Data Acquisition Using USB

Acknowledgements: Developed by JD Neglia, P.E., Electronics Program Director at Mesa Community College, Mesa, Arizona.

Lab Summary: This laboratory experiment is designed to introduce practical concepts associated with data acquisition using the USB bus. Activities will be performed that will illustrate some example possibilities of data acquisition.

Lab Goal: The goal of this lab is to build a working USB data acquisition system, observe its operation, and perform measurements of its performance including speed and resolution.

Learning Objectives

1. Assemble a working USB interface circuit while following accepted ESD practices.
2. Evaluate the input and output capabilities of the system.
3. Observe the system operate as a simple, low speed oscilloscope.
4. Use the system to obtain the characteristic curve of a diode.

Grading Criteria: Your lab grade will be determined by your performance on the experiment, the lab questions, and the content and quality of your laboratory report (if assigned).

Time Required: Approximately 6 to 7 hours

Special Safety Requirements

Electrostatic discharge can damage the USB port of your computer as well as the microcontroller device used in this lab. Use appropriate ESD methods to protect the devices. A grounded wrist-strap is provided in the parts list for this circuit. Be sure to wear it at all times while handling the electronic components in this circuit. The wrist strap need not be worn after the circuit construction is complete.

No serious hazards are involved in this laboratory experiment, but be careful to connect the components with the proper polarity to avoid damage.

Lab Preparation

- Read the WRE Data Acquisition Module.
- Read this document completely before you start on this experiment.
- Acquire required test equipment and appropriate test leads.
- Gather all circuit components and the three-panel solderless breadboard.
- Print out the laboratory experiment procedure that follows.

Equipment and Materials

Each group of students will need the items listed in the tables below. It is suggested that students work in teams of two.

Equipment	***Quantity***
IBM PC or clone with at least one USB port	1
Digital Multimeter	1
Oscilloscope (20 MHz or greater)	1
Function Generator	1
Frequency Counter	1
PIC Programmer (Note: Only one PIC Programmer is required *per class*.)	1
Solderless Breadboard, 3-panel	1
Tools	***Quantity***
Needle-nose Pliers	1
Wire Strippers	1
Wire cutters	1
ESD Wrist Strap	1
Parts	***Quantity***
Microcontroller, PIC18F4550-I/P	1
Crystal, 20 MHz	1
Resistor, 200 Ohm, 1/4 W	2
Resistor, 1 kOhm, 1/4 W	4
Resistor, 2 kOhm, 1/4 W	1
Resistor, 10 kOhm, 1/4 W	1
Resistor, 100 kOhm, 1/4 W	2
Resistor, 1 MOhm, 1/4 W	1
Capacitor, 22 pF, disk	2
Capacitor, 0.1 uF, disk	2
Capacitor, 0.47 uF, electrolytic or tantalum	1
Capacitor, 1 uF, electrolytic or tantalum	1
Capacitor, 10 uF, electrolytic	1
Diode, 1N4004 (or *any* diode you would like to test)	1
Diode, Zener, 5.1 V, 500mW (1N751A or 1N5231B)	1
Pushbutton, N.O.	1
LED, Red	4
Potentiometer, 10 kOhm	1
USB A/B Cable	1
Cable Tie	2
Tie Mount	2
Wire (as needed)	-

Required Software: The files listed below contain all the software and firmware that will be needed to perform the exercises in this document.

File	***Description***	***Web URL***
software.zip	All USB Data Acquisition-specific code	www.jneglia.com
Ooo_2.0.2_Win32Intel_install_wJRE.exe	OpenOffice 2.0	www.OpenOffice.org

Optional Software: The files listed below are needed only if you will be writing your own data acquisition software.

File	***Description***	***Web URL***
Bloodshed Dev C++	C/C++ Compiler and IDE for Windows	www.bloodshed.org

All the above software is freely available at no charge. You will also need the Windows operating system, version 98SE or later. If you already have Microsoft Excel, you may use that rather than OpenOffice.

Introduction

Data acquisition systems exist in several forms but generally share some common characteristics. The most popular forms use a personal computer as a supervisory device which controls the operation of the data acquisition hardware, directs the collection of data, and stores, processes, and/or displays the results. This hardware often consists of GPIB or proprietary interfaces that are generally extremely expensive.

More recently, the advent of the USB bus and inexpensive USB hardware has made data acquisition much more readily available. This lab exercise focuses on data acquisition using the USB bus. In this exercise, you will build a data acquisition system with a total parts cost of less than $20. You will be able to take simple voltage measurements, plot the characteristic curve of diodes, and observe voltage waveforms with a very simple oscilloscope program.

If you know how to program in C, you will also be able to examine the source code that can communicate with the USB data acquisition device and modify it to make whatever measurements you like, or interface to any kind of system you desire.

Although this system is relatively low performance, it will give you hands-on experience that will prove useful using higher performance commercial data acquisition systems.

Illustration 1 gives a basic overview of the USB data acquisition system you will build. The user interacts with the personal computer to issue commands over the USB to the data acquisition board. The data acquisition board then responds to these commands by measuring the value of an analog or digital input and sending the results to the PC, or setting an analog or digital output as commanded by the PC.

