**ESE-3014 Lab**

**Interfacing communication – USB**

**Introduction**

1. 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 communiction
* 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 processer lacks

the hardware serial interface, so one must create a software driver to emulate

the hardware interface

1. 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)
2. it is an asynchronous system, with no explicit clock and can work with up to 127 devices on a single bus
3. USB uses differential signalling, which is partly why it can achieve such high speeds.
4. the USB communications device class (or USB CDC class) is used for networking devices.
5. 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.
6. 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.

# **Task**

what you’ll need:

* your FTDI TTL-232R-3V3 USB to TTL serial cable (often used to debug the Beaglebone)
* Linux host machine with minicom installed
* a Beaglebone Black

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 at 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