LUFA & Studio5 Beginner's Guide
Part 3: Create New Project with USB CDC Connectivity

Introduction

In Part 2 of this guide, the VirtualSerial Demo in the Studio5 project was modified so that it would run on an OLIMEX162 board that uses an AVR at90usb162 microcontroller. In Part 3 we create a Studio5 New Project that uses LUFA to provide USB CDC connectivity.

Please keep in mind that this guide is "*a* way of doing it", not "*the* way of doing it".

LUFA (Lightweight USB Framework for AVRs), written and maintained by Dean Camera, is available at:
http://www.fourwalledcubicle.com/LUFA.php.

This guide uses Studio5 (5.1.208) and LUFA-120219.

The source code files used in Part 3 can be downloaded here:
http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&t=119781

Part 1: Compiling the VirtualSerial Demo can be found here:
http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&t=118597

Part 2: Modifying VirtualSerial for OLIMEX162 Board can be found here:
http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&t=119003

Caveat Emptor

I am not an expert in AVR, Studio5, or LUFA. In fact, I'm a rank beginner in both Studio5 and LUFA. I hope this guide helps you get started with LUFA and speeds you on the way to building USB enabled apps.
No warranty expressed or implied. YMMV.

Before We Start

So far in this guide, we have imported the LUFA VirtualSerial demo program into Studio5, modified it to compile, and then modified it again to work with the OLIMEX186 board. But our ultimate goal is not to run demo programs or work with development boards. Our goal is to be able to easily add USB CDC (serial port) connectivity to new programs that will run on whatever board we choose, whether an existing board or a new custom board. LUFA makes this possible and Part 3 of this guide shows how to do it.

We start by creating a new project in Studio5. Then we copy files into the project folder, add them to the project, and then copy and paste text into the main program file (the file with the main() function). We then modify the project properties to use an external makefile and edit the makefile to work with this new Studio5 project.

All in all, a straight forward and simple process. Let's get started.

Step 1: Create a New Project in Studio5

In Studio5 select File > New > Project. In the pop-up window select “AVRGCC C Executable Project” and provide a name for the project (this guide uses Widget_CDC). Click OK.

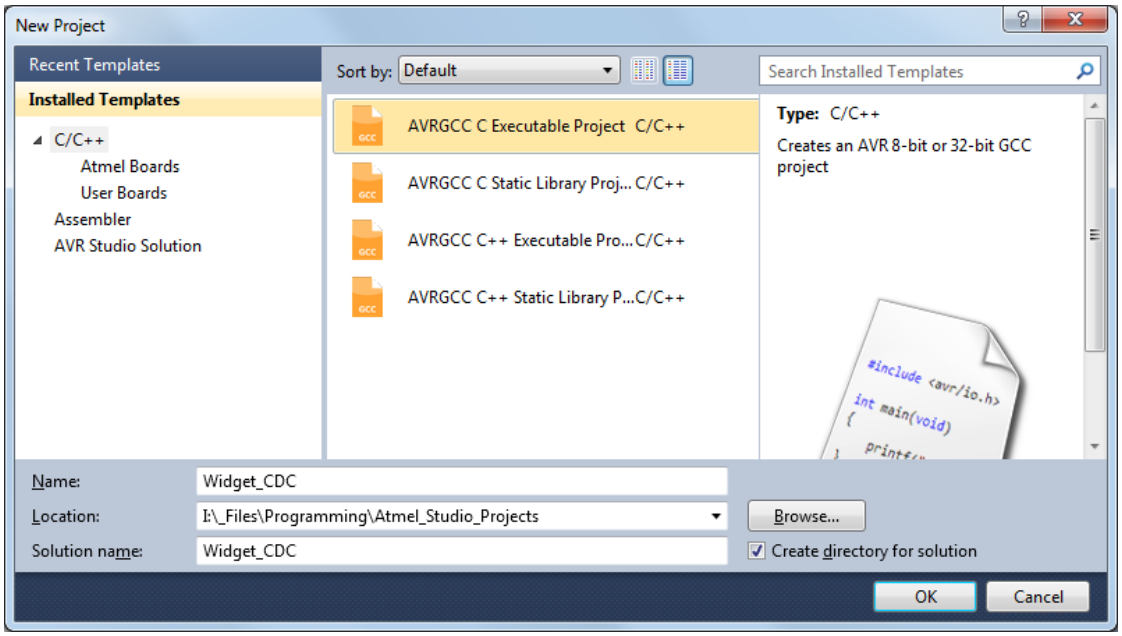

Figure 1: New Project Pop-Up Window.