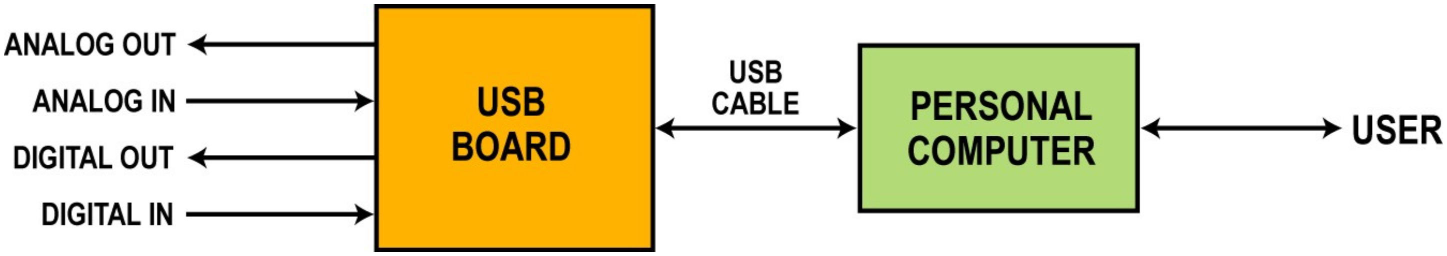


Illustration 1: Block Diagram of USB Data Acquisition System

Illustration 2 provides a more detailed view of the system. The USB board is the heart of the system. It contains a PIC 18F4550 microcontroller with an on-board USB interface as well as analog and digital inputs and outputs. The firmware for the system resides in memory on this chip. When this firmware receives an "analog input" command, it commands the ADC to begin a conversion, waits until the conversion is complete, reads the binary result, formats the result, and then sends the result over the USB to the PC. When the firmware receives an "analog output" command from the PC, it reformats and scales it appropriately and commands the on-board pulse-width modulator circuit to create a duty cycle proportional to the commanded voltage. The low pass filter then removes the high-frequency components from this PWM signal, leaving only a DC, analog signal approximately equal to the commanded output voltage.

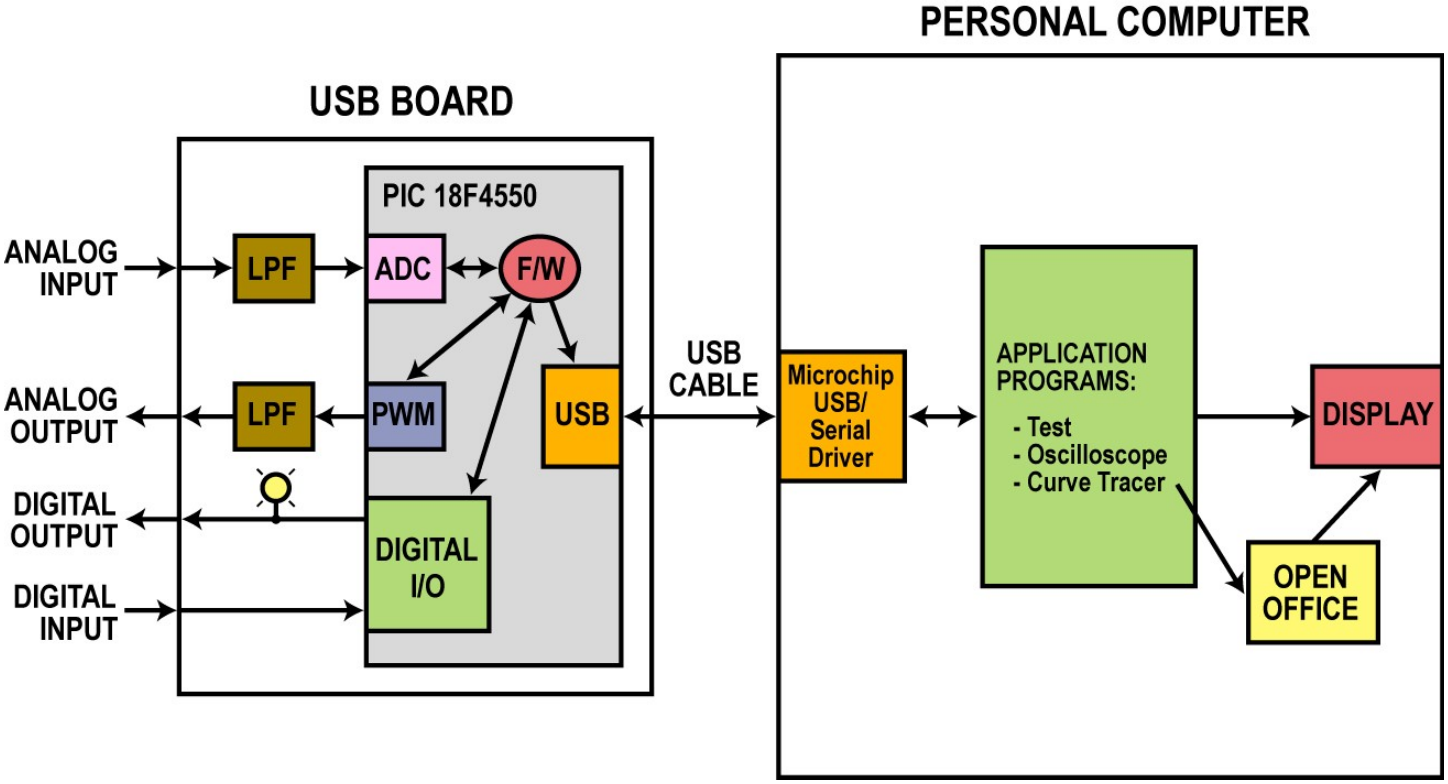


Illustration 2: Detailed View

When these analog and digital inputs and outputs are connected to signals of interest, these signals can then be collected and stored as data files on the PC. The PC can then analyze this data in a variety of ways. This document provides four different exercises.

In the first exercise, a program called UsbTest.exe allows you to simply command an analog or digital output and read an analog input. You can, for example, stimulate an analog circuit with the commanded voltage and then measure the resulting output of the circuit with the analog input.

In the second exercise, the PC runs a terminal program (usually HyperTerm since it is provided with Windows, but any other terminal program will work just as well.) When the user presses a button (which is a digital input) on the USB board, the value of the analog input is sent to the PC and displayed in the terminal program.

In the third exercise, a program called UsbScope.exe continuously monitors the analog input voltage and displays it on the system display in an "oscilloscope" fashion.

In the fourth exercise, a program called UsbTrace.exe stimulates an analog circuit with a linear voltage ramp, and monitors the analog input during this voltage ramp. In this way, you can plot the characteristic "I-V" curve of electronic components. An example is illustrated of obtaining the characteristic curve of a diode. The program UsbTrace.exe produces the voltage sweep and collects the resulting voltages, and saves them in a data file. This file is then imported into a spreadsheet (such as OpenOffice) and then plotted on a graph using the advanced graphing capabilities available.

The schematic diagram for the USB board provides the ultimate level of detail for this system. The schematic is shown after the appendices. The major component is the PIC microcontroller. The microcontroller and all the other circuits on this board obtain their power from the USB in the form of regulated 5.0 VDC. The USB signals are brought strait into the microcontroller from the USB. The microcontroller is clocked with a 20 MHz crystal oscillator.

