each (far) end, and care should be taken to limit the lengths of all cable "stubs" or "spurs" branching off of the main "trunk" cable:



The proper value for these resistors, of course, is equality with the characteristic impedance of the cable itself. A termination resistor value greater than the cable's surge impedance will still allow positive reflections of limited amplitude, while a termination resistor value less than the cable's surge impedance will still allow negative reflections of limited amplitude.

However, the inclusion of resistive loads to an EIA/TIA-422 or EIA/TIA-485 network may cause other problems. Many devices use a pair of internal *biasing resistors* to establish the "mark" state necessary for idle conditions, connecting the "A" terminal to the negative supply voltage rail through a resistor and the "B" terminal to the positive supply voltage rail through another resistor. Connecting a terminating resistor between terminals "A" and "B" will alter the voltage levels normally provided by these biasing resistors, consequently causing problems.

The following schematic diagram shows the equivalent circuit of an EIA/TIA-485 transmitter[11], with and without a terminating resistor connected:



When the driver is in high-impedance (High-Z) mode, the "idle" state of the wire pair will be established by the bias resistors (equal to the supply voltage so long as there is no loading). However, a terminating resistor will act as a DC load to this biasing network, causing a substantial reduction of the "idle" state voltage toward 0 Volts. Recall that −200 milliVolts was the receiving threshold value for a "mark" state in both EIA/TIA-422 and EIA/TIA-485 standards (terminal "A" negative and terminal "B" positive). If the presence of a terminating resistor[12] reduces the idle state voltage to less than 200 milliVolts absolute, the receiver(s) will not be able to reliably read the network's idle state and communication errors will result.

Thus, we see that the inclusion of any terminating resistors must be accompanied by an analysis of the devices' bias resistor networks if we are to ensure robust network operation. It is foolhardy to simply attach terminating resistors to an EIA/TIA-422 or EIA/TIA-485 network without considering their combined effect on biasing.

[11] A full EIA/TIA-485 *transceiver* would also have a differential receiving amplifier paralleled to the same pair of A/B terminals. That is not shown here since it's not necessarily relevant. It could also be that these two terminals are just two out of four for a four-terminal EIA/TIA-485 device with separate TD and RD A/B pairs.

[12] Actually *two* terminating resistors in parallel, since one with be at each end of the cable! The actual DC biasing network will be more complicated as well if more than one device has its own set of internal bias resistors.

Chapter 4

Derivations and Technical References

This chapter is where you will find mathematical derivations too detailed to include in the tutorial, and/or tables and other technical reference material.

4.1 ASCII character codes

ASCII characters consist of seven-bit digital words. The following table shows all 128 possible combinations of these seven bits, from 0000000 (ASCII "NUL" character) to 1111111 (ASCII "DEL" character). For ease of organization, this table's columns represent the most-significant three bits of the seven-bit word, while the table's rows represent the least-significant four bits. For example, the capital letter "C" would be encoded as 1000011 in the ASCII standard.

↓ LSB / MSB →	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	”	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	−	=	M	]	m	}
1110	SO	RS	.	>	N	ˆ	n	˜
1111	SI	US	/	?	O	_	o	DEL

It is worth noting that the ASCII codes for the Arabic numerals 0 through 9 are simply the four-bit binary representation of those numbers preceded by 011. For example, the number six (0110) is represented in ASCII as 0110110; the number three (0011) in ASCII as 0110011; etc. This is useful to know, for example, if you need to program a computer to convert single decimal digits to their corresponding ASCII codes: just take each four-bit numerical value and add forty-eight (0x30 in hexadecimal) to it.

4.2 The OSI Reference Model

Digital communication is a fairly complex topic, and there are a great many ways in which one digital device might communicate information to another. In order to help describe how different methods (protocols) function in forming a complete communication system, the *OSI Reference Model* was developed. This is not a standard itself, but rather a taxonomical system for *describing* and *classifying* various aspects of any communications system.

For example, one communication system might use discrete voltage pulses in a non-return-to-zero (NRZ) format to represent 0 and 1 bit states, while another might use analog audio tone frequency voltage signals to do the same. According to the OSI model, the choice over electrical signal types is a *Layer 1* matter. The patterns in which bit states make up larger serial data frames is another matter which must be standardized in any communication system, and according to the OSI model that is the domain of *Layer 2*. At the highest level of the OSI model we have *Layer 7* which regards the practical meaning of the bits communicated, whether those be alphanumeric codes representing an internet web page's contents, Modbus control functions, or anything else.

Layer 7 **Application**	This is where digital data takes on practical meaning in the context of some human or overall system function. *Examples: HTTP, FTP, HART, Modbus*
Layer 6 **Presentation**	This is where data gets converted between different formats. *Examples: ASCII, EBCDIC, MPEG, JPG, MP3*
Layer 5 **Session**	This is where "conversations" between digital devices are opened, closed, and otherwise managed for reliable data flow. *Examples: Sockets, NetBIOS*
Layer 4 **Transport**	This is where complete data transfer is handled, ensuring all data gets put together and error-checked before use. *Examples: TCP, UDP*
Layer 3 **Network**	This is where the system determines network-wide addresses, ensuring a means for data to get from one node to another. *Examples: IP, ARP*
Layer 2 **Data link**	This is where basic data transfer methods and sequences (frames) are defined within the smallest segment(s) of a network. *Examples: CSMA/CD, Token passing, Master/Slave*
Layer 1 **Physical**	This is where data bits are equated to electrical, optical, or other signals. Other physical details such as cable and connector types are also specified here. *Examples: EIA/TIA-232, 422, 485, Bell 202*

Students of philosophy might see a parallel between the OSI model's description of digital communication versus the ancient Greek philosopher Aristotle's classification of *causes*. Aristotle proposed four different ways to describe what causes a thing to exist or to occur. I have arranged these four causes in an order analogous to the OSI model's layers:

Aristotle's Four Causes

Layer 4 **Final**	This is the end-purpose for which something exists or has happened.
Layer 3 **Efficient**	This is the means by which something came into existence or occurred.
Layer 2 **Formal**	This is the structure or form taken by something.
Layer 1 **Material**	This is the fundamental substance something is made of.

For example, we might classify the four causes of a textbook as follows:

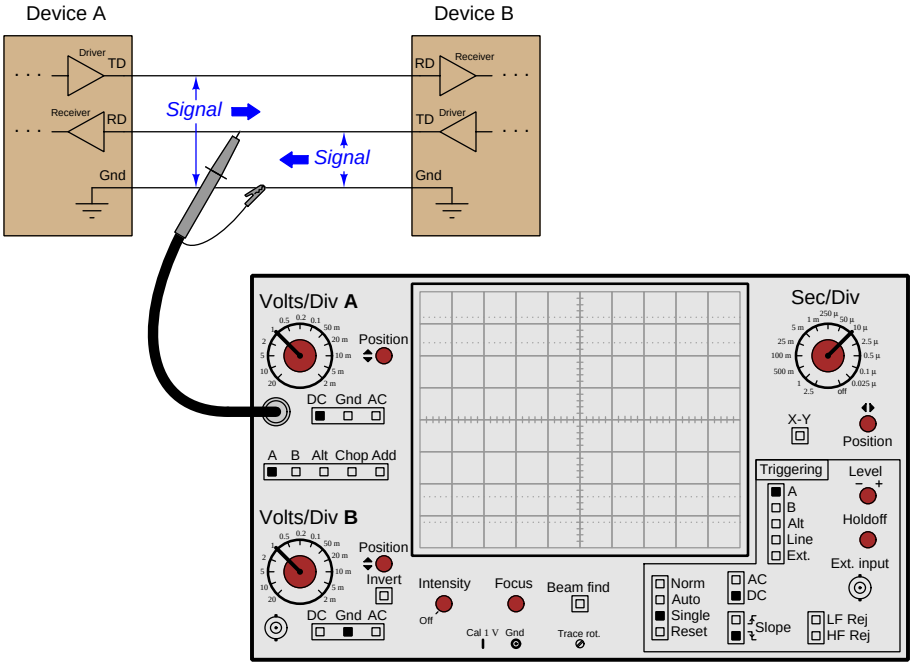
- **Final cause** – to educate and enlighten the reader
- **Efficient cause** – a human author wrote the book, and it came into existence through the operation of a printing press
- **Formal cause** – a series of characters and images arranged into words, paragraphs, diagrams, graphs, equations, etc. following one or more standards of language (e.g. English, international math notation, Cartesian-coordinate graphs)
- **Material cause** – paper and ink

All of these causes describe different aspects of what was required for a book to come into existence. Similarly, with the OSI model, each level of the model is like one of Aristotle's causes in that it describes something necessary for the communication to take place.

4.3 Using an oscilloscope on differential networks

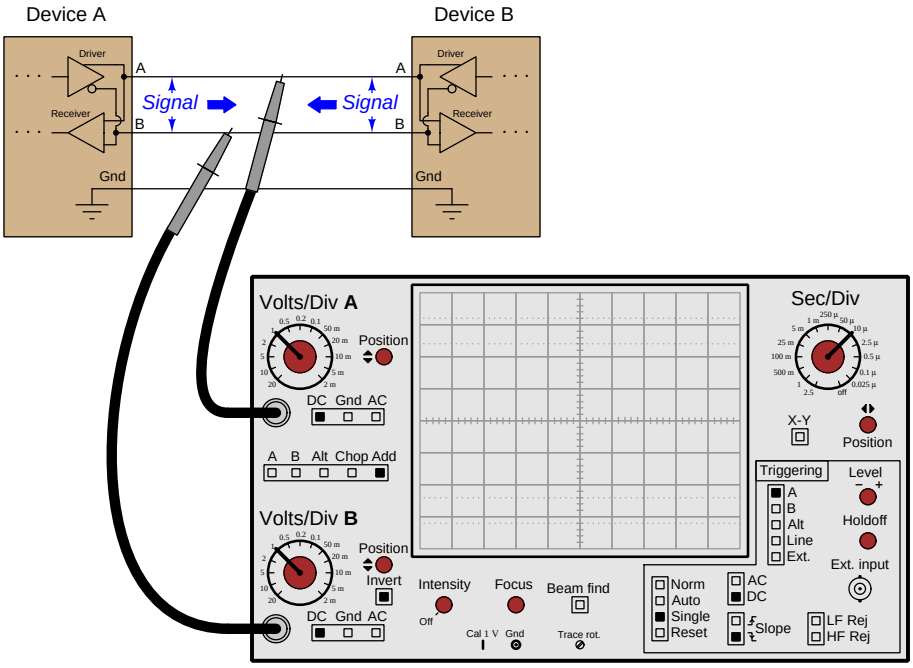
Data communication networks based on differential voltage signals such as EIA/TIA-422 or EIA/TIA-485 or twisted-pair Ethernet may be monitored using a digital-storage oscilloscope, capturing data frames for close analysis of content as well as signal integrity (e.g. proper voltage levels, tolerable noise levels, etc.). However, unlike EIA/TIA-232 which is "single-ended" with its voltage signals all referenced to a common "ground" location, we must be careful in how we connect an oscilloscope to a differential network. This is because oscilloscopes by default use ground as the reference point for all their voltage measurements. The alligator-clip portion of an oscilloscope probe is connected to the chassis of the oscilloscope itself, which is in turn connected to Earth ground through the oscilloscope's power cord.

To compare, let us first examine how an oscilloscope would be connected to one of the lines of an EIA/TIA-232 network, to capture a data frame sent by Device B and received by Device A:



We connect the two terminals of the oscilloscope's probe between the two wires conveying the voltage signal we wish to measure, in this case between one data wire and the ground wire.

Next, we will contrast the connections made to a two-wire EIA/TIA-485 network:



Since this is a *balanced* or *differential* network, neither the "A" nor the "B" lines are grounded. In order to maintain the electrical integrity of this network, we must be sure not to force either of these terminals to Earth ground by the connection of our oscilloscope, which means we cannot connect the probe's ground clip to either wire as we could in the EIA/TIA-232 network. Instead, we must configure the oscilloscope to measure differential voltage, which means using two channels with the display showing the sum of those two channels with one of them inverted (i.e. $A + (-B)$ = difference of potential between wire A and wire B).