A pop-up window appears for Device Selection. Select a microcontroller (this guide uses the AT90USB162) and click OK.

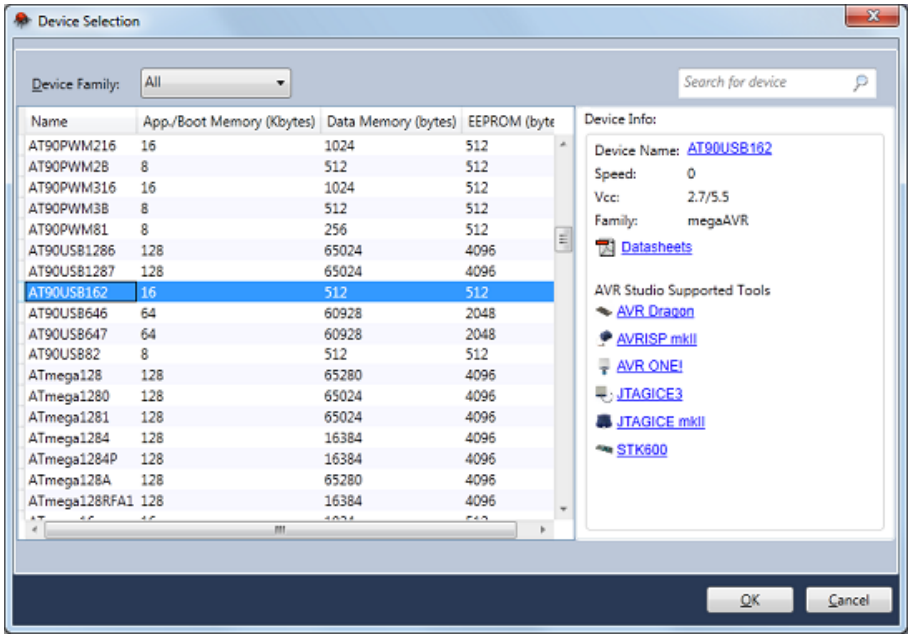

Figure 2: Device Selection Pop-Up Window.

Figure 3 shows the Studio5 Solution Explorer for the New Project (Widget_CDC).

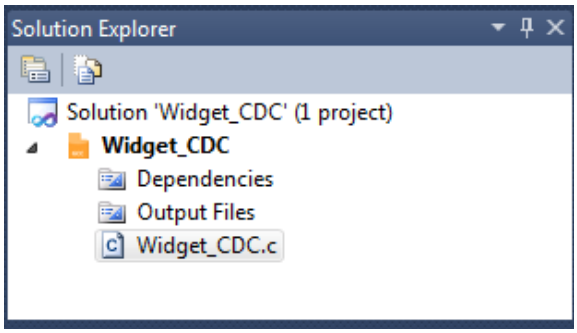

Figure 3: Initial Files in Solution Explorer

Step 2: Add Files to the Project

Using Windows Explorer, copy the folder USB_Stuff and the makefile to the project folder. These files are part of the downloaded zip file.

