CSE466 Blimp Project – Phase One

AUTUMN 2011

OBJECTIVES

In this lab you will DO the following:

- Develop an interface for wireless control of the robotic blimp

In this lab you will LEARN the following:

- How to produce a custom SimpliciTI application
- How to use the USCI peripheral in UART mode
- The inner workings of the various blimp subsystems

DELIVERABLES

At the beginning of the next lab period you will demo your remote control system and any sensors and peripherals you were able to get working**. This demo will not be graded; the purpose of the demo will be to check in with the TA and keep your project on track.**

RESOURCES

These documents and web resources will be useful in completion of the lab and/or in answering the questions posed. Additional resources are described in the procedures below.

- BlimpBot schematics and component datasheets on course website
- MSP430 Datasheets from TI.com
- MSP430 Code Examples
- MSP430 Software Coding Techniques
- Code Composer Wiki
- MSP430 Optimizing C/C++ Compiler User's Guide

IMPORTANT NOTES

There are several very important rules to follow when using the blimp hardware. If you do not follow these rules, there's a good chance you'll release the magic smoke and have to wait to have your hardware repaired.

1. **ALWAYS POWER ON THE BLIMP BATTERY SWITCH BEFORE CONNECTING THE DEBUGGER**. Never connect the debugger when the battery switch is OFF or when the battery is disconnected or dead.
2. **NEVER short the motor terminals.** The motor driver ICs and battery are very good at sourcing HUGE currents, and things will be damaged.
3. **ALWAYS plug in the battery with the proper polarity.** There is no protection against a reversed battery!
4. **ALWAYS keep the battery voltage above 3.4V.** When you notice the regulated voltage on VDD starting to drop below 3.3 V, it is time to charge the battery (HINT: Your code should watch for this "drop out" condition...)

PART 1: WIRELESS INTERFACE USING SIMPLICITI

In this section you will use the "commission" functionality of the SimpliciTI API to produce a static link between two SimpliciTI end devices. Your two eZ430-RF2500 development boards will be these two devices.

The link procedure found in a typical SimpliciTI application (e.g., the Chronos watch and USB transceiver) is not based on the device address of the nodes being linked but instead is based only on which nodes are currently in "Link" or "Link Listen" mode. A button push on the receiver followed by one on the transmitter causes the two devices to link together. This is a nice method for sensor networks, but there is an obvious downside to this method in cases where the link procedure needs to be more selective. Accidental unwanted links are very possible using this method, which is a serious concern/annoyance when the link is used to pass control information to a robot (your blimp). Instead of using this push-button linking method, we will hard-code the link procedure in our SimpliciTI application using SMPL_Commission().

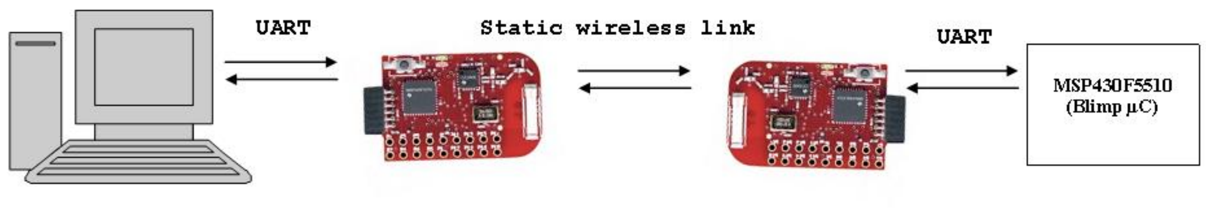
PROCEDURE:

1. Copy the entire SimpliciTI-CCS folder (from "C:\Texas Instruments\") to your working directory. If you can't find the SimpliciTI files installed on your system, download and run the latest installer from: http://www.ti.com/tool/simpliciti#description
2. In your personal copy of the SimpliciTI folder, open the Documents subfolder and follow the "SimpliciTI Sample Application User's Guide" section 2.3.2, which describes how to configure the "Simple Peer-to-Peer" demo application in CCS.
3. Program one of your RF2500 target boards with the LinkListen code, and the other with the LinkTo code. Ensure that they reliably link and communicate using the unmodified demo application.
4. Open the "SimpliciTI API" from the Documents folder and thoroughly read the section about the SMPL_Commission() API.
5. Modify the LinkListen and LinkTo applications to add selectivity to the link procedure by using the SMPL_Commission() function call. Remove unnecessary code (such as delay loops) from the demo application. You may want to rename the project, build configurations, and application source files to reflect the new nature of the two devices.
6. Verify that the link is working by remotely commanding the onboard LED to blink.

PART 2: UART RELAY

The recommended method of implementing a wireless interface to the blimp is by using the eZ430-RF2500 boards as a sort of UART relay. In this way, a wireless command-line interface can be developed to remotely control the blimp system.

The USCI peripheral supports asynchronous serial communication. You must become familiar with the USCI peripheral on both the MSP430F2274 (used on the RF2500) and the MSP430F5510 (used on the blimp). These two devices may differ slightly in the configuration details.



GOALS:

1) Implement a bi-directional UART relay, to allow for communications from the blimp to computer and vice versa.

2) Develop a command-line interface (CLI) on the MSP430F5510 which presents a set of commands for controlling the features of the blimp. The CLI should ultimately be wirelessly ported to a PC-side terminal application, but for testing purposes you may use a local UART connection.

   The following features of the blimp should be supported by your CLI. Implementation and syntax details are up to you.

   a. Control motors
   b. Read IR proximity sensor(s)
   c. Read E-field sensor(s)
   d. Read compass
   e. Read battery state (check MSP supply voltage, if below 3.3V then battery is "dead")

   NOTE: You do NOT need to implement these features at this point, but your CLI should call skeleton functions which will eventually be responsible for handling this functionality.