Chapter 5

Questions

This learning module, along with all others in the ModEL collection, is designed to be used in an inverted instructional environment where students independently read[1] the tutorials and attempt to answer questions on their own *prior* to the instructor's interaction with them. In place of lecture[2], the instructor engages with students in Socratic-style dialogue, probing and challenging their understanding of the subject matter through inquiry.

Answers are not provided for questions within this chapter, and this is by design. Solved problems may be found in the Tutorial and Derivation chapters, instead. The goal here is *independence*, and this requires students to be challenged in ways where others cannot think for them. Remember that you always have the tools of *experimentation* and *computer simulation* (e.g. SPICE) to explore concepts!

The following lists contain ideas for Socratic-style questions and challenges. Upon inspection, one will notice a strong theme of *metacognition* within these statements: they are designed to foster a regular habit of examining one's own thoughts as a means toward clearer thinking. As such these sample questions are useful both for instructor-led discussions as well as for self-study.

[1] Technical reading is an essential academic skill for any technical practitioner to possess for the simple reason that the most comprehensive, accurate, and useful information to be found for developing technical competence is in textual form. Technical careers in general are characterized by the need for continuous learning to remain current with standards and technology, and therefore any technical practitioner who cannot read well is handicapped in their professional development. An excellent resource for educators on improving students' reading prowess through intentional effort and strategy is the book textitReading For Understanding – How Reading Apprenticeship Improves Disciplinary Learning in Secondary and College Classrooms by Ruth Schoenbach, Cynthia Greenleaf, and Lynn Murphy.

[2] Lecture is popular as a teaching method because it is easy to implement: any reasonably articulate subject matter expert can talk to students, even with little preparation. However, it is also quite problematic. A good lecture always makes complicated concepts seem easier than they are, which is bad for students because it instills a false sense of confidence in their own understanding; reading and re-articulation requires more cognitive effort and serves to verify comprehension. A culture of teaching-by-lecture fosters a debilitating dependence upon direct personal instruction, whereas the challenges of modern life demand independent and critical thought made possible only by gathering information and perspectives from afar. Information presented in a lecture is ephemeral, easily lost to failures of memory and dictation; text is forever, and may be referenced at any time.

General challenges following tutorial reading

- Summarize as much of the text as you can in one paragraph of your own words. A helpful strategy is to explain ideas as you would for an intelligent child: as simple as you can without compromising too much accuracy.
- Simplify a particular section of the text, for example a paragraph or even a single sentence, so as to capture the same fundamental idea in fewer words.
- Where did the text make the most sense to you? What was it about the text's presentation that made it clear?
- Identify where it might be easy for someone to misunderstand the text, and explain why you think it could be confusing.
- Identify any new concept(s) presented in the text, and explain in your own words.
- Identify any familiar concept(s) such as physical laws or principles applied or referenced in the text.
- Devise a proof of concept experiment demonstrating an important principle, physical law, or technical innovation represented in the text.
- Devise an experiment to disprove a plausible misconception.
- Did the text reveal any misconceptions you might have harbored? If so, describe the misconception(s) and the reason(s) why you now know them to be incorrect.
- Describe any useful problem-solving strategies applied in the text.
- Devise a question of your own to challenge a reader's comprehension of the text.

General follow-up challenges for assigned problems

- Identify where any fundamental laws or principles apply to the solution of this problem, especially before applying any mathematical techniques.
- Devise a thought experiment to explore the characteristics of the problem scenario, applying known laws and principles to mentally model its behavior.
- Describe in detail your own strategy for solving this problem. How did you identify and organized the given information? Did you sketch any diagrams to help frame the problem?
- Is there more than one way to solve this problem? Which method seems best to you?
- Show the work you did in solving this problem, even if the solution is incomplete or incorrect.
- What would you say was the most challenging part of this problem, and why was it so?
- Was any important information missing from the problem which you had to research or recall?
- Was there any extraneous information presented within this problem? If so, what was it and why did it not matter?
- Examine someone else's solution to identify where they applied fundamental laws or principles.
- Simplify the problem from its given form and show how to solve this simpler version of it. Examples include eliminating certain variables or conditions, altering values to simpler (usually whole) numbers, applying a limiting case (i.e. altering a variable to some extreme or ultimate value).
- For quantitative problems, identify the real-world meaning of all intermediate calculations: their units of measurement, where they fit into the scenario at hand. Annotate any diagrams or illustrations with these calculated values.
- For quantitative problems, try approaching it qualitatively instead, thinking in terms of "increase" and "decrease" rather than definite values.
- For qualitative problems, try approaching it quantitatively instead, proposing simple numerical values for the variables.
- Were there any assumptions you made while solving this problem? Would your solution change if one of those assumptions were altered?
- Identify where it would be easy for someone to go astray in attempting to solve this problem.
- Formulate your own problem based on what you learned solving this one.

General follow-up challenges for experiments or projects

- In what way(s) was this experiment or project easy to complete?
- Identify some of the challenges you faced in completing this experiment or project.

- Show how thorough documentation assisted in the completion of this experiment or project.
- Which fundamental laws or principles are key to this system's function?
- Identify any way(s) in which one might obtain false or otherwise misleading measurements from test equipment in this system.
- What will happen if (component X) fails (open/shorted/etc.)?
- What would have to occur to make this system unsafe?

5.1 Conceptual reasoning

These questions are designed to stimulate your analytic and synthetic thinking[3]. In a Socratic discussion with your instructor, the goal is for these questions to prompt an extended dialogue where assumptions are revealed, conclusions are tested, and understanding is sharpened. Your instructor may also pose additional questions based on those assigned, in order to further probe and refine your conceptual understanding.

Questions that follow are presented to challenge and probe your understanding of various concepts presented in the tutorial. These questions are intended to serve as a guide for the Socratic dialogue between yourself and the instructor. Your instructor's task is to ensure you have a sound grasp of these concepts, and the questions contained in this document are merely a means to this end. Your instructor may, at his or her discretion, alter or substitute questions for the benefit of tailoring the discussion to each student's needs. The only absolute requirement is that each student is challenged and assessed at a level equal to or greater than that represented by the documented questions.

It is far more important that you convey your *reasoning* than it is to simply convey a correct answer. For this reason, you should refrain from researching other information sources to answer questions. What matters here is that *you* are doing the thinking. If the answer is incorrect, your instructor will work with you to correct it through proper reasoning. A correct answer without an adequate explanation of how you derived that answer is unacceptable, as it does not aid the learning or assessment process.

You will note a conspicuous lack of answers given for these conceptual questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your conceptual answers, where applicable, is to use circuit simulation software to explore the effects of changes made to circuits. For example, if one of these conceptual questions challenges you to predict the effects of altering some component parameter in a circuit, you may check the validity of your work by simulating that same parameter change within software and seeing if the results agree.

[3] *Analytical* thinking involves the "disassembly" of an idea into its constituent parts, analogous to dissection. *Synthetic* thinking involves the "assembly" of a new idea comprised of multiple concepts, analogous to construction. Both activities are high-level cognitive skills, extremely important for effective problem-solving, necessitating frequent challenge and regular practice to fully develop.

5.1.1 Reading outline and reflections

"Reading maketh a full man; conference a ready man; and writing an exact man" – Francis Bacon

Francis Bacon's advice is a blueprint for effective education: reading provides the learner with knowledge, writing focuses the learner's thoughts, and critical dialogue equips the learner to confidently communicate and apply their learning. Independent acquisition and application of knowledge is a powerful skill, well worth the effort to cultivate. To this end, students should read these educational resources closely, journal their own reflections on the reading, and discuss in detail their findings with classmates and instructor(s). You should be able to do all of the following after reading any instructional text:

☑ Briefly SUMMARIZE THE TEXT in the form of a journal entry documenting your learning as you progress through the course of study. Share this summary in dialogue with your classmates and instructor. Journaling is an excellent self-test of thorough reading because you cannot clearly express what you have not read or did not comprehend.

☑ Demonstrate ACTIVE READING STRATEGIES, including verbalizing your impressions as you read, simplifying long passages to convey the same ideas using fewer words, annotating text and illustrations with your own interpretations, working through mathematical examples shown in the text, cross-referencing passages with relevant illustrations and/or other passages, identifying problem-solving strategies applied by the author, etc. Technical reading is a special case of problem-solving, and so these strategies work precisely because they help solve any problem: paying attention to your own thoughts (metacognition), eliminating unnecessary complexities, identifying what makes sense, paying close attention to details, drawing connections between separated facts, and noting the successful strategies of others.

☑ Identify IMPORTANT THEMES, especially GENERAL LAWS and PRINCIPLES, expounded in the text and express them in the simplest of terms as though you were teaching an intelligent child. This emphasizes connections between related topics and develops your ability to communicate complex ideas to anyone.

☑ Form YOUR OWN QUESTIONS based on the reading, and then pose them to your instructor and classmates for their consideration. Anticipate both correct and incorrect answers, the incorrect answer(s) assuming one or more plausible misconceptions. This helps you view the subject from different perspectives to grasp it more fully.

☑ Devise EXPERIMENTS to test claims presented in the reading, or to disprove misconceptions. Predict possible outcomes of these experiments, and evaluate their meanings: what result(s) would confirm, and what would constitute disproof? Running mental simulations and evaluating results is essential to scientific and diagnostic reasoning.

☑ Specifically identify any points you found CONFUSING. The reason for doing this is to help diagnose misconceptions and overcome barriers to learning.

5.1.2 Foundational concepts

Correct analysis and diagnosis of electric circuits begins with a proper understanding of some basic concepts. The following is a list of some important concepts referenced in this module's full tutorial. Define each of them in your own words, and be prepared to illustrate each of these concepts with a description of a practical example and/or a live demonstration.

Digital signal

Bit

Noise immunity

NRZ encoding

Bit rate

Baud rate

Data frame

Synchronous versus asynchronous communication

Parity

UART

Flow control

OSI model

Single-ended voltage signal

Differential voltage signal

Null modem

DTE

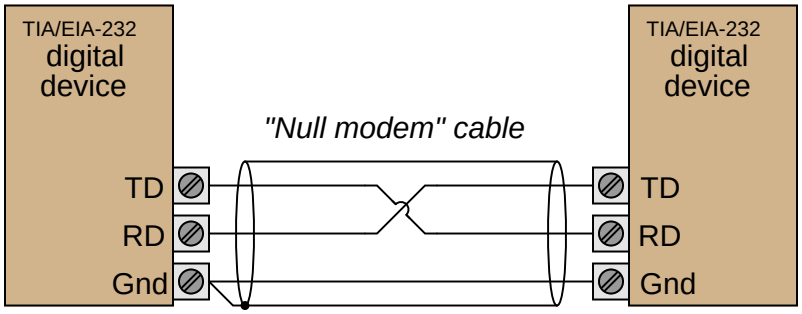
DCE

Simplex

Duplex

5.1.3 Minimalist EIA/TIA-232 system

Examine the following minimalist EIA-TIA-232 serial data network and answer the following questions:



- Is this system capable of *simplex*, *half-duplex*, or *full-duplex* operation?
- Explain why the shield wire is connected to the Ground terminal only at one end, and not at both ends.
- Explain what the term "null modem" means with reference to the cable connecting these two devices together, and why it is important we use this special cable for the job?