The digital input provided on this board is Pin 37 of the microcontroller. It is stimulated with a simple pushbutton. The digital outputs are indicated with four LEDs. LEDs D1 and D2 are used to indicate the status of the USB communication link with the PC, and will blink alternately at about 2 Hz when the USB port is fully enumerated and functioning correctly. LEDs D3 and D4 are general-purpose digital outputs that can be toggled by the PC as needed by any application. In the example program UsbTest.exe, these two LEDs can be toggled on and off by pressing "3" or "4" on the PC keyboard. At this time, answer question 1 in the Questions and Report section following the lab procedure.

The analog output is created by applying a PWM output signal through a low pass filter. At this time, answer question 2.

The analog input to the system is first passed through an antialiasing low pass filter before reaching the analog input (pin 2) of the microcontroller. Since the input voltage range is 0.0V - 5.0 V, this analog input pin is also protected from over- and under- voltages with a 5.1V Zener diode. At this time, answer questions 3-5.

Three different types of circuits that may be attached to the USB board are also shown on the schematic.

Circuit "A" simply provides a DC voltage into the analog input and a way to display the analog output voltage. These circuits are mainly for functionality testing.

Circuit "B" is for loopback testing; the analog output is fed directly back into the analog input. Thus, when the PC commands a given analog voltage output, it should expect to read in a reasonably well matching analog input. Although this technique is somewhat less accurate than the previous technique, it doesn't require any external equipment (other than the jumper to connect the output to the input.)

Circuit "C" is the configuration necessary to measure the characteristic (I-V) curve of a diode. Answer question 6 at this time.

Lab Procedure

PART I: CONSTRUCTION OF THE DAQ

1. Examine the PIC18F4550 Data Sheet.
    a. Access the Microchip web site at www.microchip.com.
    b. Locate the data sheet for the 18F4550 microcontroller.
    c. Examine the first page of the data sheet. This device provides our circuit with a USB interface as well as simple analog input/output capabilities that we can use for our data acquisition functions.
2. Download the required firmware and software listed in the table at the beginning of this document.
    a. Run the OpenOffice file to install a MS-Office compatible suite of applications including a spreadsheet that will be useful in the curve tracing exercise later in this lab.
    b. Unzip the software.zip file to a convenient location on your hard drive. The instructions below assume that you have unzipped this file to a directory named c:\UsbDaq. Two subdirectories will be created with all the software you will need for these exercises. Although many files will be created when you extract this file, you will only need the files listed below.
    c. The other files that will be created are optional and will be needed only if you are a C programmer and will be developing your own data acquisition software.

File	*Description*
`mchpcdc.inf`	Microchip USB CDC driver. This will be installed on your PC.
`JdnUsb.hex`	PIC Firmware. This will be programmed into the PIC18F4550.
`UsbTest.exe`	Data Acquisition PC Test Program.
`UsbScope.exe`	Data Acquisition PC Oscilloscope Program
`UsbTrace.exe`	Data Acquisition PC Curve Tracer Program

3. Program the PIC18F4550. Use a PIC programmer to burn the JdnUsb.hex file into your PIC18F4550. (Note: Your instructor may have already performed this step for you.)
4. Build the circuit.
    a. There is nothing unusual about this circuit; build it using your standard solderless breadboard techniques.
    b. The 20 MHz crystal should be located adjacent to the appropriate microcontroller pins for proper operation.
    c. Lop off the "B" end of a standard USB A/B cable, strip back the outer insulation, and strip and tin the four inner wires. Note that on standard USB cables, Red is +5V, Black is GND, Green is D+, and White is D-. You can plug these tinned wires directly into the solderless breadboard, or, for a more rugged assembly, solder them to a standard header and then plug the header into the solderless breadboard. In either case, be sure to use the cable ties and mounts to keep the cable secure.

Note: A three-panel solderless breadboard is listed for this project. Although this will provide plenty of room for prototyping and experimentation, it is not strictly necessary. The entire circuit could be constructed on a two-panel board if desired.

