Lab 2: Intro to ADC and USART

OBJECTIVES

This lab will further introduce you to the concept of developing with a microcontroller. Focus will be placed on the use of the Analog to Digital Converter (ADC) system as well as the Universal Synchronous / Asynchronous Receiver/Transmitter (USART).

THIS Lab expects that you have an understanding of programming and simple circuit design. If at any time you are lost, please ask for clarification.

REQUIRED MATERIALS

- Epiphany-DAQ board
- Wire Jumpers
- IR Rangefinder Sensor
- LED
- Resistors
- Breadboard
- Multimeter (if needed)

DISCUSSION

In this section we will review common concepts for developing a simple circuit and program as this lab will review. Please read the sections that you may be unfamiliar with from the following list:

- Analog Devices
- ADC
- IR Rangefinder
- USART
- Terminal Emulator

Analog Devices

An analog device is a component that typically outputs some sort of analog signal (a ranging voltage value). This signal can be connected to an Analog to Digital Converter (ADC) to read in a digital representation of the analog signal. Analog devices may come in the form of various sensors such as IR Rangefinders, Sonar, CDS cells, etc.

Most analog devices use a three wire connection: vcc (power), ground, and signal. To understand the wiring, you will typically review a datasheet for the device to understand how the wiring is setup.

ADC

An Analog to Digital Converter (ADC) system gives the ability to take an analog signal and convert it to a digital signal. A digital representation of an analog signal is only as accurate as how many bits are used to represent the signal. The process of conversion is called quantization, where ranges of analog values are tied to a specific digital value, i.e., 0-0.25 V is 000, 0.26 – 0.5 V is 001, etc.

To be specific, if the resolution was performed using 8 bits, then the number of quantization levels is:

$2^8 - 1 = 256 - 1 = 255$ quantization levels

The resolution of such a system depends on the range of voltage values that may be sent via the analog signal. For example, if the range of the sensor was -5 V to 5 V with 255 quantization levels:

$(10V - 0V) / 255 = 10V / 255 = 0.039$ V

To use an ADC system, it is typically simple enough to connect an analog signal to the ADC.

To use an ADC system on most microcontroller development boards, analog signals are typically connected via a single source pin, while ground and vcc pins are connected through the typical ports. The ADC value read in is compared to a reference value during the process. The reference value may be customized and given, or it may be configured to be set to ground.

This is the case with the Epiphany and Arduino boards, where an analog device is connected via a single line and the reference voltage is already configured to be ground for you.

The Epiphany-DAQ board is multi-purpose, so it requires a small modification. A single analog channel on the Epiphany-DAQ consists of an ADCx- and an ADCx+ pin, where **x** represents which ADC is being used (0 – 3). The ADCx- pin behaves as a reference pin. For our purposes, this pin can just be set to ground, while the ADCx+ pin is used as the single input pin for the analog device.

IR Rangefinder

For the ADC in this lab, you will use an IR Rangefinder similar to that shown in Figure 1. The sensor, information, and datasheet may be found at the following website for sparkfun:

https://www.sparkfun.com/products/8959.

In this case, see Figure 2 for the wiring diagram and note that V_0 is the analog signal wire.

Figure 1 IR Rangefinder

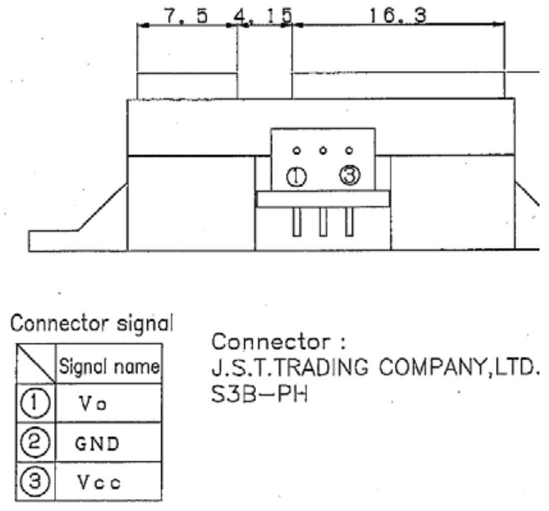


Figure 2 IR Rangefinder Wiring

USART

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) system is used for transmitting data between devices such as computers, microcontrollers, bluetooth, or RF devices. Data is a single direction transmission, where data is either going in or coming out at the same time, but not both directions.

A USART consists of TX and RX lines beyond the standard power and ground lines. When wiring the USART, which port to use is often found by reading your chip's datasheet and finding out which pins are used per USART. When connecting devices using the USART system, the TX line of device one connects to the RX line of device two. This is also done for the other path where the RX line of device one connects to the TX line of device two. Basically, the idea is to cross the lines when connecting devices.

1. Choose a specific USART port and find the TX and RX pins for that port.
2. Connect the RX of the device to the TX of the board.
3. Connect the TX of the device to the RX of the board.

For this lab, we will use the already wired and configured USB USART that the board has for debugging. For the Epiphany-DAQ, the USB is setup on PORTD USART channel 1.