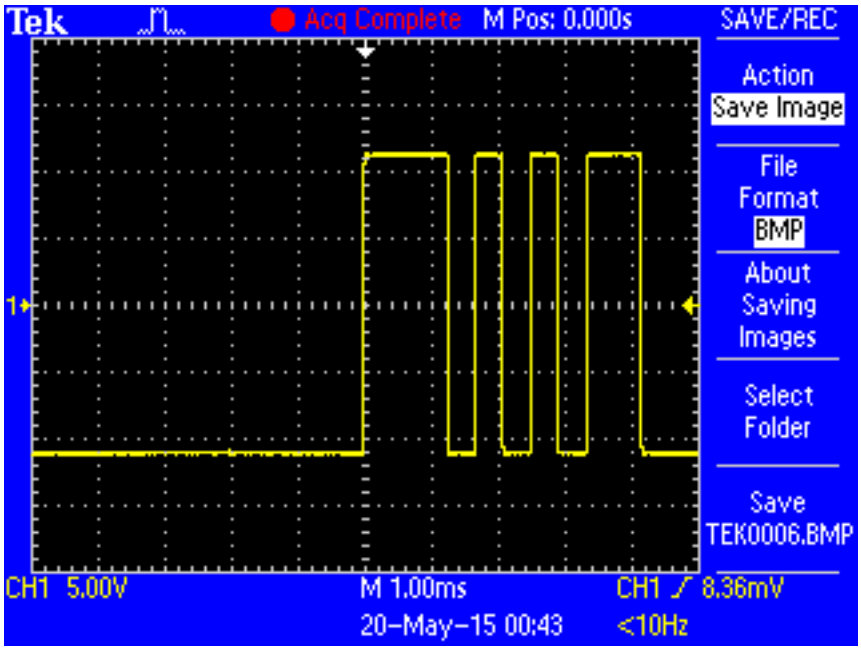
Challenges

- Identify a situation where we could get two serial devices to communicate with each other while using a "straight-through" cable rather than a "null modem" cable.
- Suppose we needed to verify the communication of data in this system, and the only piece of test equipment we had was a digital multimeter (DMM). Explain how we could use a DMM to sense digital data in this network.
- A special tool useful for testing EIA/TIA-232 serial data lines is called a *breakout box*. Research what a "breakout box" is and how it may be used.

5.1.4 EIA/TIA-232 data frames of ASCII characters

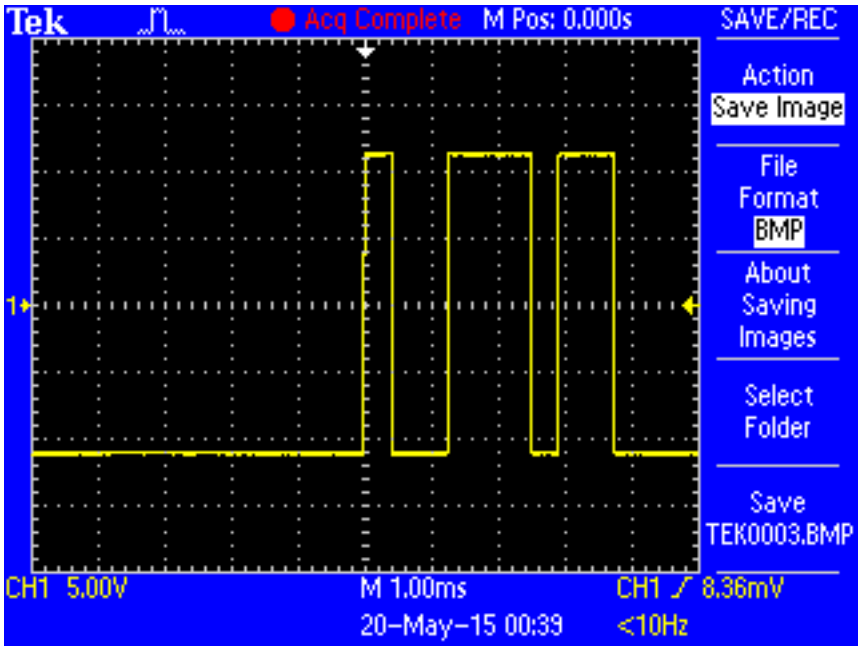
Decode the following EIA/TIA-232 data "frames" captured on a digital oscilloscope, each image showing a single ASCII character. Please note that each of these examples are of serial data communicated at the exact same bit rate, with a successful parity check (i.e. no corrupted bits)[4]:

Example #1 – 8 data bits, odd parity, 1 stop bit:

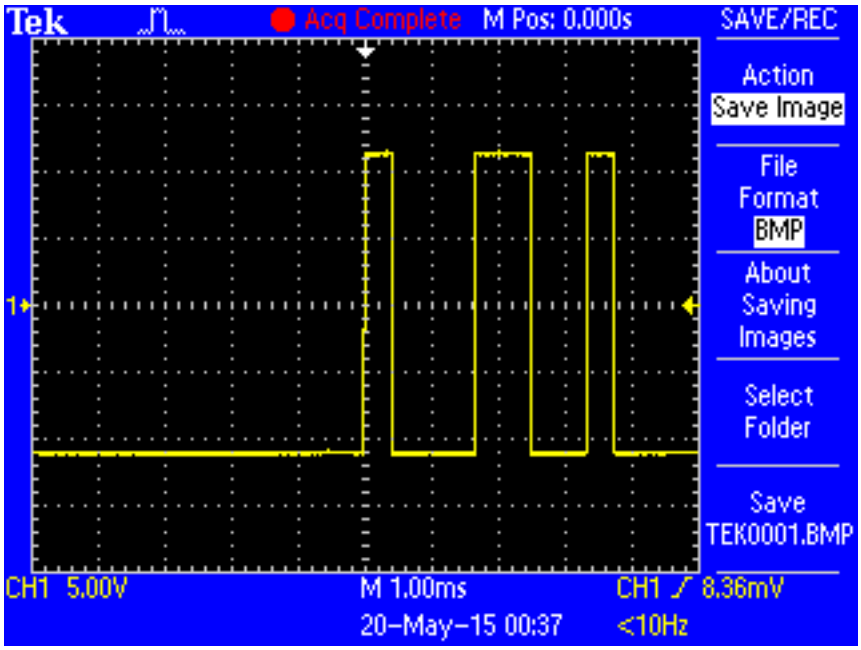


[4] All these oscilloscope screen captures were taken by measuring data signals at the "transmit" pin of a personal computer's EIA/TIA-232 serial port, as single letter keys were pressed while running serial terminal software (e.g. `Kermit`, `Termite`, `Hyperterminal`).

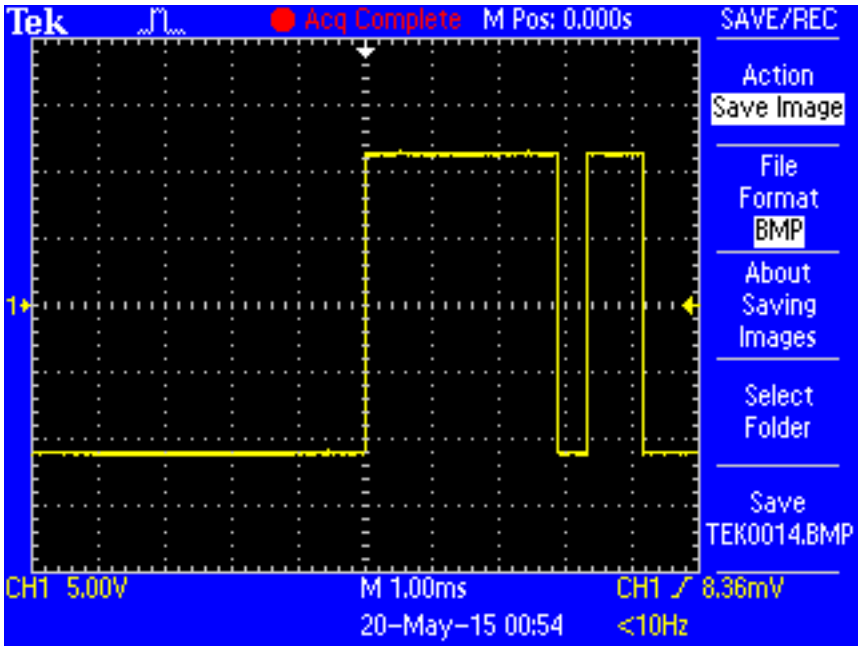
Example #2 – 8 data bits, even parity, 1 stop bit:



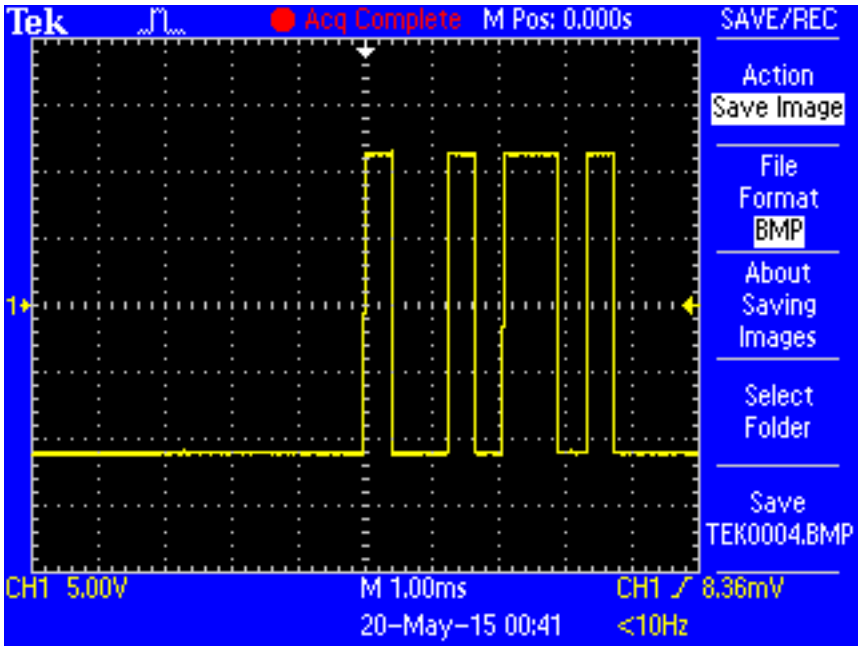
Example #3 – 8 data bits, even parity, 1 stop bit:



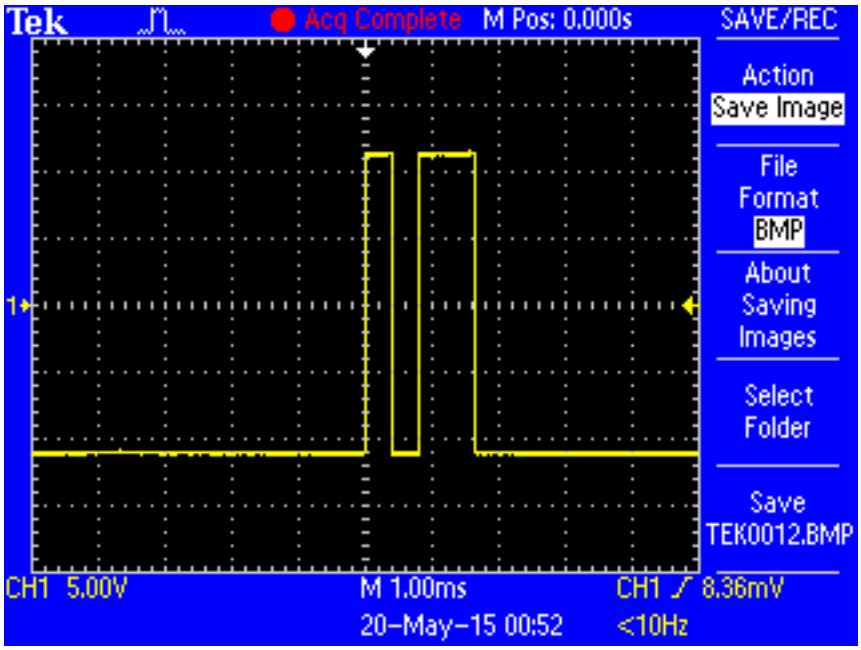
Example #4 – 8 data bits, odd parity, 1 stop bit:



Example #5 – 8 data bits, odd parity, 1 stop bit:



Example #6 – 7 data bits, even parity, 2 stop bits:

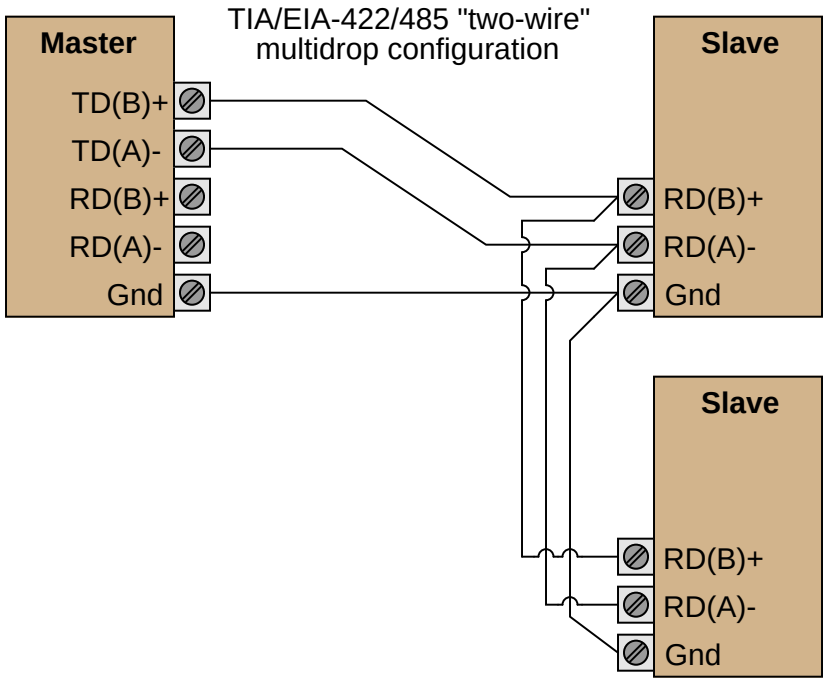


Challenges

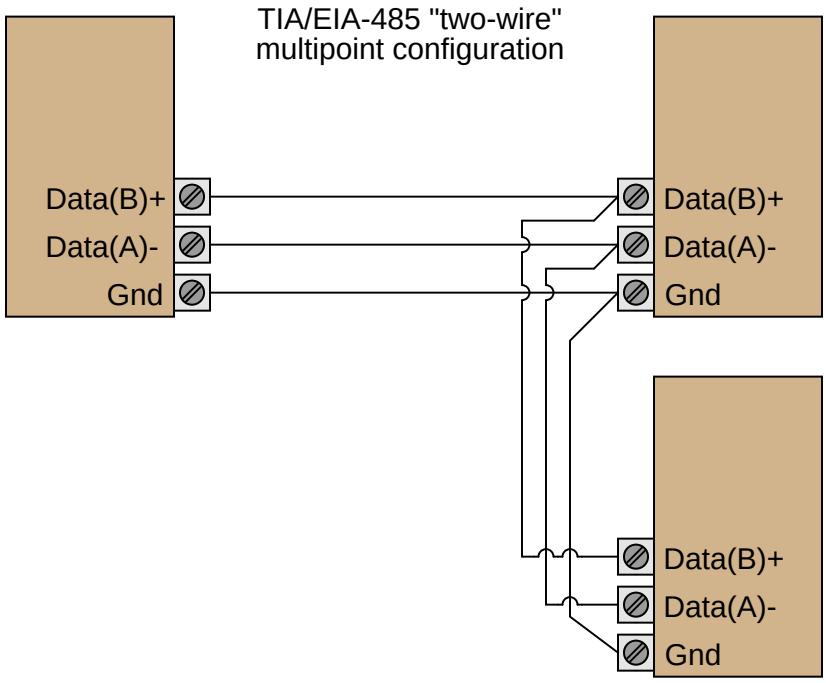
- Identify the bit rate at which all these ASCII characters were transmitted.

5.1.5 Multipoint and multidrop communications

Both EIA/TIA-422 and EIA/TIA-485 networks have the ability to operate in *multidrop mode*, where devices are connected together like this:



However, only EIA/TIA-485 networks are able to operate in *multipoint mode*, where devices are connected together a bit differently:



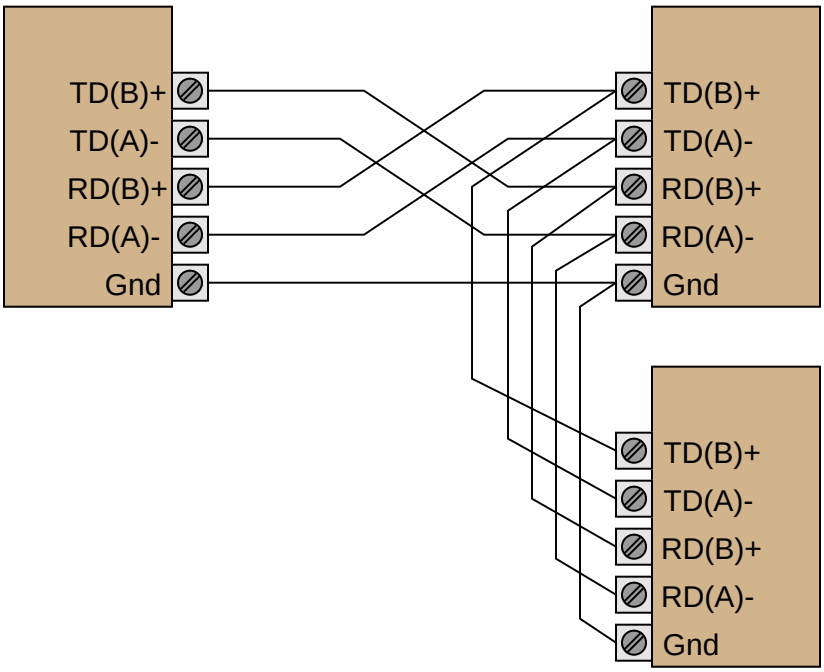
Explain the significance of the "Master/Slave" labels in the multidrop network, and also how the multipoint (EIA/TIA-485 only) network is fundamentally different.

Challenges

- What challenge exists in the device programming for a multipoint network that simply does not exist in a multidrop network?

5.1.6 422 or 485?

Is this network a EIA/TIA-422 or a EIA/TIA-485? Is it possible to tell from this diagram, and if so, how?



Challenges

- Is this system capable of simplex communication, half-duplex communication, and/or full-duplex communication?

5.2 Quantitative reasoning

These questions are designed to stimulate your computational thinking. In a Socratic discussion with your instructor, the goal is for these questions to reveal your mathematical approach(es) to problem-solving so that good technique and sound reasoning may be reinforced. Your instructor may also pose additional questions based on those assigned, in order to observe your problem-solving firsthand.

Mental arithmetic and estimations are strongly encouraged for all calculations, because without these abilities you will be unable to readily detect errors caused by calculator misuse (e.g. keystroke errors).

You will note a conspicuous lack of answers given for these quantitative questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. My advice is to use circuit simulation software such as SPICE to check the correctness of quantitative answers. Refer to those learning modules within this collection focusing on SPICE to see worked examples which you may use directly as practice problems for your own study, and/or as templates you may modify to run your own analyses and generate your own practice problems.

Completely worked example problems found in the Tutorial may also serve as "test cases[5]" for gaining proficiency in the use of circuit simulation software, and then once that proficiency is gained you will never need to rely[6] on an answer key!

[5] In other words, set up the circuit simulation software to analyze the same circuit examples found in the Tutorial. If the simulated results match the answers shown in the Tutorial, it confirms the simulation has properly run. If the simulated results disagree with the Tutorial's answers, something has been set up incorrectly in the simulation software. Using every Tutorial as practice in this way will quickly develop proficiency in the use of circuit simulation software.

[6] This approach is perfectly in keeping with the instructional philosophy of these learning modules: *teaching students to be self-sufficient thinkers*. Answer keys can be useful, but it is even more useful to your long-term success to have a set of tools on hand for checking your own work, because once you have left school and are on your own, there will no longer be "answer keys" available for the problems you will have to solve.

5.2.1 Miscellaneous physical constants

Note: constants shown in **bold** type are *exact*, not approximations. Values inside of parentheses show one standard deviation (σ) of uncertainty in the final digits: for example, the magnetic permeability of free space value given as $1.25663706212(19) \times 10^{-6}$ H/m represents a center value (i.e. the location parameter) of $1.25663706212 \times 10^{-6}$ Henrys per meter with one standard deviation of uncertainty equal to $0.0000000000019 \times 10^{-6}$ Henrys per meter.

Avogadro's number (N_A) = **6.02214076** $\times 10^{23}$ **per mole** (mol^{-1})

Boltzmann's constant (k) = **1.380649** $\times 10^{-23}$ **Joules per Kelvin** (J/K)

Electronic charge (e) = **1.602176634** $\times 10^{-19}$ **Coulomb** (C)

Faraday constant (F) = **96,485.33212...** $\times 10^{4}$ **Coulombs per mole** (C/mol)

Magnetic permeability of free space (μ_0) = $1.25663706212(19) \times 10^{-6}$ Henrys per meter (H/m)

Electric permittivity of free space (ϵ_0) = $8.8541878128(13) \times 10^{-12}$ Farads per meter (F/m)

Characteristic impedance of free space (Z_0) = 376.730313668(57) Ohms (Ω)

Gravitational constant (G) = $6.67430(15) \times 10^{-11}$ cubic meters per kilogram-seconds squared ($m^3/kg\text{-}s^2$)

Molar gas constant (R) = **8.314462618... Joules per mole-Kelvin** (J/mol-K) = 0.08205746(14) liters-atmospheres per mole-Kelvin

Planck constant (h) = **6.62607015** $\times 10^{-34}$ **joule-seconds** (J-s)

Stefan-Boltzmann constant (σ) = **5.670374419...** $\times 10^{-8}$ **Watts per square meter-Kelvin**^{4} ($W/m^2 \cdot K^4$)

Speed of light in a vacuum (c) = **299,792,458 meters per second** (m/s) = 186282.4 miles per second (mi/s)

Note: All constants taken from NIST data "Fundamental Physical Constants – Complete Listing", from `http://physics.nist.gov/constants`, National Institute of Standards and Technology (NIST), 2018 CODATA Adjustment.

5.2.2 Introduction to spreadsheets

A powerful computational tool you are encouraged to use in your work is a *spreadsheet*. Available on most personal computers (e.g. Microsoft Excel), *spreadsheet* software performs numerical calculations based on number values and formulae entered into cells of a grid. This grid is typically arranged as lettered columns and numbered rows, with each cell of the grid identified by its column/row coordinates (e.g. cell `B3`, cell `A8`). Each cell may contain a string of text, a number value, or a mathematical formula. The spreadsheet automatically updates the results of all mathematical formulae whenever the entered number values are changed. This means it is possible to set up a spreadsheet to perform a series of calculations on entered data, and those calculations will be re-done by the computer any time the data points are edited in any way.

For example, the following spreadsheet calculates average speed based on entered values of distance traveled and time elapsed:

	A	B	C	D
1	Distance traveled	46.9	Kilometers	
2	Time elapsed	1.18	Hours	
3	Average speed	= B1 / B2	km/h	
4				
5				

Text labels contained in cells `A1` through `A3` and cells `C1` through `C3` exist solely for readability and are not involved in any calculations. Cell `B1` contains a sample distance value while cell `B2` contains a sample time value. The formula for computing speed is contained in cell `B3`. Note how this formula begins with an "equals" symbol (`=`), references the values for distance and speed by lettered column and numbered row coordinates (`B1` and `B2`), and uses a forward slash symbol for division (`/`). The coordinates `B1` and `B2` function as *variables*[7] would in an algebraic formula.

When this spreadsheet is executed, the numerical value `39.74576` will appear in cell `B3` rather than the formula `= B1 / B2`, because 39.74576 is the computed speed value given 46.9 kilometers traveled over a period of 1.18 hours. If a different numerical value for distance is entered into cell `B1` or a different value for time is entered into cell `B2`, cell `B3`'s value will automatically update. All you need to do is set up the given values and any formulae into the spreadsheet, and the computer will do all the calculations for you.

Cell `B3` may be referenced by other formulae in the spreadsheet if desired, since it is a variable just like the given values contained in `B1` and `B2`. This means it is possible to set up an entire chain of calculations, one dependent on the result of another, in order to arrive at a final value. The arrangement of the given data and formulae need not follow any pattern on the grid, which means you may place them anywhere.