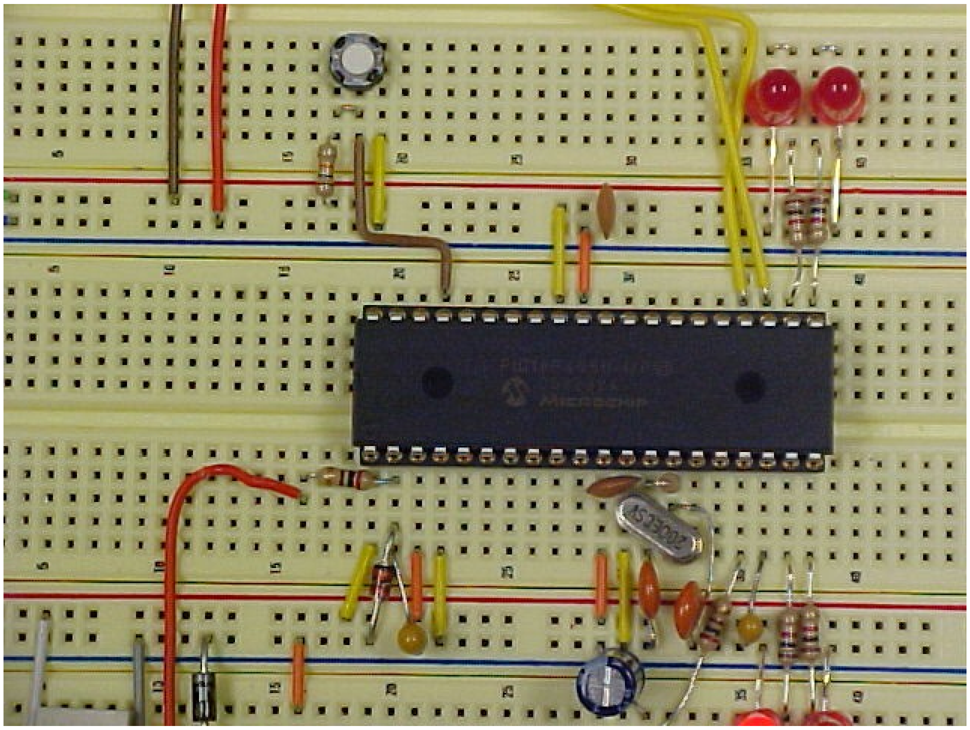


Illustration 3: Microcontroller Circuit

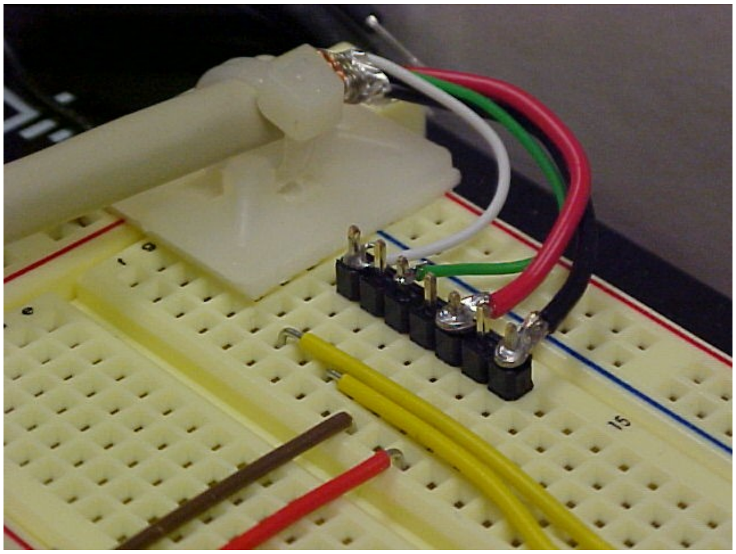


Illustration 4: Connecting the USB Cable to the Solderless Breadboard

5. Test the circuit.
   a. Now that you have completed assembling the circuit, double-check all your connections.
   b. Use an ohmmeter to verify that there is at least 50 Ohms resistance between +5V and GND.

PART II: INSTALLATION OF THE DAQ

6. Connect the DAQ to your PC and install the driver.
    a. To install the driver that is necessary for communication with the DAQ, you need to plug in a completed, correctly wired DAQ board.
    b. If there are wiring or component errors on the DAQ board, the circuit will not be detected by the PC, and the driver will not be installed. These instructions also assume that you have already downloaded and extracted the required software listed above.
    c. Since there are many different versions of Windows, with yet additional versions yet to be released, only general installation guidelines are provided here. If the actual selections you are offered differ from these, choose the closest approximations to these selections.

Windows Driver Installation:

If you have not already done so, login to your system as an Administrator.

Plug the USB DAQ into an available USB port on your computer.

Wait for the New Hardware Wizard to appear.

In the New Hardware Wizard, make the following selections:

- Connect to Windows Update? NO
- Select "Install from a specific location."
- Browse to c:\UsbDaq\Software\. (contains the file mchpcdc.inf).

At some point you will be "strongly recommended" not to proceed because this driver has not passed "Windows Logo Testing." Some versions of Windows will encourage you to contact the manufacturer to tell us we need to pass "Windows Logo Testing". Feel free to send me an e-mail. I'll take it into consideration. Meanwhile, Microsoft has very generously provided a button which allows you to "Continue Anyway". Select it.

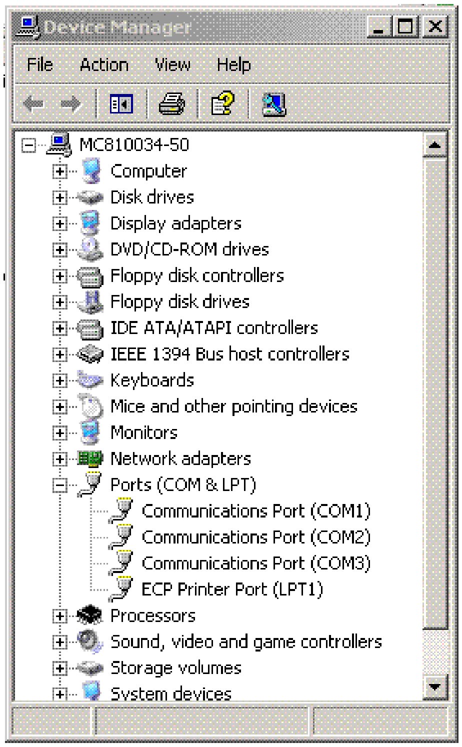
Wait while Windows installs the necessary driver.

Select "Finish".

    d. If everything worked, and your hardware was built correctly, you will get a "Your hardware is installed and ready to use" message. Also, LED's 1 and 2 on the DAQ should be blinking alternately. If you do not observe these two events, find the error in your wiring, fix it, and try again.

7. Determine which COM port is being used.
    a. Open the Windows Device Manager. Usually, but not always, you can get to this by selecting <Control Panel>, <System>, <Hardware>, and then finally <Device Manager>.
    b. In the Device Manager, expand the "Ports (COM & LPT)" entry. This will show you a list of the serial and parallel ports in your system. If your system already had two serial ports (most do), then Windows most likely assigned your DAQ to COM3 or COM4.

Illustration 5: Device Manager



c. Unplug the DAQ from the USB port. Notice this new serial port disappears. Plug it back in and the serial port reappears. Note the number of the serial port that appears; you will need it to run the application programs below.

PART III: OPERATION OF THE DAQ

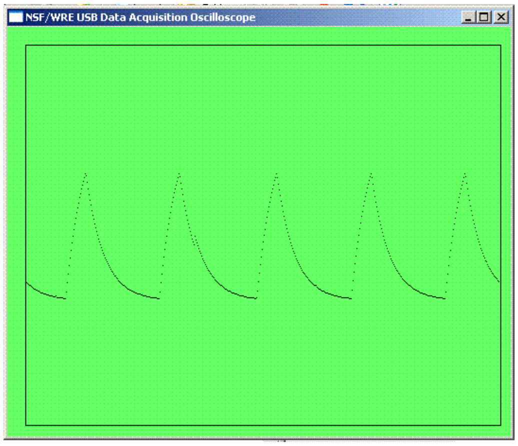
8. Run the basic test program.
    a. Unplug the USB DAQ from your PC.
    b. Connect circuit "B" (the Loopback mode) to the analog output/input connections.
    c. Connect a DMM in voltage mode to the analog output.
    d. Connect in the USB DAQ board to your PC. LED's D1 and D2 should begin to blink alternately.
    e. Open a "Command Prompt" window, and browse to c:\UsbDaq\Software\UsbTest\.
    f. Type "dir" to verify that the file UsbTest.exe exists.
    g. To run this program, enter the command UsbTest com3 where "com3" is the comm port that Windows has assigned to your USB DAQ board, as shown in the Device Manager.