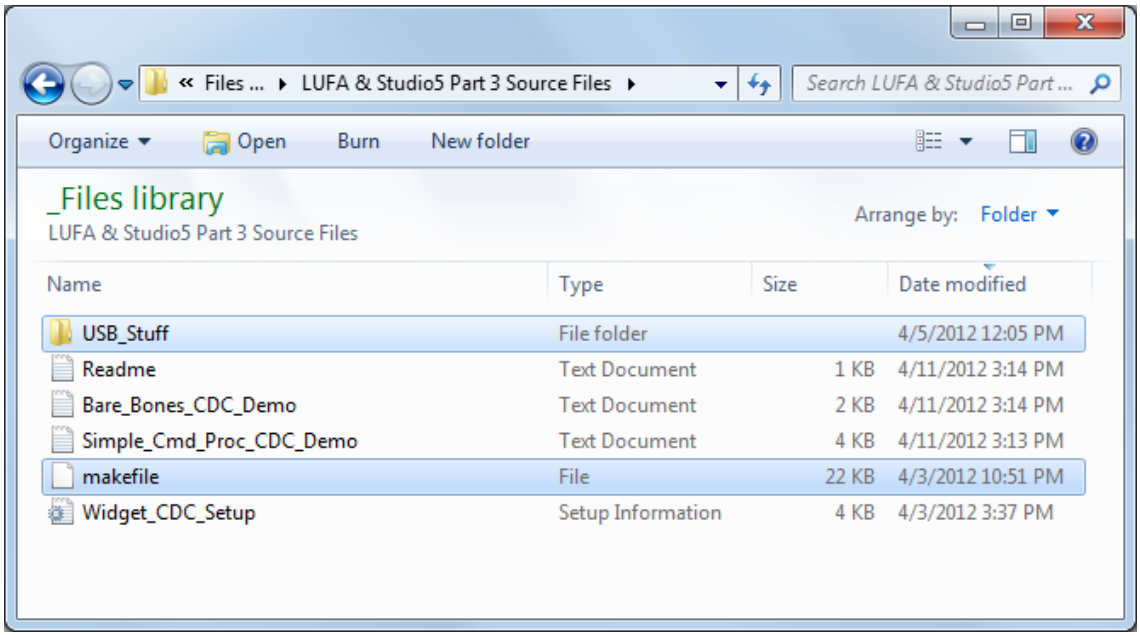

Figure 4: Copy the folder USB_Stuff and the makefile from the downloaded zip file to the project folder.

In Studio5, select Project > Show All Files, as shown in figure 5.

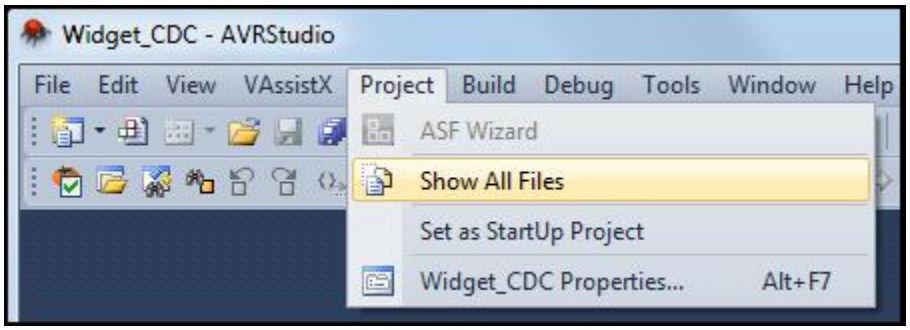

Figure 5: Select "Show All Files"

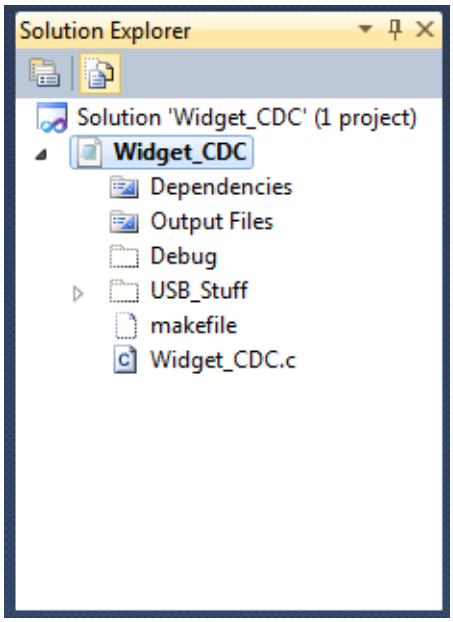

Figure 6: Solution Explorer showing all the files in the project folder.

In Studio5 Solution Explorer, right click on the folder USB_Stuff, and select “Include In Project”.

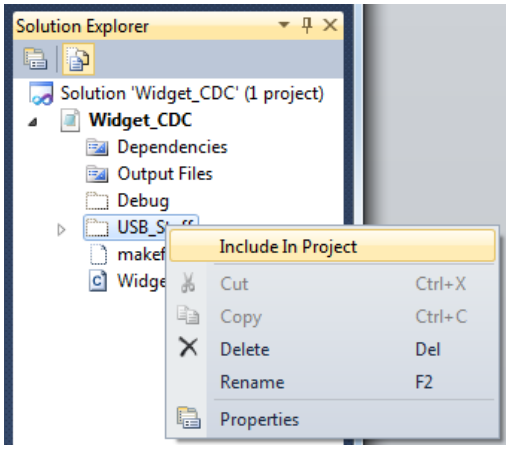

Figure 7: Include folder USB_Stuff in project.

When you include the folder, all the files in the folder are added to the project.

Now, right click makefile, and select “Include In Project”.

(Adding the makefile to the project allows you to edit the makefile from within Studio5.)