[7] Spreadsheets may also provide means to attach text labels to cells for use as variable names (Microsoft Excel simply calls these labels "names"), but for simple spreadsheets such as those shown here it's usually easier just to use the standard coordinate naming for each cell.

Common[8] arithmetic operations available for your use in a spreadsheet include the following:

- Addition (`+`)
- Subtraction (`-`)
- Multiplication (`*`)
- Division (`/`)
- Powers (`^`)
- Square roots (`sqrt()`)
- Logarithms (`ln()` , `log10()`)

Parentheses may be used to ensure[9] proper order of operations within a complex formula. Consider this example of a spreadsheet implementing the *quadratic formula*, used to solve for roots of a polynomial expression in the form of $ax^2 + bx + c$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

	A	B
1	x_1	= (-B4 + sqrt((B4^2) - (4*B3*B5))) / (2*B3)
2	x_2	= (-B4 - sqrt((B4^2) - (4*B3*B5))) / (2*B3)
3	a =	9
4	b =	5
5	c =	-2

This example is configured to compute roots[10] of the polynomial $9x^2 + 5x - 2$ because the values of 9, 5, and -2 have been inserted into cells `B3`, `B4`, and `B5`, respectively. Once this spreadsheet has been built, though, it may be used to calculate the roots of *any* second-degree polynomial expression simply by entering the new a, b, and c coefficients into cells `B3` through `B5`. The numerical values appearing in cells `B1` and `B2` will be automatically updated by the computer immediately following any changes made to the coefficients.

[8] Modern spreadsheet software offers a bewildering array of mathematical functions you may use in your computations. I recommend you consult the documentation for your particular spreadsheet for information on operations other than those listed here.

[9] Spreadsheet programs, like text-based programming languages, are designed to follow standard order of operations by default. However, my personal preference is to use parentheses even where strictly unnecessary just to make it clear to any other person viewing the formula what the intended order of operations is.

[10] Reviewing some algebra here, a *root* is a value for x that yields an overall value of zero for the polynomial. For this polynomial ($9x^2 + 5x - 2$) the two roots happen to be $x = 0.269381$ and $x = -0.82494$, with these values displayed in cells `B1` and `B2`, respectively upon execution of the spreadsheet.

Alternatively, one could break up the long quadratic formula into smaller pieces like this:

$$y = \sqrt{b^2 - 4ac} \qquad z = 2a$$

$$x = \frac{-b \pm y}{z}$$

	A	B	C
1	x_1	= (-B4 + C1) / C2	= sqrt((B4^2) - (4*B3*B5))
2	x_2	= (-B4 - C1) / C2	= 2*B3
3	a =	9	
4	b =	5	
5	c =	-2	

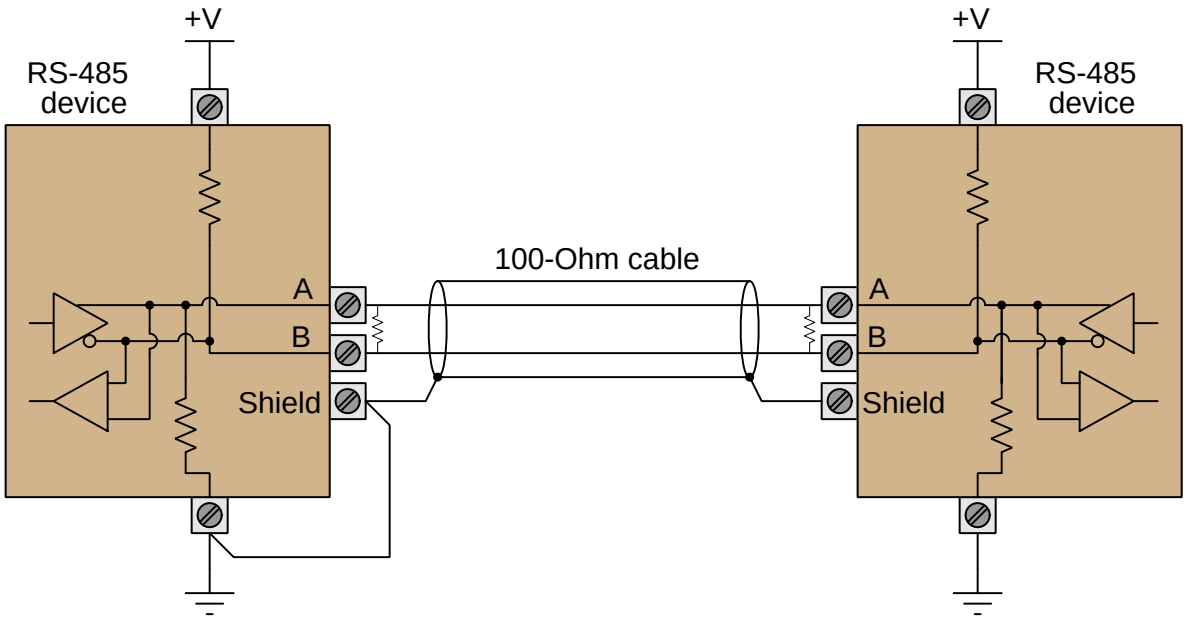
Note how the square-root term (y) is calculated in cell C1, and the denominator term (z) in cell C2. This makes the two final formulae (in cells B1 and B2) simpler to interpret. The positioning of all these cells on the grid is completely arbitrary[11] – all that matters is that they properly reference each other in the formulae.

Spreadsheets are particularly useful for situations where the same set of calculations representing a circuit or other system must be repeated for different initial conditions. The power of a spreadsheet is that it automates what would otherwise be a tedious set of calculations. One specific application of this is to simulate the effects of various components within a circuit failing with abnormal values (e.g. a shorted resistor simulated by making its value nearly zero; an open resistor simulated by making its value extremely large). Another application is analyzing the behavior of a circuit design given new components that are out of specification, and/or aging components experiencing drift over time.

[11] My personal preference is to locate all the "given" data in the upper-left cells of the spreadsheet grid (each data point flanked by a sensible name in the cell to the left and units of measurement in the cell to the right as illustrated in the first distance/time spreadsheet example), sometimes coloring them in order to clearly distinguish which cells contain entered data versus which cells contain computed results from formulae. I like to place all formulae in cells below the given data, and try to arrange them in logical order so that anyone examining my spreadsheet will be able to figure out *how* I constructed a solution. This is a general principle I believe all computer programmers should follow: *document and arrange your code to make it easy for other people to learn from it.*

5.2.3 Terminating and bias resistors

Terminating resistors are not always necessary in EIA/TIA-485 networks, but when they are it is important to ensure their presence does not compromise biasing. Explain how termination resistors may adversely affect the biasing of a EIA/TIA-485 network, based on what you see in this schematic:



Calculate the "idle" voltage for this data network, assuming termination resistors of 100 Ω each, bias resistors of 1 kΩ each, and a 15 Volt power supply at each end. Does this meet the standard for a EIA/TIA-485 network?

Challenges

- Are terminating resistors always needed in an EIA/TIA-485 network? If not, what applications can do without them?
- Calculate new bias resistor values to meet the minimum transmitter voltage levels for the EIA/TIA-485 standard.
- Demonstrate how to *estimate* numerical answers for this problem without using a calculator.
- If the cable used in this system is replaced by one having significantly greater length but possessing the same characteristic impedance rating as before, will the terminating resistor values need to be altered? Why or why not? If the resistor values do need to be changed, will they need to be larger (more Ohms) or smaller (less Ohms) than they are now?

5.2.4 EIA/TIA-232 data transmission time

How long does it take to transmit a single ASCII character via an EIA/TIA-232 network operating at a bit rate of 19200 bits per second, with an 8-bit payload, odd parity, and one stop bit?

How long does it take to transmit a single ASCII character via an EIA/TIA-232 network operating at a bit rate of 2400 bits per second, with a 7-bit payload, even parity, and one stop bit?

Do parity bits affect transmission time? Why or why not?

Challenges

- Why not just use the absolute maximum bit-rate offered by any EIA/TIA-232 devices for all communication needs?

5.3 Diagnostic reasoning

These questions are designed to stimulate your deductive and inductive thinking, where you must apply general principles to specific scenarios (deductive) and also derive conclusions about the failed circuit from specific details (inductive). In a Socratic discussion with your instructor, the goal is for these questions to reinforce your recall and use of general circuit principles and also challenge your ability to integrate multiple symptoms into a sensible explanation of what's wrong in a circuit. Your instructor may also pose additional questions based on those assigned, in order to further challenge and sharpen your diagnostic abilities.

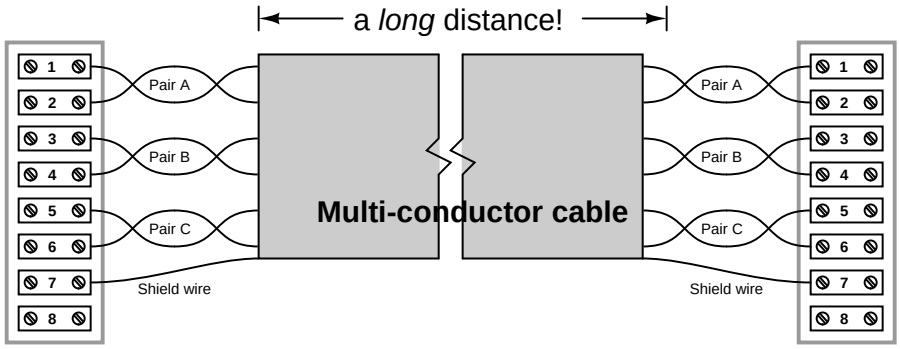
As always, your goal is to fully *explain* your analysis of each problem. Simply obtaining a correct answer is not good enough – you must also demonstrate sound reasoning in order to successfully complete the assignment. Your instructor's responsibility is to probe and challenge your understanding of the relevant principles and analytical processes in order to ensure you have a strong foundation upon which to build further understanding.

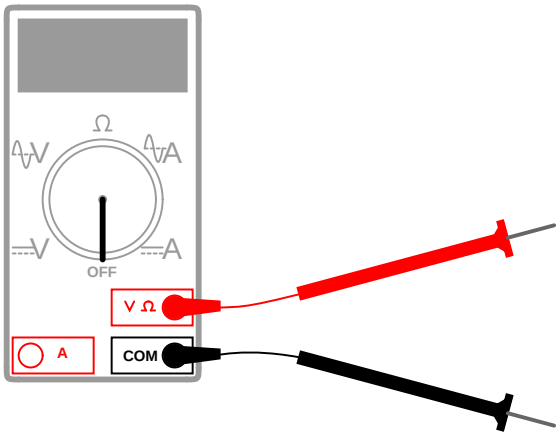
You will note a conspicuous lack of answers given for these diagnostic questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your diagnostic answers, where applicable, is to use circuit simulation software to explore the effects of faults placed in circuits. For example, if one of these diagnostic questions requires that you predict the effect of an open or a short in a circuit, you may check the validity of your work by simulating that same fault (substituting a very high resistance in place of that component for an open, and substituting a very low resistance for a short) within software and seeing if the results agree.