h. The program will run and you will be presented with a menu of options. See the sample run below.

```
 -------------- USB DATA ACQUISITION MAIN MENU ----------------
                Rev: 2006 Jun 05, JD Neglia, PE

 A - display ADC input voltage
 S - Set pwm output voltage
 3 - Toggle LED 3
 4 - Toggle LED 4
 V - display firmware revision
 M - display this Menu again
 X - eXit (mandatory before unplugging USB device)
 --------------------------------------------------------------

Using comm port com3

You pressed v.  The firmware rev. is: Data Acq F/W V1.0

You pressed s.  Enter analog value in mV [0-5000]: 2500
PWM has been set.

You pressed a.  The ADC voltage is: 2460mV.

Led3 toggled.
Led3 toggled.
Led4 toggled.
```

9. Use the program to characterize the accuracy of the conversions.
    a. Complete the table in the Questions and Report section, and then graph the error as a function of the commanded voltage using a spreadsheet such as OpenOffice or Excel.
    b. Print out your graph and attach it to this document.
10. Run the digital input test program.
    a. Close the UsbTest program or any other programs that are communicating with the USB DAQ board, and run a terminal program such as Hyperterm or Teraterm.
    b. Configure it to use the comm port that Windows has assigned to your USB DAQ board.
    c. Press S2 on the DAQ board. The voltage that is present on the analog input will be sent to the PC and displayed in the terminal program in the format RA0xxxx, where xxxx is the analog voltage in millivolts.

11. Run the Oscilloscope Demo. The Oscilloscope Demo is a Windows program that continuously monitors the analog input and displays the voltage found in an oscilloscope style window. Run this program as follows:
    a. Shutdown any other programs that are communicating with your USB DAQ.
    b. Connect a function generator to the analog input.
        i) Set the output amplitude of the function generator to MINIMUM.
        ii) Set the waveform to SQUARE.
        iii) Set the DC OFFSET to about 2.5 V. (Do not apply negative voltages to this system, or damage to the microcontroller may result.)
12. Connect your USB DAQ board to your PC, and verify that it is detected and both LED's are blinking alternately.
13. Using Windows Explorer, browse to c:\UsbDaq\Software\UsbScope. Note that there are four programs, named UsbScope1 through UsbScope4.
    a. Select the one that corresponds to the Comm port that you are using and run it. You should see a window similar to this.



    b. Slowly increase the amplitude setting on your function generator, and verify that a sine wave is displayed. (Note: The waveform shown here is the charge/discharge cycle of the capacitor in a 555 oscillator circuit.)
14. "Calibrate" this window by determining the time base of this oscilloscope display, as follows:
    a. Set the function generator to "square wave" mode, and adjust it's frequency so that one complete period fills the display.
    b. Calculate this period. The result is the sweep time for this oscilloscope display. Note that in a commercial grade DAQ system, both the time base and the vertical gain would be adjustable just like on a real oscilloscope.

Note: The above technique will require a very low frequency wave. If your function generator will not go that low, here is an alternative technique for measuring the sweep speed.

c. Count the number of screen sweeps that occur in 30 seconds. Use a stopwatch, or have your lab partner give you "start" and "stop" signals.

d. Divide the number seconds by the number of sweeps you counted.

15. Answer questions 1 and 2 in the Post Lab Questions and Report section

16. While the oscilloscope is running, open a multimedia application in Windows (i.e. play an MP3 audio file or a video file). Answer question 3.

17. Run a curve trace on a diode.

    a. Unplug the USB DAQ from your PC.

    b. Connect circuit "C" (the diode/resistor) to the analog output/input connections.

    c. Connect in the USB DAQ board to your PC. LED's D1 and D2 should blink alternately.

    d. Open a "Command Prompt" window, and browse to c:\UsbDaq\Software\UsbTrace\. Then type "dir" to verify that the file UsbTrace.exe exists. To run this program, enter the command UsbTrace com3 where "com3" is the comm port that Windows has assigned to your USB DAQ board, as shown in the Device Manager.

    e. The program will run and you will be presented with a menu of options. See the sample run below.

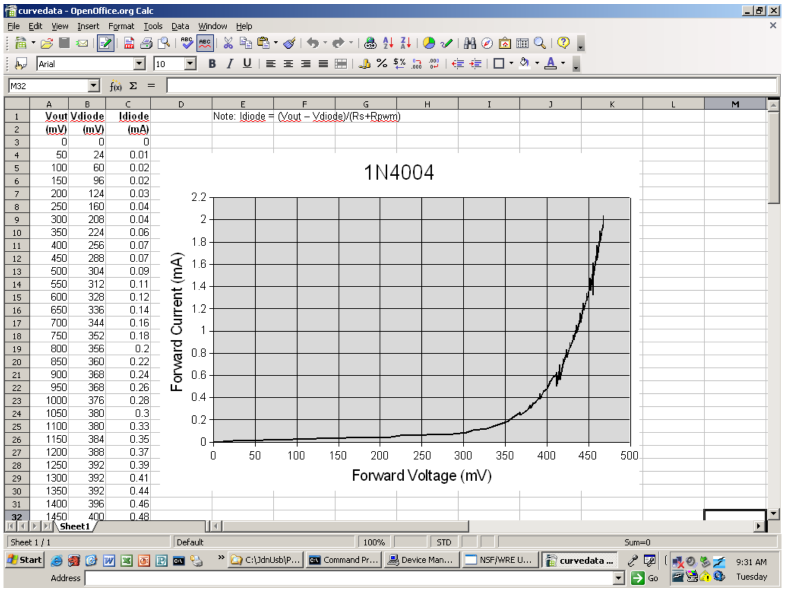
```
 -------------- USB DATA ACQUISITION MAIN MENU ----------------
                Rev: 2006 Jun 05, JD Neglia, PE

                     EXAMPLE PROGRAM:
                   --- CURVE TRACER ---

 C - Collect data
 E - Examine data
 S - Save data
 V - display firmware revision
 M - display this Menu again
 X - eXit (mandatory before unplugging USB device)
 --------------------------------------------------------------

Using comm port com3

You pressed c.  Hit any key to collect data...
Now collecting data...

                      (LARGE DATA TABLE OMITTED HERE)

Done collecting data.  Now you may (e)xamine and (s)ave the data.

You pressed e to Examine the data... (All data is shown in millivolts.)
```

```
                    (LARGE DATA TABLE OMITTED HERE)

You pressed s to Save the data...
Saving with filename curvedata.txt...
Done saving.

You pressed v.  The firmware rev. is: Data Acq F/W V1.0

You pressed x.  Now exiting...
Press any key to continue . . .
```