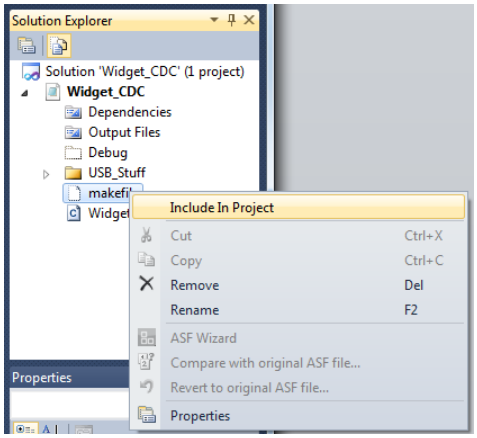

Figure 8: Include makefile to the project.

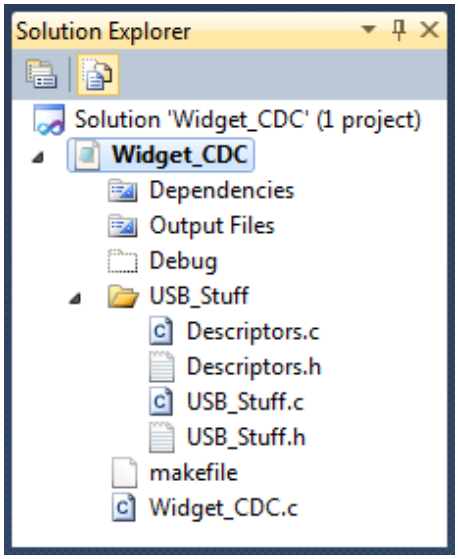

Figure 9: Solution Explorer showing that the files were added to the project.

Step 3: Modify Widget_CDC.c

```
/*
 * Widget_CDC.c
 *
 * Created: 4/3/2012 10:12:40 PM
 *  Author: C
 */

#include <avr/io.h>

int main(void)
{
    while(1)
    {
        //TODO:: Please write your application code
    }
}
```

Figure 10: Original Contents of Wiget_CDC.c

The downloaded zip file has two files, Simple_Cmd_Proc_CDC_Demo.txt and Bare_Bones_CDC_Demo.txt, that we'll copy into Widget_CDC.c (one at a time). The first includes a simple command processor that receives commands from, and sends responses to, the host. This will allow us to test the program using a terminal emulator on a PC, so we'll start with that one.

Use a text editor such as Notepad to copy the contents of Simple_Cmd_Proc_CDC_Demo.txt to the clipboard. Open Widget_CDC.c in Studio5 and delete all the contents. Paste the contents of the clipboard into Widget_CDC.c. Save the file.

Step 4: Modify Project Properties

In Solution Explorer, right click Widget_CDC and click on Properties.

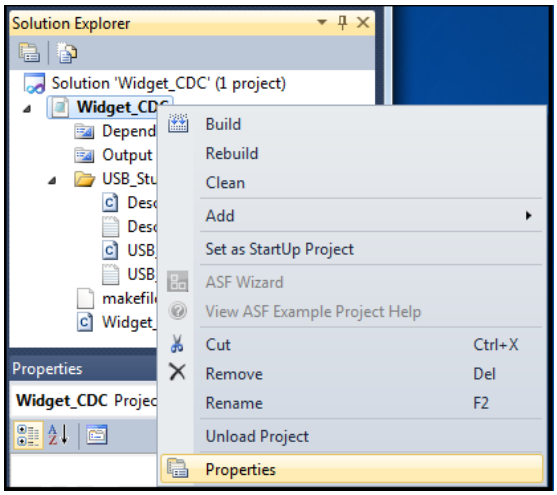

Figure 11: Call up the Project Properties Window

In the Project Properties window, click on “Use External Makefile” checkbox, and then on “Browse”. Select the makefile in the Widget_CDC project directory.

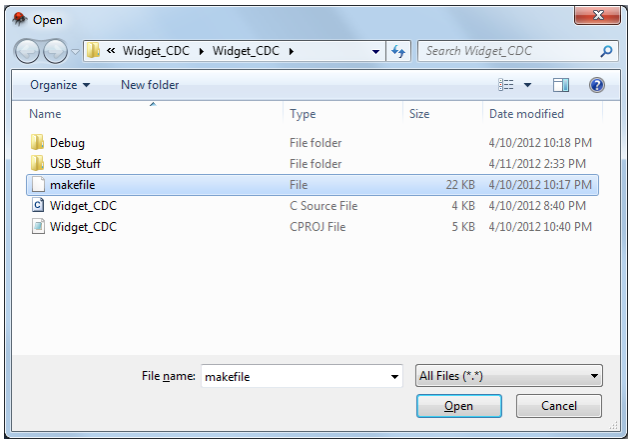

Figure 12: Select the makefile in the Widget_CDC Project Directory

Figure 13 shows the Use External Makefile selected and the makefile location specified.