5.3.1 Testing a multi-pair data cable

Suppose you are asked to check the integrity of a multi-conductor signal cable run between two locations. The cable has six signal conductors in it plus a shield, each one terminated at a terminal block at each end:





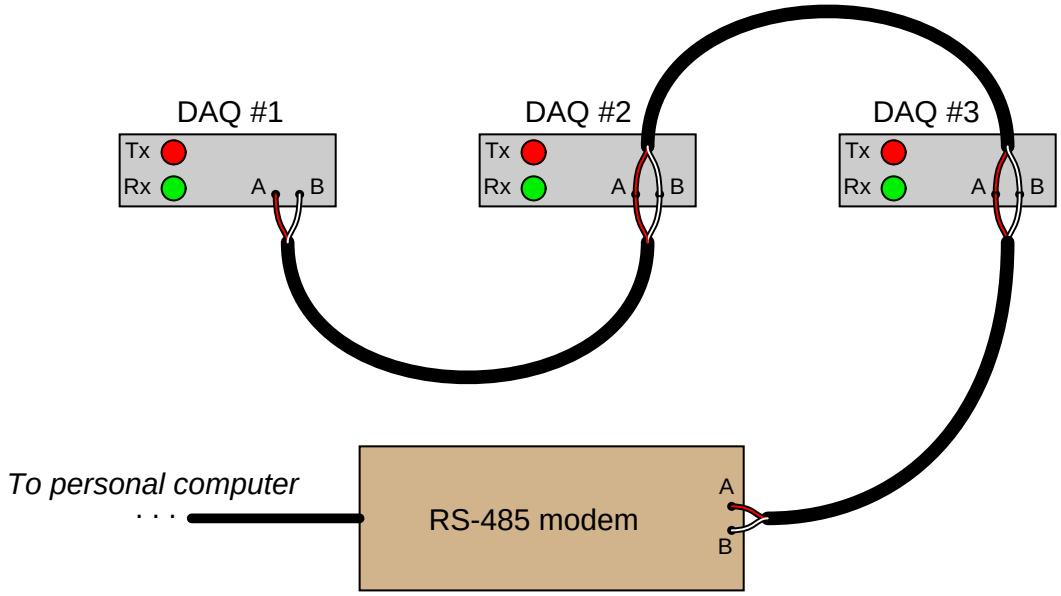
Faults you are looking for include *open* conductors, as well as *shorts* between conductors and/or shorts to ground. Devise a series of tests you could perform with nothing but a multimeter to comprehensively check the electrical integrity of this cable.

Challenges

- Suppose you did not have a multimeter available for this test, but only possessed a 6 Volt battery and 6 Volt incandescent lamp. Explain how you could use these components to test the cable just as adequately as with a multimeter.

5.3.2 Networked DAQ modules

A data acquisition system consisting of three DAQ[12] modules connected to a personal computer using RS-485 (multipoint) through a modem has a problem. The DAQ software running in the personal computer does not seem to "recognize" DAQ #1, although it does recognize the other two DAQ modules and is receiving data from them:



Looking at the LED indicators, you notice that the green "Rx" LEDs on all three DAQ units blink. You also notice that the red "Tx" LEDs on DAQ modules #2 and #3 are blinking as well, but that the "Tx" LED on DAQ #1 is constantly off.

Identify the likelihood of each specified fault for this system. Consider each fault one at a time (i.e. no coincidental faults), determining whether or not each fault is compatible with *all* measurements and symptoms in this network.

- Open cable between DAQ #1 and DAQ #2 =
- Open cable between DAQ #2 and DAQ #3 =
- Open cable between DAQ #3 and modem =
- Incorrect serial settings (e.g. parity, bit rate) in DAQ #1 =
- Incorrect serial settings (e.g. parity, bit rate) in DAQ #2 =
- Incorrect serial settings (e.g. parity, bit rate) in DAQ #3 =

[12] A "DAQ" module is a *Data AcQuisition* circuit that usually takes in analog or digital data from sensors and converts it into digital data readable to a computer. This functionality is not important to the question of networking.

- Incorrect serial settings (e.g. parity, bit rate) in computer =
- Missing termination resistor =

Challenges

- Is this network capable of simplex, half-duplex, or full-duplex operation?
- Identify another possible fault not present in this list.
- Identify how a digital multimeter (DMM) could be used to perform useful diagnostic tests in an RS-485 network such as this. What sorts of things can we measure using a DMM connected to a digital network? What sorts of things *can't* we measure using a DMM connected to a digital network?

Appendix A

Problem-Solving Strategies

The ability to solve complex problems is arguably one of the most valuable skills one can possess, and this skill is particularly important in any science-based discipline.

- Study principles, not procedures. Don't be satisfied with merely knowing how to compute solutions – learn *why* those solutions work.
- Identify what it is you need to solve, identify all relevant data, identify all units of measurement, identify any general principles or formulae linking the given information to the solution, and then identify any "missing pieces" to a solution. Annotate all diagrams with this data.
- Sketch a diagram to help visualize the problem. When building a real system, always devise a plan for that system and analyze its function *before* constructing it.
- Follow the units of measurement and meaning of every calculation. If you are ever performing mathematical calculations as part of a problem-solving procedure, and you find yourself unable to apply each and every intermediate result to some aspect of the problem, it means you don't understand what you are doing. Properly done, every mathematical result should have practical meaning for the problem, and not just be an abstract number. You should be able to identify the proper units of measurement for each and every calculated result, and show where that result fits into the problem.
- Perform "thought experiments" to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults, the best diagnostic test being the one giving you the most information about the nature and/or location of the fault with the fewest steps.
- Simplify the problem until the solution becomes obvious, and then use that obvious case as a model to follow in solving the more complex version of the problem.
- Check for exceptions to see if your solution is incorrect or incomplete. A good solution will work for *all* known conditions and criteria. A good example of this is the process of testing scientific hypotheses: the task of a scientist is not to find support for a new idea, but rather to *challenge* that new idea to see if it holds up under a battery of tests. The philosophical

principle of *reductio ad absurdum* (i.e. disproving a general idea by finding a specific case where it fails) is useful here.

- Work "backward" from a hypothetical solution to a new set of given conditions.
- Add quantities to problems that are qualitative in nature, because sometimes a little math helps illuminate the scenario.
- Sketch graphs illustrating how variables relate to each other. These may be quantitative (i.e. with realistic number values) or qualitative (i.e. simply showing increases and decreases).
- Treat quantitative problems as qualitative in order to discern the relative magnitudes and/or directions of change of the relevant variables. For example, try determining what happens if a certain variable were to increase or decrease before attempting to precisely calculate quantities: how will each of the dependent variables respond, by increasing, decreasing, or remaining the same as before?
- Consider limiting cases. This works especially well for qualitative problems where you need to determine which direction a variable will change. Take the given condition and magnify that condition to an extreme degree as a way of simplifying the direction of the system's response.
- Check your work. This means regularly testing your conclusions to see if they make sense. This does *not* mean repeating the same steps originally used to obtain the conclusion(s), but rather to use some other means to check validity. Simply repeating procedures often leads to *repeating the same errors* if any were made, which is why alternative paths are better.

Appendix B

Instructional philosophy

B.1 First principles of learning

- **Anyone can learn anything** given appropriate time, effort, resources, challenges, encouragement, and expectations. Dedicating time and investing effort are the student's responsibility; providing resources, challenges, and encouragement are the teacher's responsibility; high expectations are a responsibility shared by both student and teacher.

- **Transfer is not automatic.** The human mind has a natural tendency to compartmentalize information, which means the process of taking knowledge learned in one context and applying it to another usually does not come easy and therefore should never be taken for granted.

- **Learning is iterative.** The human mind rarely learns anything perfectly on the first attempt. Anticipate mistakes and plan for multiple tries to achieve full understanding, using the lessons of those mistakes as feedback to guide future attempts.

- **Information is absorbed, but understanding is created.** Facts and procedures may be memorized easily enough by repeated exposure, but the ability to reliably apply principles to novel scenarios only comes through intense personal effort. This effort is fundamentally creative in nature: casting new concepts in one's own words, running mental experiments to perceive patterns and trends, etc. These acts of making knowledge "one's own" need not be perfect in order to be effective, as the value lies in the activity and not necessarily the finished product. Depth of understanding and permanence of memory are both directly related to mental effort during the learning process – i.e. *easy learning is shallow and fleeting.*

- **Education trumps training.** There is no such thing as an entirely isolated subject, as all fields of knowledge are connected. Training is narrowly-focused and task-oriented. Education is broad-based and principle-oriented. When preparing for a life-long technical career, education beats training every time.

- **Character matters.** Poor habits are more destructive than deficits of knowledge or skill. This is especially true in collective endeavors, where a team's ability to function depends on trust between its members. Simply put, no one wants an untrustworthy person on their team. An essential component of education then, is character development.

- **People learn to be responsible by bearing responsibility.** An irresponsible person is someone who has never *had* to be responsible for anything that mattered enough to them. Just as anyone can learn anything, anyone can become responsible if the personal cost of irresponsibility becomes high enough.

- **What gets measured, gets done.** Accurate and relevant assessment of learning is key to ensuring all students learn. Therefore, it is imperative to measure what matters.

- **Failure is nothing to fear.** Every human being fails, and fails in multiple ways at multiple times. Eventual success only happens when we don't stop trying.

B.2 Proven strategies for instructors

- Assume every student is capable of learning anything they desire given the proper conditions. Treat them as capable adults by granting real responsibility and avoiding artificial incentives such as merit or demerit points.
- Create a consistent culture of high expectations across the entire program of study. Demonstrate and encourage patience, persistence, and a healthy sense of self-skepticism. Anticipate and de-stigmatize error. Teach respect for the capabilities of others as well as respect for one's own fallibility.
- Replace lecture with "inverted" instruction, where students first encounter new concepts through reading and then spend class time in Socratic dialogue with the instructor exploring those concepts and solving problems individually. There is a world of difference between observing someone solve a problem versus actually solving a problem yourself, and so the point of this form of instruction is to place students in a position where they *cannot* passively observe.
- Require students to read extensively, write about what they learn, and dialogue with you and their peers to sharpen their understanding. Apply Francis Bacon's advice that "reading maketh a full man; conference a ready man; and writing an exact man". These are complementary activities helping students expand their confidence and abilities.
- Use artificial intelligence (AI) to challenge student understanding rather than merely provide information. Find productive ways for AI to critique students' clarity of thought and of expression, for example by employing AI as a Socratic-style interlocutor or as a reviewer of students' journals. Properly applied, AI has the ability to expand student access to critical review well outside the bounds of their instructor's reach.
- Build frequent and rapid feedback into the learning process so that students know at all times how well they are learning, to identify problems early and fix them before they grow. Model the intellectual habit of self-assessing and self-correcting your own understanding (i.e. a cognitive *feedback loop*), encouraging students to do the same.
- Use "mastery" as the standard for every assessment, which means the exam or experiment or project must be done with 100% competence in order to pass. Provide students with multiple opportunity for re-tries (different versions of the assessment every time).
- Require students to devise their own hypotheses and procedures on all experiments, so that the process is truly a scientific one. Have students assess their proposed experimental procedures for risk and devise mitigations for those risks. Let nothing be pre-designed about students' experiments other than a stated task (i.e. what principle the experiment shall test) at the start and a set of demonstrable knowledge and skill objectives at the end.
- Have students build as much of their lab equipment as possible: building power sources, building test assemblies[1], and building complete working systems (no kits!). In order to provide

[1] In the program I teach, every student builds their own "Development Board" consisting of a metal chassis with DIN rail, terminal blocks, and an AC-DC power supply of their own making which functions as a portable lab environment they can use at school as well as take home.

this same "ground-up" experience for every new student, this means either previous students take their creations with them, or the systems get disassembled in preparation for the new students, or the systems grow and evolve with each new student group.

- Incorporate external accountability for you and for your students, continuously improving the curriculum and your instructional methods based on proven results. Have students regularly network with active professionals through participation in advisory committee meetings, service projects, tours, jobshadows, internships, etc. Practical suggestions include requiring students to design and build projects for external clients (e.g. community groups, businesses, different departments within the institution), and also requiring students attend all technical advisory committee meetings and to dialogue with the industry representatives at those meetings.
- Repeatedly explore difficult-to-learn concepts across multiple courses, so that students have multiple opportunities to build their understanding.
- Relate all new concepts, whenever possible, to previous concepts and to relevant physical laws. Challenge each and every student, every day, to *reason* from concept to concept and to explain the logical connections between. Challenge students to verify their conclusions by multiple approaches (e.g. double-checking their work using different methods). Ask *"Why?"* often.
- Maintain detailed records on each student's performance and share these records privately with them. These records should include academic performance as well as professionally relevant behavioral tendencies.
- Hold mandatory "check-in" meetings between all program faculty and each new student during their first term. Offer these to all other students as an option, except for any students continuing to manifest unprofessional behaviors, poor academic performance, or who have some other need for a face-to-face meeting with faculty.
- Address problems while they are small, before they grow larger. This is equally true for tutoring technical concepts as it is for helping students build professional habits.
- Build rigorous quality control into the curriculum to ensure every student masters every important concept, and that the mastery is retained over time. This includes (1) review questions added to every exam to re-assess knowledge taught in previous terms, (2) cumulative exams at the end of every term to re-assess all important concepts back to the very beginning of the program, and (3) review assessments in practical (hands-on) coursework to ensure critically-important skills were indeed taught and are still retained. What you will find by doing this is that it actually boosts retention of students by ensuring that important knowledge gets taught and is retained over long spans of time. In the absence of such quality control, student learning and retention tends to be spotty and this contributes to drop-out and failure rates later in their education.
- Finally, *never rush learning.* Education is not a race. Give your students ample time to digest complex ideas, as you continually remind yourself of just how long it took you to achieve mastery! Long-term retention and the consistently correct application of concepts are always the result of *focused effort over long periods of time* which means there are no shortcuts to learning.