18. Run OpenOffice or Excel, import the data file (curvedata.txt) that you saved above, and calculate the diode current.
    a. Note that since we did not measure the diode current directly, we must calculate it.
    b. Since the resistor and diode are in series, the current through the diode is the same as the current through the resistor.
    c. We can calculate the current through the resistor since we know the voltage on both sides of the resistor: the analog output voltage, and the diode voltage. The equation is

$$I_{DIODE} = (V_{OUT} - V_{DIODE})/(R_S + R_{PWM}).$$

    d. Create a new column for the diode current, and then graph it as a function of the diode voltage. If you are using OpenOffice, your result should appear similar to the figure below.
19. Print out the first page of your spreadsheet (you may omit the rest of the data) and attach it to this document with the rest of your results.



Note: Idiode = (Vout – Vdiode)/(Rs+Rpwm)

Vout (mV)	Vdiode (mV)	Idiode (mA)
0	0	0
50	24	0.01
100	60	0.02
150	96	0.02
200	124	0.03
250	160	0.04
300	208	0.04
350	224	0.06
400	256	0.07
450	288	0.07
500	304	0.09
550	312	0.11
600	328	0.12
650	336	0.14
700	344	0.16
750	352	0.18
800	356	0.2
850	360	0.22
900	368	0.24
950	368	0.26
1000	376	0.28
1050	380	0.3
1100	380	0.33
1150	384	0.35
1200	388	0.37
1250	392	0.39
1300	392	0.41
1350	392	0.44
1400	396	0.46
1450	400	0.48

PART IV: PROGRAMMING PROJECT (OPTIONAL -- for C programmers only)

The source code in C for the test program UsbTest.exe is provided in the appendix and in the software download that you installed in this lab exercise. Examine this source code listing to see how the program works. You can use this code as a starting point for your own projects. You can modify this code to do anything you like. If you can think of some other application that you would like to develop, write a brief proposal to your instructor. If your instructor approves, proceed with your project. Have fun!

Questions and Report

Introduction Questions:

1. What is the diode current of these diodes when illuminated? (Assume $V_F = 1.7$ V)

2. What is the cutoff frequency of this LPF?

3. What is the cutoff frequency of the antialiasing filter?

4. If a 10V DC signal is connected to the analog input, what is the maximum voltage that Pin 2 of the microcontroller will see?

5. If a -6V DC signal is connected to the analog input, what is the minimum voltage that Pin 2 of the microcontroller will see?

6. What is the maximum current that this circuit can apply to the diode?

7. What is the measured oscilloscope window sweep time?

Lab Table

LOOPBACK TEST

Commanded (mV)	***DAQ Displayed (mV)***	***DMM (Actual) (mv)***	***Error (mV)***
0			
500			
1000			
1500			
2000			
2500			
3000			
3500			
4000			
4500			
5000			

Post Lab Questions

8. Question: The oscilloscope window displays 500 data points per screen. How many samples per second is the data acquisition system displaying?

9. Question: Each sample that the data acquisition system takes has 10-bit resolution. How many bits per second is the data acquisition system displaying?

Note also that this sweep time is not constant; it will depend on what other tasks your computer may also be running. It also depends on the clock speed of your computer.

10. Does the sweep speed of the oscilloscope change when Windows is running the multimedia application?

Lab Report

Write a brief report on the current state of the art in data acquisition systems. Do some online research on commercial data acquisition systems, and be sure your report addresses the following issues:

1. What are the most popular interface bus technologies?
2. What approximate maximum speeds are associated with these buses?
3. What accuracy and resolutions are available?
4. What are the approximate costs for these systems?
5. What programming languages are commonly used for these systems? Are they proprietary? Are they standard?
6. Finally, address how well the data acquisition system we built compares with the commercial systems on all of these dimensions.