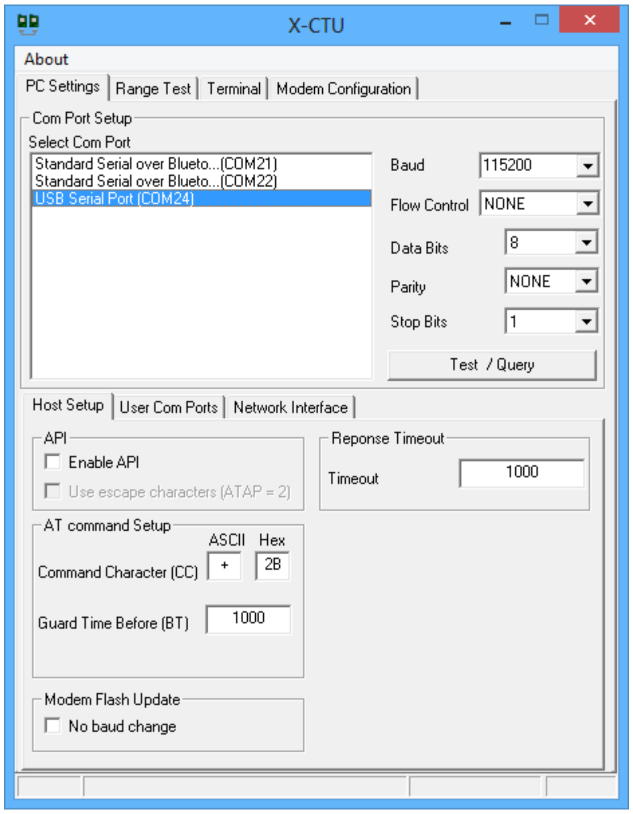
You will use a Terminal Emulator program to communicate between the device and your computer.

Terminal Emulator

When working with a USART device like you will in this lab, one end of the USART device is connected to the microcontroller while the other is connected to your computer. This then generates a communications (COM) port for that device. In some cases, you want to communicate to the device directly from your computer.

Terminal emulator applications are very useful in creating a connection between a device such as a USART and your computer. A suggested application that you may use is **X-CTU**, shown in Figure 3, which can be downloaded at (pick X-CTU Installer):

http://www.digi.com/support/productdetail?pid=3352&osvid=57&type=utilities



Com Port Setup
Select Com Port
Standard Serial over Blueto...(COM21)
Standard Serial over Blueto...(COM22)
USB Serial Port (COM24)

Baud: 115200
Flow Control: NONE
Data Bits: 8
Parity: NONE
Stop Bits: 1

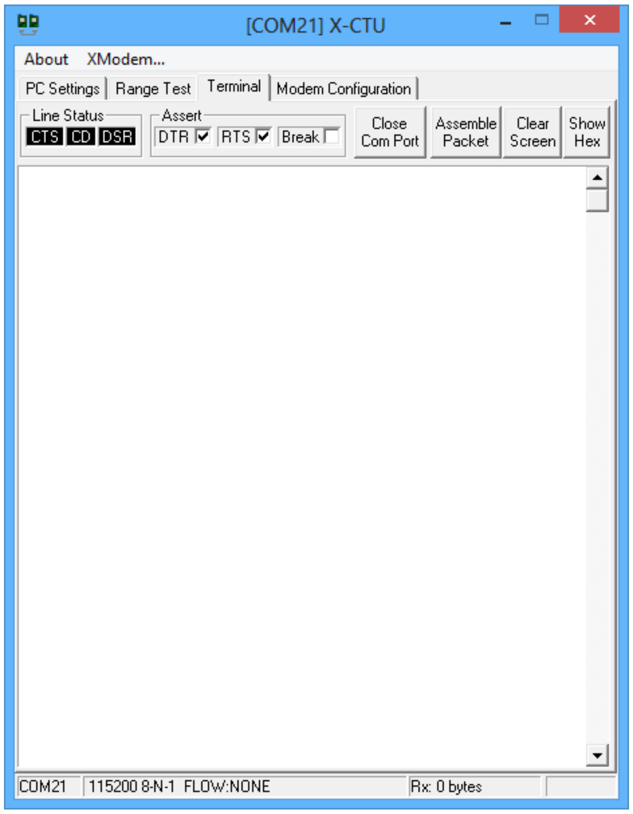
API
☐ Enable API
☐ Use escape characters (ATAP = 2)

Reponse Timeout
Timeout: 1000

AT command Setup
Command Character (CC): ASCII + Hex 2B
Guard Time Before (BT): 1000

Modem Flash Update
☐ No baud change

Figure 3 XCTU Terminal Emulator Setup



COM21 | 115200 8-N-1 FLOW:NONE | Rx: 0 bytes

Figure 4 XCTU Terminal Emulator Communication

To use X-CTU, use the following steps:

1. Choose which Com Port is your device (which can be seen by connecting and disconnecting your device while opening X-CTU).
2. You then set the USART settings on the right. Most default settings are normally fine, but you will probably need to change the Buad Rate.
3. You can then choose the Terminal tab to connect to the device and begin communication.
4. In Figure 4, you will see Red text to represent data received and Blue text to represent text transmitted. To transmit data, simply type in the text box.
5. If needed, you can close the Com Port to switch to another program that connects to the port.