B.3 Proven strategies for students

The single most important piece of advice I have for any student of any subject is to take responsibility for your own development in all areas of life including mental development. Expecting others in your life to entirely guide your own development is a recipe for disappointment. This is just as true for students enrolled in formal learning institutions as it is for auto-didacts pursuing learning entirely on their own. Learning to think in new ways is key to being able to gainfully use information, to make informed decisions about your life, and to best serve those you care about. With this in mind, I offer the following advice to students:

- **Approach all learning as valuable**. No matter what course you take, no matter who you learn from, no matter the subject, there is something useful in every learning experience. If you don't see the value of every new experience, you are not looking closely enough!
- **Continually challenge yourself**. Let other people take shortcuts and find easy answers to easy problems. The purpose of education is to stretch your mind, in order to shape it into a more powerful tool. This doesn't come by taking the path of least resistance. An excellent analogy for an empowering education is productive physical exercise: becoming stronger, more flexible, and more persistent only comes through intense personal effort.
- **Master the use of language**. This includes reading extensively, writing every day, listening closely, and speaking articulately. To a great extent language channels and empowers thought, so the better you are at wielding language the better you will be at grasping abstract concepts and articulating them not only for your benefit but for others as well.
- **Do not limit yourself to the resources given to you**. Read books that are not on the reading list. Run experiments that aren't assigned to you. Form study groups outside of class. Take an entrepreneurial approach to your own education, as though it were a business you were building for your future benefit.
- **Express and share what you learn**. Take every opportunity to teach what you have learned to others, as this will not only help them but will also strengthen your own understanding[2].
- Realize that **no one can give you understanding**, just as no one can give you physical fitness. These both must be *built*.
- **Above all, recognize that learning is hard work, and that a certain level of frustration is unavoidable**. There are times when you will struggle to grasp some of these concepts, and that struggle is a natural thing. Take heart that it will yield with persistent and varied[3] effort, and never give up! That concepts don't immediately come to you is not a sign of something wrong, but rather of something right: that you have found a worthy challenge!

[2] On a personal note, I was surprised to learn just how much my own understanding of electronics and related subjects was strengthened by becoming a teacher. When you are tasked every day with helping other people grasp complex topics, it catalyzes your own learning by giving you powerful incentives to study, to articulate your thoughts, and to reflect deeply on the process of learning.

[3] As the old saying goes, "Insanity is trying the same thing over and over again, expecting different results." If you find yourself stumped by something in the text, you should attempt a different approach. Alter the thought experiment, change the mathematical parameters, do whatever you can to see the problem in a slightly different light, and then the solution will often present itself more readily.

B.4 Design of these learning modules

"The unexamined circuit is not worth energizing" – Socrates (if he had taught electricity)

These learning modules, although useful for self-study, were designed to be used in a formal learning environment where a subject-matter expert challenges students to digest the content and exercise their critical thinking abilities in the answering of questions and in the construction and testing of working circuits. Every effort has been made to embed the following instructional and assessment philosophies within:

- The first goal of education is to enhance clear and independent thought, in order that every student reach their fullest potential in a highly complex and inter-dependent world. Robust reasoning is *always* more important than particulars of any subject matter, because its application is universal.
- Literacy is fundamental to independent learning and thought because text continues to be the most efficient way to communicate complex ideas over space and time. Those who cannot read with ease are limited in their ability to acquire knowledge and perspective.
- Articulate communication is fundamental to work that is complex and interdisciplinary.
- Faulty assumptions and poor reasoning are best corrected through challenge, not presentation. The rhetorical technique of *reductio ad absurdum* (disproving an assertion by exposing an absurdity) works well to discipline student's minds, not only to correct the problem at hand but also to learn how to detect and correct future errors.
- Important principles should be repeatedly explored and widely applied throughout a course of study, not only to reinforce their importance and help ensure their mastery, but also to showcase the interconnectedness and utility of knowledge.

These learning modules were expressly designed to be used in an "inverted" teaching environment[4] where students first read the introductory and tutorial chapters on their own, then individually attempt to answer the questions and construct working circuits according to the experiment and project guidelines. The instructor never lectures, but instead meets regularly with each individual student to review their progress, answer questions, identify misconceptions, and challenge the student to new depths of understanding through further questioning. Regular meetings between instructor and student should resemble a Socratic[5] dialogue, where questions serve as scalpels to dissect topics and expose assumptions. The student passes each module only after consistently demonstrating their ability to logically analyze and correctly apply all major concepts in each question or project/experiment. The instructor must be vigilant in probing each student's understanding to ensure they are truly *reasoning* and not just *memorizing*. This is why "Challenge" points appear throughout, as prompts for students to think deeper about topics and as starting points for instructor queries. Sometimes these challenge points require additional knowledge that hasn't been covered in the series to answer in full. This is okay, as the major purpose of the Challenges is to stimulate analysis and synthesis on the part of each student.

The instructor must possess enough mastery of the subject matter and awareness of students' reasoning to generate their own follow-up questions to practically any student response. Even completely correct answers given by the student should be challenged by the instructor for the purpose of having students practice articulating their thoughts and defending their reasoning. Conceptual errors committed by the student should be exposed and corrected not by direct instruction, but rather by reducing the errors to an absurdity[6] through well-chosen questions and thought experiments posed by the instructor. Becoming proficient at this style of instruction requires time and dedication, but the positive effects on critical thinking for both student and instructor are spectacular.

An inspection of these learning modules reveals certain unique characteristics. One of these is a bias toward thorough explanations in the tutorial chapters. Without a live instructor to explain concepts and applications to students, the text itself must fulfill this role. This philosophy results in lengthier explanations than what you might typically find in a textbook, each step of the reasoning process fully explained, including footnotes addressing common questions and concerns students raise while learning these concepts. Each tutorial seeks to not only explain each major concept in sufficient detail, but also to explain the logic of each concept and how each may be developed

[4] In a traditional teaching environment, students first encounter new information via *lecture* from an expert, and then independently apply that information via *homework*. In an "inverted" course of study, students first encounter new information via *homework*, and then independently apply that information under the scrutiny of an expert. The expert's role in lecture is to simply *explain*, but the expert's role in an inverted session is to *challenge*, *critique*, and if necessary *explain* where gaps in understanding still exist.

[5] Socrates is a figure in ancient Greek philosophy famous for his unflinching style of questioning. Although he authored no texts, he appears as a character in Plato's many writings. The essence of Socratic philosophy is to leave no question unexamined and no point of view unchallenged. While purists may argue a topic such as electric circuits is too narrow for a true Socratic-style dialogue, I would argue that the essential thought processes involved with scientific reasoning on *any* topic are not far removed from the Socratic ideal, and that students of electricity and electronics would do very well to challenge assumptions, pose thought experiments, identify fallacies, and otherwise employ the arsenal of critical thinking skills modeled by Socrates.

[6] This rhetorical technique is known by the Latin phrase *reductio ad absurdum*. The concept is to expose errors by counter-example, since only one solid counter-example is necessary to disprove a universal claim. As an example of this, consider the common misconception among beginning students of electricity that voltage cannot exist without current. One way to apply *reductio ad absurdum* to this statement is to ask how much current passes through a fully-charged battery connected to nothing (i.e. a clear example of voltage existing without current).

from "first principles". Again, this reflects the goal of developing clear and independent thought in students' minds, by showing how clear and logical thought was used to forge each concept. Students benefit from witnessing a model of clear thinking in action, and these tutorials strive to be just that.

Another feature of these learning modules is that they do not treat topics in isolation. Rather, important concepts are introduced early in the series, and appear repeatedly as stepping-stones toward other concepts in subsequent modules. This helps to avoid the "compartmentalization" of knowledge, demonstrating the inter-connectedness of concepts and simultaneously reinforcing them. Each module is fairly complete in itself, reserving the beginning of its tutorial to a review of foundational concepts.

To high standards of education,

Tony R. Kuphaldt

Appendix C

Tools used

I am indebted to the developers of many open-source software applications in the creation of these learning modules. The following is a list of these applications with some commentary on each.

You will notice a theme common to many of these applications: a bias toward *code*. Although I am by no means an expert programmer in any computer language, I understand and appreciate the flexibility offered by code-based applications where the user (you) enters commands into a plain ASCII text file, which the software then reads and processes to create the final output. Code-based computer applications are by their very nature *extensible*, while WYSIWYG (What You See Is What You Get) applications are generally limited to whatever user interface the developer makes for you.

The `GNU/Linux` computer operating system

> There is so much to be said about Linus Torvalds' `Linux` and Richard Stallman's `GNU` project. First, to credit just these two individuals is to fail to do justice to the *mob* of passionate volunteers who contributed to make this amazing software a reality. I first learned of `Linux` back in 1996, and have been using this operating system on my personal computers almost exclusively since then. It is *free*, it is completely *configurable*, and it permits the continued use of highly efficient `Unix` applications and scripting languages (e.g. shell scripts, Makefiles, `sed`, `awk`) developed over many decades. `Linux` not only provided me with a powerful computing platform, but its open design served to inspire my life's work of creating open-source educational resources.

Bram Moolenaar's `Vim` text editor

> Writing code for any code-based computer application requires a *text editor*, which may be thought of as a word processor strictly limited to outputting plain-ASCII text files. Many good text editors exist, and one's choice of text editor seems to be a deeply personal matter within the programming world. I prefer `Vim` because it operates very similarly to `vi` which is ubiquitous on `Unix`/`Linux` operating systems, and because it may be entirely operated via keyboard (i.e. no mouse required) which makes it fast to use.

Donald Knuth's TEX typesetting system

Developed in the late 1970's and early 1980's by computer scientist extraordinaire Donald Knuth to typeset his multi-volume magnum opus *The Art of Computer Programming*, this software allows the production of formatted text for screen-viewing or paper printing, all by writing plain-text code to describe how the formatted text is supposed to appear. TEX is not just a markup language for documents, but it is also a Turing-complete programming language in and of itself, allowing useful algorithms to be created to control the production of documents. Simply put, TEX *is a programmer's approach to word processing*. Since TEX is controlled by code written in a plain-text file, this means anyone may read that plain-text file to see exactly how the document was created. This openness afforded by the code-based nature of TEX makes it relatively easy to learn how other people have created their own TEX documents. By contrast, examining a beautiful document created in a conventional WYSIWYG word processor such as Microsoft `Word` suggests nothing to the reader about *how* that document was created, or what the user might do to create something similar. As Mr. Knuth himself once quipped, conventional word processing applications should be called WYSIAYG (What You See Is *All* You Get).

Leslie Lamport's LATEX extensions to TEX

Like all true programming languages, TEX is inherently extensible. So, years after the release of TEX to the public, Leslie Lamport decided to create a massive extension allowing easier compilation of book-length documents. The result was LATEX, which is the markup language used to create all ModEL module documents. You could say that TEX is to LATEX as `C` is to `C++`. This means it is permissible to use any and all TEX commands within LATEX source code, and it all still works. Some of the features offered by LATEX that would be challenging to implement in TEX include automatic index and table-of-content creation.