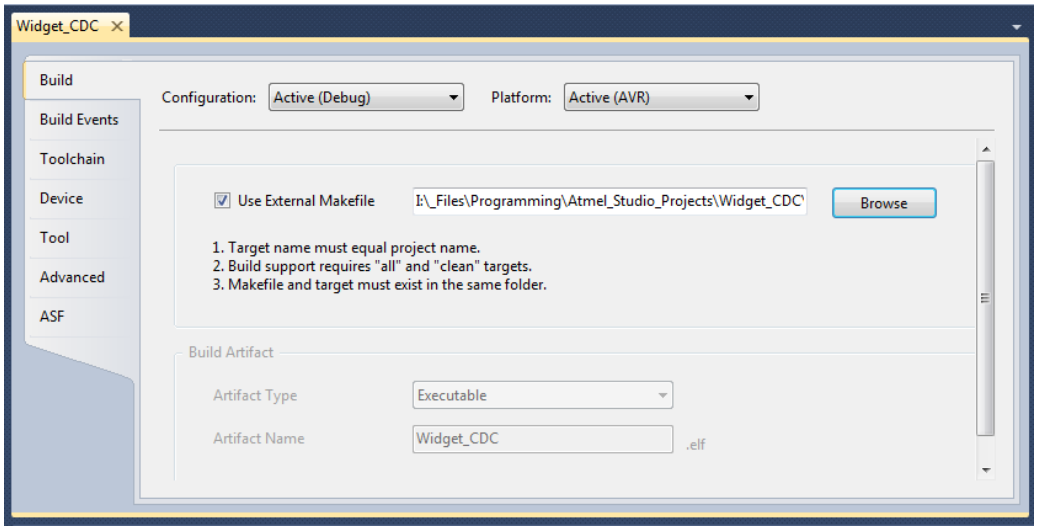

Figure 13: Project Properties showing the makefile in the project folder is now specified.

Step 4: Modify the Makefile

In Studio5 Solution Explorer, double click on the makefile to open it in the editing window.

Figure 14 shows the various items in the makefile that need to be specified.

```
MCU = at90usb162

ARCH = AVR8

BOARD = NONE

F_CPU = 8000000

F_USB = $(F_CPU)

TARGET = Widget_CDC

LUFA_PATH=../../../Atmel_Studio_LIbraries/LUFA/LUFA-120219/LUFA-120219

include $(LUFA_PATH)/LUFA/makefile

SRC = $(TARGET).c                                                    \
      USB_Stuff/Descriptors.c                                        \
      USB_Stuff/USB_Stuff.c                                          \
      $(LUFA_SRC_USB)                                                \
      $(LUFA_SRC_USBCLASS)
```

Figure 14: Items that need to be specified in the makefile.

TARGET =

This specifies the name of the main file of the project, which has the same name as the project.

LUFA_PATH=

To check LUFA_PATH, we need the directory structure for the project makefile and the library makefile.

```
I:\_Files\Programming\Atmel_Studio_Projects\Widget_CDC\Widget_CDC
         makefile      (in project folder)


I:\_Files\Programming\Atmel_Studio_Libraries\LUFA\LUFA-120219\LUFA-120219
         LUFA\makefile (in LUFA library)
```

Figure 15: Directory structure for the project makefile and the library makefile.

As can be seen, the LUFA_PATH as used in Part 2 will work for Part3 for this computer. You need to check if it will work on your computer and change as needed. (See Part 1 of this guide for a detailed explanation.)

SRC =

Note that we need to specify all the source files to the compiler, including any that you add to the project. Be sure to include `Descriptors.c and USB_Stuff.c with the USB_Stuff folder specified.`

Save the makefile and close the window.

Save All (In Studio5 select: File > Save All).

Step 5: Build the Project

Clean before building.