<u>LAB PROCEDURE</u>

ADC and USART Program

Today you will continue learning how to use a microcontroller. The following sections will be used to create a program combining an Analog Device and USB USART. Steps 1 – 9 must be completed, while any further steps are for better understanding the USART system.

1. Setup LED circuit on breadboard.
2. Open Project Solution and create new file.
3. Wire ADC on any channel to read in the IR Rangefinder
4. Wire USART if required
5. Initialize the ADC
6. Initialize the USART
7. Read in the analog value from the IR Rangefinder
8. Print analog value out to the USB USART (great for future debugging)
9. If the analog value is within a certain threshold (distance from target), then trigger the LED to turn on. Experiment and choose this value yourself

REACH GOAL STEPS:

10. Initialize the debug LED

11. Setup the code to read in a single character from the USB USART. If that character is **'w'**, turn the debug LED on. If the character is **'s'** turn the debug LED off. You are welcome to choose whichever characters to use.

<u>Setting up the Program for Epiphany-DAQ</u>

First, collect the required LEDs, resistors, wires, IR Rangefinder, and breadboard. Next, setup an Active High LED circuit similar to that created for lab 1 (Figure 5). You should then take the input to the circuit and attach it to one of the PORT pins of the board.

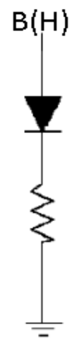


Figure 5 Active High LED Circuit

Now, you should create a new C file and copy the code in the blank.c file just as you did in lab 1.

ADC Setup

You will setup the IR Rangefinder device by wiring it to the Epiphany-DAQ. To do this, you use the following steps:

Choose one of the four ADC inputs (AI0, AI1, AI2, AI3)

1. **Connect the ADCx- pin to any G pin**
2. **Connect VCC (Power) to the 5V pin**
3. **Connect GND to any G pin**
4. **Connect the Analog Signal (V_0) to the ADCx+ pin. Make sure this signal is using the same ADC Port number as step 1**

USART Setup

There is no work needed to wire and setup the USB USART. At this point, you may wire up a separate device such as Bluetooth or XBee RF, but that is not covered by this lab. Just know that it is setup for **USARTD1**.

Programming the ADC

For the ADC, code must be added to the **Setup** function for initialization and to the **Loop** function for collecting the analog value. In the setup function, add the following initialization function to initialize the ADC system.

adcInit();

To read from the ADC, all that is needed is the function:

analogRead(x)

X is replaced by the ADC port number being used. The value returned is based on a resolution of 0 – 8192 (14 bit resolution). You can either use the exact value returned, or convert it into the real value by using the following equation:

*5 * (adcValue + 1) / 8192*

Programming the USART

For the USART, code must be added to the **Setup** function for initialization and to the **Loop** function for sending and receiving data over the USART. In the setup function, add the following initialization functions to initialize the USART system.

usartInit(&usartChannel, baudrate)
sei();

Select which channel you want to use for the USART based on your wired settings. Once the USART is selected (USARTD0, USARTD1, USARTE0, etc.), replace the usartChannel option with the correct value. Also enter a baudrate for the device into the function. For the USB USART, select **115200**. The **sei()** function is to initialize the interrupt system, which the USART uses.

Once the USART is initialized, the Epiphany is setup to send data via the function

fprintf(&usartStr, string)

Where usartStr is a preconfigured string (USB is set to **USB_str**). The string component of fprintf follows standard rules of printf functions, which can be found at http://www.cplusplus.com/reference/cstdio/printf/. If you are unfamiliar with printf and C programming, you may need help for this section.

To create your own usartStr, if needed, you can create your own string definition before any function in the included space by creating a precompiler definition:

```
#define USB_str      usartD1_str
```

An example of using the fprintf command is below:

fprintf(&USB_str, "%c", char_value);

Reading in data via the USART is done using the function

fscanf(&usartStr, string, variable)

The string component of the function also follows the printf standards. The fscanf function is used to read in characters via the USART buffer and then store it into a variable.

fscanf(&USB_str, “%c”, value_read);

When reading data, it is typically better to wait until data is on the USART buffer to attempt a read. The following function may be used to check if there is data in a port:

dataInBufXY()

X designates the PORT and the USART channel designated by **Y**. An example with an if Block is below:

```
if(dataInBufD1) {
    ……..
}
```

Programming the Debug LED

The Debug Led for the Epiphany is on PORT C pin 0. To initialize the Debug LED use the following command:

PORTC.DIRSET = 0x01; // For Epiphany-DAQ

To turn the LED on or off, use the same port described in part 5, but the OUTSET and OUTCLR functions like lab 1.