Tim Edwards' `Xcircuit` drafting program

This wonderful program is what I use to create all the schematic diagrams and illustrations (but not photographic images or mathematical plots) throughout the ModEL project. It natively outputs PostScript format which is a true vector graphic format (this is why the images do not pixellate when you zoom in for a closer view), and it is so simple to use that I have never had to read the manual! Object libraries are easy to create for `Xcircuit`, being plain-text files using PostScript programming conventions. Over the years I have collected a large set of object libraries useful for drawing electrical and electronic schematics, pictorial diagrams, and other technical illustrations.

`Gimp` graphic image manipulation program

Essentially an open-source clone of Adobe's `PhotoShop`, I use `Gimp` to resize, crop, and convert file formats for all of the photographic images appearing in the ModEL modules. Although `Gimp` does offer its own scripting language (called `Script-Fu`), I have never had occasion to use it. Thus, my utilization of `Gimp` to merely crop, resize, and convert graphic images is akin to using a sword to slice bread.

`SPICE` circuit simulation program

`SPICE` is to circuit analysis as TEX is to document creation: it is a form of markup language designed to describe a certain object to be processed in plain-ASCII text. When the plain-text "source file" is compiled by the software, it outputs the final result. More modern circuit analysis tools certainly exist, but I prefer `SPICE` for the following reasons: it is *free*, it is *fast*, it is *reliable*, and it is a fantastic tool for *teaching* students of electricity and electronics how to write simple code. I happen to use rather old versions of `SPICE`, version 2g6 being my "go to" application when I only require text-based output. `NGSPICE` (version 26), which is based on Berkeley `SPICE` version 3f5, is used when I require graphical output for such things as time-domain waveforms and Bode plots. In all `SPICE` example netlists I strive to use coding conventions compatible with all `SPICE` versions.

Andrew D. Hwang's `ePiX` mathematical visualization programming library

This amazing project is a `C++` library you may link to any `C/C++` code for the purpose of generating PostScript graphic images of mathematical functions. As a completely free and open-source project, it does all the plotting I would otherwise use a Computer Algebra System (CAS) such as `Mathematica` or `Maple` to do. It should be said that `ePiX` is *not* a Computer Algebra System like `Mathematica` or `Maple`, but merely a mathematical *visualization* tool. In other words, it won't determine integrals for you (you'll have to implement that in your own `C/C++` code!), but it can graph the results, and it does so beautifully. What I really admire about `ePiX` is that it is a `C++` programming library, which means it builds on the existing power and toolset available with that programming language. Mr. Hwang could have probably developed his own stand-alone application for mathematical plotting, but by creating a `C++` library to do the same thing he accomplished something much greater.

`gnuplot` mathematical visualization software

Another open-source tool for mathematical visualization is `gnuplot`. Interestingly, this tool is *not* part of Richard Stallman's GNU project, its name being a coincidence. For this reason the authors prefer "gnu" *not* be capitalized at all to avoid confusion. This is a much "lighter-weight" alternative to a spreadsheet for plotting tabular data, and the fact that it easily outputs directly to an X11 console or a file in a number of different graphical formats (including PostScript) is very helpful. I typically set my `gnuplot` output format to default (X11 on my Linux PC) for quick viewing while I'm developing a visualization, then switch to PostScript file export once the visual is ready to include in the document(s) I'm writing. As with my use of `Gimp` to do rudimentary image editing, my use of `gnuplot` only scratches the surface of its capabilities, but the important points are that it's *free* and that it *works well*.

`Python` programming language

Both Python and C++ find extensive use in these modules as instructional aids and exercises, but I'm listing Python here as a *tool* for myself because I use it almost daily as a *calculator*. If you open a Python interpreter console and type `from math import *` you can type mathematical expressions and have it return results just as you would on a hand calculator. Complex-number (i.e. *phasor*) arithmetic is similarly supported if you include the complex-math library (`from cmath import *`). Examples of this are shown in the Programming References chapter (if included) in each module. Of course, being a fully-featured programming language, Python also supports conditionals, loops, and other structures useful for calculation of quantities. Also, running in a console environment where all entries and returned values show as text in a chronologically-ordered list makes it easy to copy-and-paste those calculations to document exactly how they were performed.

Appendix D

Creative Commons License

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or

limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures.

For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors,

whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully

be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at `creativecommons.org/policies`, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at `creativecommons.org`.

Appendix E

References

"422 and 485 Standards Overview and System Configurations" Application Report SLLA070C, Texas Instruments Incorporated, Dallas, TX, 2002.

"B&B Converters for the Industrial Bus World" Technical Article 13, B&B Electronics Manufacturing Company, Ottawa, IL, 2000.

Floyd, Thomas L., *Digital Fundamentals*, 6th edition, Prentice-Hall, Inc., Upper Saddle River, NJ, 1997.

"Fundamentals of RS-232 Serial Communications" Application Note 83 (AN83), Maxim Integrated Products, 2001.

Horak, Ray, *Telecommunications and Data Communications Handbook*, John Wiley & Sons, Inc., New York, NY, 2007.

Horak, Ray, *Webster's New World Telecom Dictionary*, Wiley Publishing, Inc., Indianapolis, IN, 2008.

Hutchinson, Chuck, *The ARRL Handbook For Radio Amateurs*, 2001 edition, The American Radio Relay League, CT, 2000.

Rector, B.E. et al.,*Industrial Electronics Reference Book*, Westinghouse Electric Corporation, John Wiley & Sons Inc., New York, NY, 1948.

"Selecting and Using RS-232, RS-422, and RS-485 Serial Data Standards" Application Note 723 (AN723), Maxim Integrated Products, 2000.

Appendix F

Version history

This is a list showing all significant additions, corrections, and other edits made to this learning module. Each entry is referenced by calendar date in reverse chronological order (newest version first), which appears on the front cover of every learning module for easy reference. Any contributors to this open-source document are listed here as well.

14 March 2026 – corrected a typographical error in the Tutorial.

8 January 2026 – added more Quantitative Reasoning questions.

24 February 2025 – added image_3688 to the EIA/TIA-232 section of the Tutorial reviewing NRZ-encoded data frames.

10 February 2025 – added a new Challenge question to the "Terminating and bias resistors" Quantitative Reasoning question.

9 November 2024 – edited one of the "Challenging concepts" listed in the Introduction chapter, single-ended versus differential voltage signals.

22 October 2024 – included a reference on loopback connections for RS-232 in the Tutorial.

10-15 September 2024 – divided the Introduction chapter into sections, one with recommendations for students, one with a listing of challenging concepts, and one with recommendations for instructors. Also added some instructor notes.

9 September 2024 – added a new Case Tutorial section showing UART data frame examples with and without parity.

2-4 June 2024 – added new Tutorial section on the use of shift registers within all serial communication networks. Also made minor changes to the Introduction chapter.

21 February 2024 – clarified text prior to image_3893 to say that only the transmitter portion of an RS-485 device is shown there.

12 February 2024 – deleted a footnote in the Tutorial referencing a page/section that no longer existed.

7 February 2024 – added some index entries for noise margin to the Tutorial.

29 November 2022 – placed questions at the top of the itemized list in the Introduction chapter prompting students to devise experiments related to the tutorial content.

12 August 2022 – added comments on the meaning of "DB", "DE", and other connector designations.

11 March 2022 – added clarifications for male/female DE-9 connectors.

21 September 2021 – corrected a mistake in the Tutorial chapter where a "previous" illustration reference was in fact meant to refer to an illustration further back than that. Also added a Case Tutorial section showing a "Hello World" example of serial data transmitted by a PC and read by a digital oscilloscope, courtesy of Luke Jones.

2 August 2021 – added a Case Tutorial chapter with a section showing the use of an EIA/TIA-232 transceiver IC.

9 July 2021 – replaced some TeX-style italicizing markup with LaTeX-style.

10 May 2021 – commented out or deleted empty chapters.

15 February 2021 – added some questions to the Introduction. Also, capitalized "Ohm" in image_3899 for the "Terminating and bias resistors" question.

11 February 2021 – corrected instances where "Ohms" was uncapitalized.

28 December 2020 – added a Technical Reference section on using an oscilloscope to measure differential data signals.

21 September 2020 – minor edits to some questions and instructor notes.

1 September 2020 – significantly edited the Introduction chapter to make it more suitable as a pre-study guide and to provide cues useful to instructors leading "inverted" teaching sessions.

30 July 2020 – added footnote to the "EIA/TIA-232 data frames of ASCII characters" Conceptual reasoning question explaining how the oscilloscope screenshots were gathered.

25 June 2020 – added a Technical Reference section on the OSI Reference Model.

21 June 2020 – added some Foundational Concepts.

17 June 2020 – document first created.

Index

`Termite`, 11

Adding quantities to a qualitative problem, 70
Allen-Bradley Data Highway (DH) network, 29
Annotating diagrams, 69
ASCII, 11, 13
Asynchronous data transfer, 19

Bit, 16
Byte, 16

Carriage return, 12
Characteristic impedance, 30
Charge pump, 13
Checking for exceptions, 70
Checking your work, 70
Clock pulse, 16
Code, computer, 79
Common-mode voltage, 25
Control character, ASCII, 12

Data Communications Equipment, 23
Data Terminal Equipment, 22
DCE, 23
Differential signal, 19
Dimensional analysis, 69
Driver, 20, 25
DTE, 22

Edwards, Tim, 80
EIA/TIA-232 serial communication, 20
EIA/TIA-422 serial communication, 25
EIA/TIA-485 serial communication, 25
Error detection, 9
Even parity, 9

Flip-flop, 17
Flow control (serial data communication), 22

Graph values to solve a problem, 70
Greenleaf, Cynthia, 39

Handshaking (serial data communication), 22
How to teach with these modules, 77
Hwang, Andrew D., 81

ICL3232 serial transceiver, 13
Identify given data, 69
Identify relevant principles, 69
Idle condition, 21
Impedance, characteristic, 30
INTERBUS-S, 29
Intermediate results, 69
Inverted instruction, 77

Knuth, Donald, 80

Lamport, Leslie, 80
Limiting cases, 70
Linefeed, 12
Loopback jumper, 24

Mark state, 12, 21
Metacognition, 44
Moolenaar, Bram, 79
Murphy, Lynn, 39

NMEA-0183 data format, 13, 14
Noise margin, 20, 27
Null modem, 24
Nybble, 16

Odd parity, 9
Open-source, 79
Opto 22 Optomux network, 29

Parallel communication, 16

Parity, 11
Parity bit, 9, 21
Picoscope, 11
Positive logic, 8
Problem-solving: annotate diagrams, 69
Problem-solving: check for exceptions, 70
Problem-solving: checking work, 70
Problem-solving: dimensional analysis, 69
Problem-solving: graph values, 70
Problem-solving: identify given data, 69
Problem-solving: identify relevant principles, 69
Problem-solving: interpret intermediate results, 69
Problem-solving: limiting cases, 70
Problem-solving: qualitative to quantitative, 70
Problem-solving: quantitative to qualitative, 70
Problem-solving: reductio ad absurdum, 70
Problem-solving: simplify the system, 69
Problem-solving: thought experiment, 5, 69
Problem-solving: track units of measurement, 69
Problem-solving: visually represent the system, 69
Problem-solving: work in reverse, 70

Qualitatively approaching a quantitative problem, 70

Reading Apprenticeship, 39
Receiver, 20, 25
Reductio ad absurdum, 70, 76, 77
RS-232 serial communication, 20
RS-422 serial communication, 25
RS-485 serial communication, 25

SAM-M8Q GPS radio receiver, 13
Schoenbach, Ruth, 39
Scientific method, 44
Serial communication, 16
Simplifying a system, 69
Single-ended signal, 19
Single-ended signaling, 20
Socrates, 76
Socratic dialogue, 77
Space state, 12, 21
SPICE, 39
Stallman, Richard, 79
Swamping, 27
Synchronous data transfer, 19

Thought experiment, 5, 69
Torvalds, Linus, 79
Transceiver, 20, 25

Unbalanced signaling, 20
Units of measurement, 69
Universal Serial Bus, 20, 26
USB, 20, 26

Visualizing a system, 69

Word, 16
Work in reverse to solve a problem, 70
WYSIWYG, 79, 80