APPENDIX: Test Program Source Code

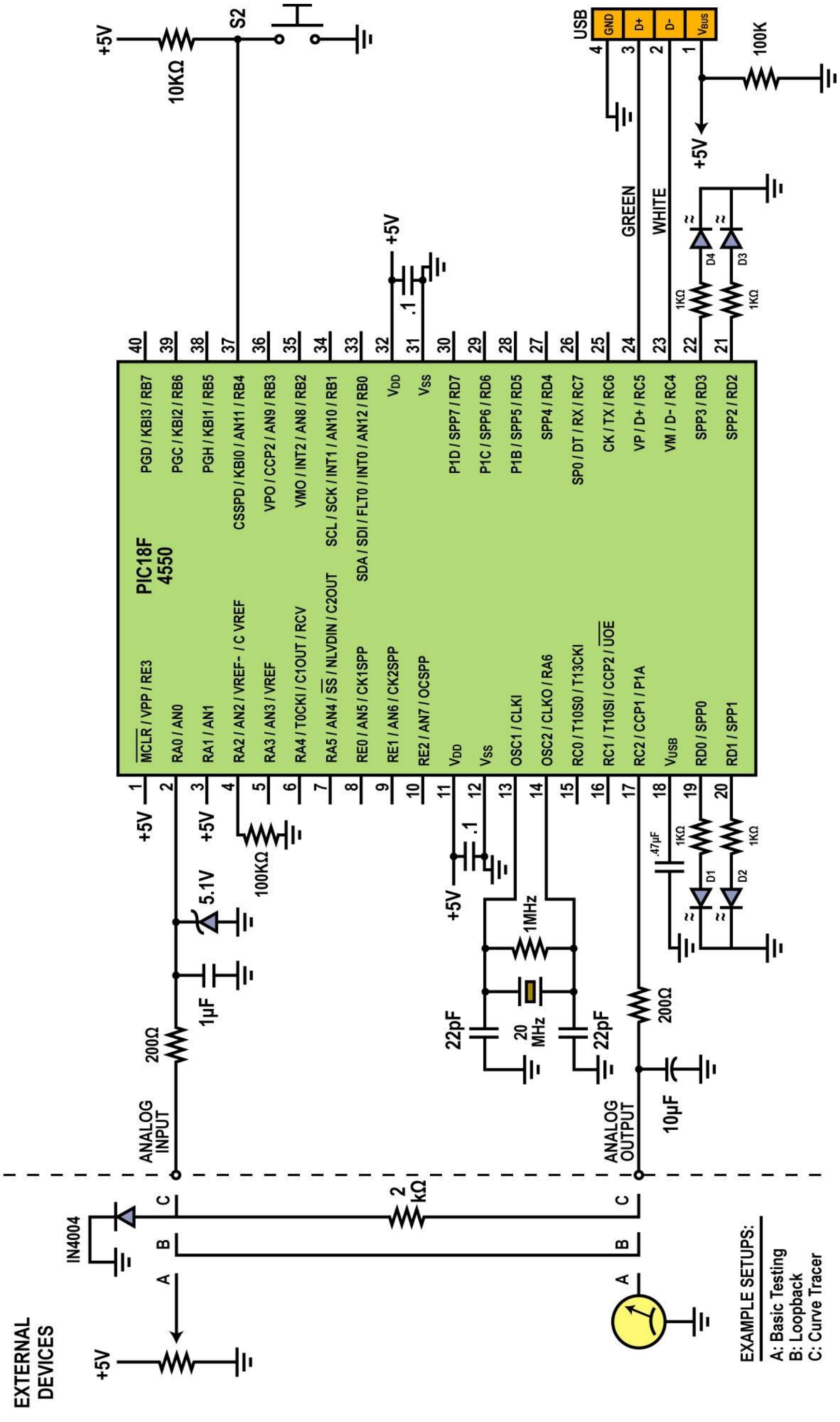
```
/*
National Science Foundation (NSF) (http://www.nsf.gov)
Maricopa Advanced Technology Education Center (MATEC) (http://www.matec.org)
USB DATA ACQUISITION SYSTEM
J.D. Neglia, P.E. (http://www.jneglia.com)

File:     UsbTest.cpp
Compiler: Bloodshed DevC++ 4.9.9.2

Rev: 2006 Jun 05: Initial Release
*/

#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <windows.h>
#include <conio.h>    // for kbhit()

using namespace std;

void DisplayWindowsErrorMessage();
void ClearXmitBuff(char * szWrBuff);
void DisplayMenu();

int main(int argc, char *argv[])
{
   int  PwmCmdInt, AdcMeas;
   char PwmCmdBuff[10], szAnalog[5], UserChar;
   char szBuff[11] = {0}, szWrBuff[6] = {0};
   DWORD dwBytesRead = 0, dwBytesWritten = 0;

   if (argc != 2)
   {
      cout << "\nYou must specify the COM port in the command line, "
           << "such as \n\n\tc:\\>UsbTest COM3 \n\n"
           << "Please retry!\n" << endl;
      exit(0);
   }
   DisplayMenu();
   cout << "Using comm port " << argv[1] << endl;

   // SERIAL PORT: OPEN
   HANDLE hSerial;
   hSerial = CreateFile(argv[1],
                       GENERIC_READ | GENERIC_WRITE,
                       0,
                       0,
                       OPEN_EXISTING,
                       FILE_ATTRIBUTE_NORMAL,
                       0);

   // SERIAL PORT: CHECK FOR ERRORS
   if (hSerial == INVALID_HANDLE_VALUE)
      DisplayWindowsErrorMessage();

   // SERIAL PORT: SET PARAMETERS  (most settings are arbitrary!)
   DCB dcbSerialParams = {0};
```

```
dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
   if (!GetCommState(hSerial, &dcbSerialParams))
      cout << "Error: Serial Port: cannot get port state." << endl;
   dcbSerialParams.BaudRate = CBR_19200;
   dcbSerialParams.ByteSize = 8;
   dcbSerialParams.StopBits = ONESTOPBIT;
   dcbSerialParams.Parity   = NOPARITY;
   if (!SetCommState(hSerial, &dcbSerialParams))
      cout << "Error: Serial Port: cannot set port state." << endl;
   COMMTIMEOUTS timeouts = {0};
   timeouts.ReadIntervalTimeout = 50;
   timeouts.ReadTotalTimeoutConstant = 50;
   timeouts.ReadTotalTimeoutMultiplier = 10;
   timeouts.WriteTotalTimeoutConstant = 50;
   timeouts.WriteTotalTimeoutMultiplier = 10;
   if (!SetCommTimeouts(hSerial, &timeouts))
      cout << "Error: Serial Port: cannot set timeouts." << endl;

   // ----MAIN LOOP ----
   while (1)
   {
      // IF A KEY IS PRESSED, SEND A MESSAGE TO USB DATA ACQUISITION SYSTEM
      if (kbhit())
      {
         UserChar = tolower(getch());     // Get the key that was pressed.
         switch (UserChar)
         {
         case 'v':
            cout << "\nYou pressed v.  The firmware rev. is: " << flush;
            ClearXmitBuff(szWrBuff);
            szWrBuff[0] = 'v';
            break;

         case 'a':
            cout << "\nYou pressed a.  The ADC voltage is: " << flush;
            ClearXmitBuff(szWrBuff);
            szWrBuff[0] = 'a';
            szWrBuff[1] = 0;     // NULL termination
            break;

         case 's':
            do {
               cout << "\nYou pressed s.  Enter analog value in mV [0-5000]: "
                    << flush;
               cin >> PwmCmdInt;
               }
            while (PwmCmdInt > 5000 || PwmCmdInt < 0);
            PwmCmdInt = PwmCmdInt*270/5000;  // scale for PWM
            itoa(PwmCmdInt, PwmCmdBuff, 10);  // ascii result is left justified;
            // PwmCmdBuff[0] always gets MSD,
            // so commanded integer will be 1,2,3, or 4 chars long.
            ClearXmitBuff(szWrBuff);
            szWrBuff[0] = 's';
            szWrBuff[1] = PwmCmdBuff[0]; // MSD ascii goes here.
            szWrBuff[2] = PwmCmdBuff[1]; // possible LSD
            szWrBuff[3] = PwmCmdBuff[2]; // possible LSD
            szWrBuff[4] = PwmCmdBuff[3]; // possible LSD
            szWrBuff[5] = 0;             // NULL termination
            break;

         case '3':
```

```
ClearXmitBuff(szWrBuff);
            szWrBuff[0] = '3';
            szWrBuff[1] = 0;     // NULL termination
            break;

         case '4':
            ClearXmitBuff(szWrBuff);
            szWrBuff[0] = '4';
            szWrBuff[1] = 0;     // NULL termination
            break;

         case 'x':
            cout << "\nYou pressed x.  Now exiting..." << endl;
            goto done;

         case 'm':
            DisplayMenu();
            goto SkipSend;

         default:
            cout << "\nYou pressed " << UserChar
                 << ", which is an invalid character." << endl;
            goto SkipSend;
         } /* switch */

         if (!WriteFile(hSerial, szWrBuff, 5 /* chars */, &dwBytesWritten, NULL))
            cout << "Error: Serial Port: cannot write characters." << endl;

         // Since a key was just pressed and a command was just sent,
         // wait for a response from the USB device.
         // SERIAL PORT: READ MESSAGE FROM SERIAL PORT
         if (!ReadFile(hSerial, szBuff, 20 /* chars */, &dwBytesRead, NULL))
            cout << "Error: Serial Port: cannot read characters." << endl;
         // DISPLAY RESPONSE MESSAGE FROM USB DEVICE
         if (szBuff[0] == 'R')  // analog reply measurement?
         {
            szAnalog[0] = szBuff[3];
            szAnalog[1] = szBuff[4];
            szAnalog[2] = szBuff[5];
            szAnalog[3] = szBuff[6];
            szAnalog[4] = '.';       // non-digit char to end the integer
            AdcMeas = atoi(szAnalog);
            AdcMeas *= 5000;
            AdcMeas /= 1023;
            cout << AdcMeas << "mV." << endl;
            ClearXmitBuff(szWrBuff);
         }
         else
         {
            cout << szBuff << endl;  // display the buffer
            ClearXmitBuff(szWrBuff);
         }
         SkipSend:; // invalid key pressed, so skipped sending command to USB
      } /* if kbhit */
   } /* while (1) */
   done:

   CloseHandle(hSerial);            // SERIAL PORT: CLOSE
   system("pause");
   return 0;
```

```
} /* main */


/*----------------------------------------------------------------------------*/
/* CLEAR THE TRANSMIT BUFFER */
/*----------------------------------------------------------------------------*/
void ClearXmitBuff(char * szWrBuff)
{
   for (int i = 0; i < 7; i++) szWrBuff[i] = 0;
}

/*----------------------------------------------------------------------------*/
/* DISPLAY THE MENU */
/*----------------------------------------------------------------------------*/
void DisplayMenu()
{
   system("cls");
   cout << " -------------- USB DATA ACQUISITION MAIN MENU ----------------\n"
        << "                Rev: 2006 Jun 05, JD Neglia, PE               \n"
        << "                                                              \n"
        << " A - display ADC input voltage                                \n"
        << " S - Set pwm output voltage                                   \n"
        << " 3 - Toggle LED 3                                             \n"
        << " 4 - Toggle LED 4                                             \n"
        << " V - display firmware revision                                \n"
        << " M - display this Menu again                                  \n"
        << " X - eXit (mandatory before unplugging USB device)            \n"
        << " --------------------------------------------------------------\n"
        << endl;
}

/*******************************************************************************
* Function:                                                                    *
*            Convert and display last windows error message                    *
*******************************************************************************/
void DisplayWindowsErrorMessage()
{
   char LastError[1024];
   FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS,
                 NULL,
                 GetLastError(),
                 MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                 LastError,
                 1024,
                 NULL);
   cout << LastError << endl;
}
```