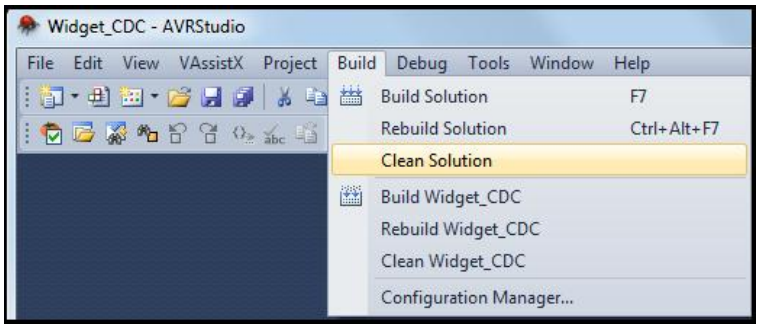

Figure 16: Clean Solution.

Now Build.

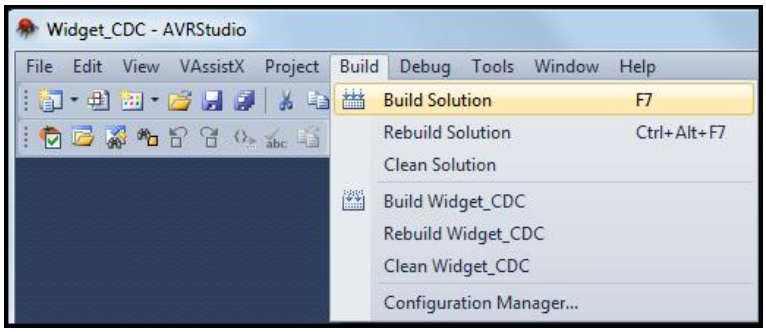

Figure 17: Build Solution.

Build succeeded.

```
Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Figure 18: Shows that the build process succeeded.

Check the memory usage, as shown in the Studio5 Output pane.

```
AVR Memory Usage
----------------
Device: at90usb162
Program:    3894 bytes (23.8% Full)
(.text + .data + .bootloader)
Data:        158 bytes (30.9% Full)
(.data + .bss + .noinit)
-------- end --------
```

Figure 19: Memory usage with the command processor demo in Widget_CDC.c.

As can be seen in figure 19, for a virtual serial connection the LUFA library has a light footprint.
Even a small uC such as the AT90USB162 has plenty of Flash and RAM left for the application program.

Step 6: Test the Program

Program the uC using the .hex file generated in step 5.

Verify that the device made a proper connection to the PC.

In Widows 7, use Control Panel > Hardware and Sound > Device and Printers > Device Manager > Ports (COM & LPT)

You may need to disconnect and reconnect the USB cable from the board you are using.

Figure 20 shows the commands and responses, indicating that the firmware is indeed able to communicate using USB CDC - the simple command processor used here is a tad contrarian.

(Bonus points to the first person to identify the Star Trek episode referenced by the 4th command-response.)

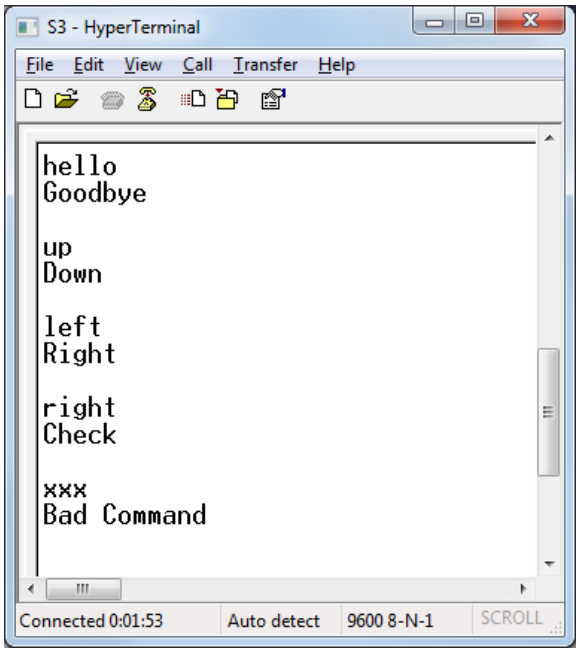

Figure 20: Commands and responses from the demo program.

The Bare_Bones_CDC_Demo.txt file in the downloaded zip file contains the same routines as Simple_Cmd_Proc_CDC_Demo.txt but with the functions related to receiving and processing the commands removed. You can go ahead and test it if you want. It will allow the terminal program to connect but it won't do anything. It's really just a shell waiting to be filled in.

Onward and Upward

This concludes this three part guide. Where you take it from here is up to you. The sky's the limit.