












USB

ESE3014

introduction

- in general, serial communication can refer to the transmission of a single bit of information at a time, over some communication channel
- thus, serial communication takes many forms:
 - USB, ethernet, I2C and SPI can be viewed as serial forms of communication
 - however, usually, serial communication refers to group of specific protocols
 - most notably, UART (USART being the synchronous version, with an explicit clock signal), and its physical-layer specific implementations such as RS-232 (which uses 12-V transmission) and “TTL” which refers to UART at either 3.3V or 5V
 - *bit-banging* is a term used to describe using GPIO pins to implement serial communications; this is sometimes needed if your embedded processor lacks the hardware serial interface, so one must create a software driver to emulate the hardware interface

- USB is a high-performance serial bus, capable of transmission speeds of hundreds or even thousands of megabits per second (over “medium distances” of a few metres)
- it is an asynchronous system, with no explicit clock and can work with up to 127 devices on a single bus
- USB uses differential signalling, which is partly why it can achieve such high speeds
- from Wikipedia (<https://en.wikipedia.org/wiki/USB>):
 - A broad variety of **USB hardware** exists, including eleven different **connectors**, of which **USB-C** is the most recent.
 - There have been four generations of USB specifications: USB 1.x, USB 2.0, USB 3.x, and USB4
 - The USB interface is self-configuring, eliminating the need for the user to adjust the device's settings for speed or data format, or configure interrupts, input/output addresses, or direct memory access channels.
 - USB has a strict tree network topology and master/slave protocol for addressing peripheral devices; those devices cannot interact with one another except via the host, and two hosts cannot communicate over their USB ports directly.
 - USB 2.0 was released in April 2000, adding a higher maximum signaling rate of 480 Mbit/s (60 MB/s)
 - USB 3.0 adds a SuperSpeed transfer mode, with associated backward compatible plugs, receptacles, and cables. The SuperSpeed bus provides for a transfer mode at a nominal rate of 5.0 Gbit/s
 - USB4 is based on the Thunderbolt 3 protocol specification. It supports 40 Gbit/s throughput, is compatible with Thunderbolt 3, and backwards compatible with USB 3.2 and USB 2.0

Connectors		USB 1.0 1996	USB 1.1 1998	USB 2.0 2001	USB 2.0 Revised	USB 3.0 2011	USB 3.1 2014	USB 3.2 2017	USB4 2019
Data rate		1.5 Mbit/s (Low Speed) 12 Mbit/s (Full Speed)	1.5 Mbit/s (Low Speed) 12 Mbit/s (Full Speed)	1.5 Mbit/s (Low Speed) 12 Mbit/s (Full Speed) 480 Mbit/s (High Speed)		5 Gbit/s (SuperSpeed)	10 Gbit/s (SuperSpeed+)	20 Gbit/s (SuperSpeed+)	40 Gbit/s (SuperSpeed+ and Thunderbolt 3)
Standard	A	<div>Type A</div> 				<div>Type A</div>  <div>Type-A SuperSpeed</div>			Deprecated
	B	<div>Type B</div> 				<div>Type B</div>  <div>Type-B SuperSpeed</div>			Deprecated
	C	N/A				<div>Type C (enlarged)</div> 			
Mini	A	N/A		<div>Mini A</div>  <div>Mini-A</div>		Deprecated			
	<div>Mini B</div>  <div>Mini-B</div>								
	AB	N/A			<div>Mini AB</div>  <div>Mini-AB</div>				
Micro	A	N/A							
	B	N/A			<div>Micro B</div>  <div>Micro-B</div>	<div>Micro B</div>  <div>Micro-B SuperSpeed</div>		Deprecated	
	AB				<div>Micro AB</div>  <div>Micro-AB</div>		Deprecated		
Connectors		USB 1.0 1996	USB 1.1 1998	USB 2.0 2001	USB 2.0 Revised	USB 3.0 2011	USB 3.1 2014	USB 3.2 2017	USB4 2019

- the USB communications device class (or USB CDC class) is used for networking devices.
- it provides an interface for transmitting Ethernet or ATM frames onto some physical media. It is also used for modems, ISDN, fax machines, and telephony applications for performing regular voice calls
- Virtual COM port (VCP) drivers cause the USB device to appear as an additional COM port available to the PC. Application software can access the USB device in the same way as it would access a standard COM port.
 - Refer to TN-101 (FTDI document) if you would like to learn more about VCP under Linux or if you need a custom VCP driver in Linux, however, VCP drivers are now integrated into the Linux kernel.

USB Virtual COM Port (VCP) Linux Lab

what you'll need:

- your FTDI TTL-232R-3V3 USB to TTL serial cable (often used to debug the Beaglebone)
- a Linux host machine with minicom installed
- a Beaglebone Black
- Derek Molloy's UART sample code (to run on your BBB), found here:
 - <https://github.com/derekmolloy/exploringBB/tree/version2/chp08>

1. boot your Linux host machine
2. insert the FTDI cable into your host machine's USB port
3. type `dmesg | tail`, and interpret what you are seeing
 - a. a new device should have been created; what is it called?
 - b. what kind of device is it?
 - c. are there any performance limitations on the USB device, now that a VCP driver is in place?
4. launch `minicom` on your host machine and associate it with this new device
5. set up your Beaglebone to perform serial communications over UART4
(consult the P8/P9 header tables, pp. 262-263)
6. using Derek Molloy's `uart.c` code as a starting point, can you transmit some sample strings to your host machine?
 - a. what is the maximum data rate at which you can transmit data over this USB-VCP channel?
 - b. what protocol are you using?