Schematic


PIC18F 4550
MCLR / VPP / RE3
RA0 / AN0
RA1 / AN1
RA2 / AN2 / VREF- / C VREF
RA3 / AN3 / VREF
RA4 / T0CKI / C1OUT / RCV
RA5 / AN4 / SS / NLVDIN / C2OUT
RE0 / AN5 / CK1SPP
RE1 / AN6 / CK2SPP
RE2 / AN7 / OCSPP
VDD
VSS
OSC1 / CLKI
OSC2 / CLKO / RA6
RC0 / T1OS0 / T13CKI
RC1 / T1OSI / CCP2 / UOE
RC2 / CCP1 / P1A
VUSB
RD0 / SPP0
RD1 / SPP1
PGD / KBI3 / RB7
PGC / KBI2 / RB6
PGH / KBI1 / RB5
CSSPD / KBI0 / AN11 / RB4
VPO / CCP2 / AN9 / RB3
VMO / INT2 / AN8 / RB2
SCL / SCK / INT1 / AN10 / RB1
SDA / SDI / FLT0 / INT0 / AN12 / RB0
VDD
VSS
P1D / SPP7 / RD7
P1C / SPP6 / RD6
P1B / SPP5 / RD5
SPP4 / RD4
SP0 / DT / RX / RC7
CK / TX / RC6
VP / D+ / RC5
VM / D- / RC4
SPP3 / RD3
SPP2 / RD2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21
+5V
10KΩ
S2
USB
GND
D+
D-
VBUS
4 3 2 1
100K
GREEN
WHITE
.1
1KΩ
D4
D3
D1
D2
.47µF
100KΩ
5.1V
1µF
200Ω
ANALOG INPUT
1MHz
22pF
20 MHz
ANALOG OUTPUT
10µF
EXTERNAL DEVICES
IN4004
2 kΩ
A
B
C
EXAMPLE SETUPS:
A: Basic Testing
B: Loopback
C: Curve Tracer