CSE466 LAB 3 – Electric field sensing, Phase 2

AUTUMN 2011

OBJECTIVES

In this lab you will DO the following:

- Develop a software UART implementation
- Develop a multichannel electric field sensor

In this lab you will LEARN the following:

- Asynchronous serial communication methods
- How to use the ADC10 peripheral
- Basics of electric field sensing
- Code division multiplexing methods

DELIVERABLES

Lab 3 is split into two phases spanning two lab weeks. The deliverables for each phase are due at the beginning of the lab period in the specified week.

PHASE 1, DUE OCT 25TH

- Demo of single-channel electric field sensor.
- Demo of UART transmit functionality.

PHASE 2, DUE NOV 1ST

- **Demo of multiple channel electric field sensing, comparing CDM & TDM techniques for channel separation.**
- **Demo of PC-side GUI for your multichannel electric field sensor.**
- **A PDF containing answers to all questions posed in the phase 1 and phase 2 lab prompts.**
- **Your fully commented and neatly presented code for phase 1 and 2, in a zipped format.**

RESOURCES

These documents and web resources will be useful in completion of the lab and/or in answering the questions posed.

- MSP430F2012 Datasheet
- MSP430F20XX Family User's Guide
- MSP430 Code Examples
- MSP430 Software Coding Techniques
- MSP430 Optimizing C/C++ Compiler User's Guide

PART 3 (PHASE 2): MULTICHANNEL E-FIELD SENSING AND TDM/CDM

In this section you will add an additional transmitter channel to the electric field sensing system for a more versatile sensing device. A total of two (2) sensing channels are required.

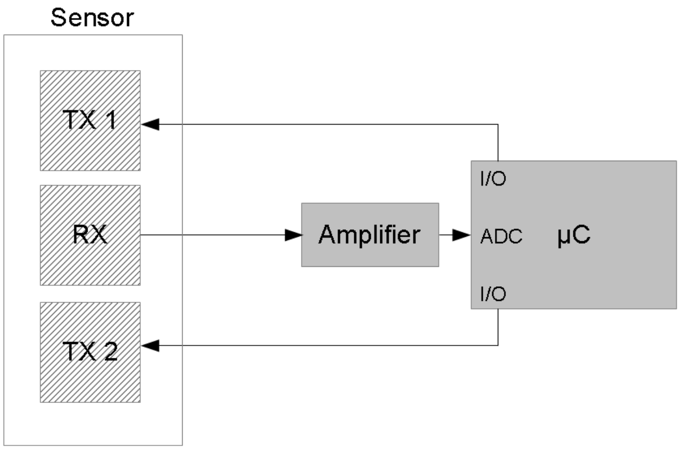

Figure 1. Two-channel E-field sensor overview

MULTIPLEXING

With multiple transmitters, a method must be developed to isolate, or multiplex, each sensing channel. You will implement your multichannel sensor using two methods for channel isolation, and will compare the effectiveness of each of these methods.

In both of the methods described, extracting the measurement value for a particular sensor channel is done by taking the inner product (dot product) of the received signal vector with the transmitted signal vector:

$$\text{Measured value} = \vec{S} \cdot \overrightarrow{P_C} = \sum_{t=0}^{N-1} S[t]P_C[t]$$

Where ***S*** is the received signal
P

$\boldsymbol{P_C}$ is the transmitted signal for a given channel ***C***
N is the number of samples collected (length of ***P*** and ***S***)

TDM

The first method you will use to implement this channel isolation will be a simple technique known as Time Division Multiplexing (TDM). In TDM, the transmitters are operated sequentially and separated into time slots. The signals transmitted are completely orthogonal, as only one signal is nonzero at any given time.

To use the TDM method, simply poll one sensor channel at a time using the method in Part 2. As described in lecture, **this is equivalent to performing the inner-product calculation described above.**

CDM

A second method is the use of Code Division Multiplexing (CDM), in which all transmitters are simultaneously operating but each transmitting a unique pseudorandom pulse train. Over a sufficiently large number of samples, these pseudorandom transmitted signals will be nearly orthogonal. Performing the inner-product calculation described above will allow extraction of each sensor channel value. **In the inner-product calculation, a transmitted LO should be represented as -1, and a transmitted HI should be represented as (+)1.**

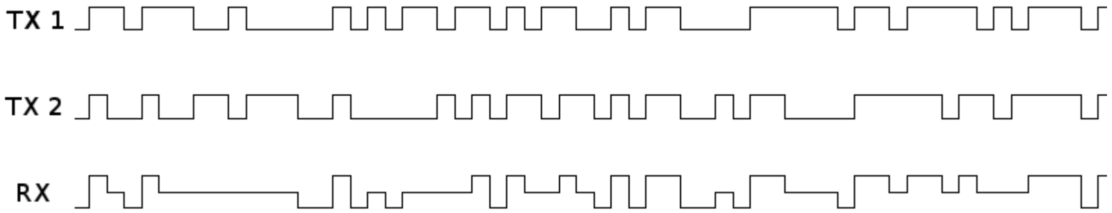

Figure 2. The received signal is a linear combination of the two transmitted signals. (The actual received signal is the derivative of the RX signal shown.)

In order to produce these pseudorandom signals, a linear feedback shift register (LFSR) must be implemented in software. One particular bit of the LFSR should be used to generate the transmitted pulsetrain. An 8-bit LFSR, shown below, with taps a bits 3,4,5, and 7, is recommended for this application. Observe that the LFSR will "stick" when in the 0x00 state, and so must be initialized to a non-zero state. You will need separate LFSRs, each initialized to a different state, for each channel.

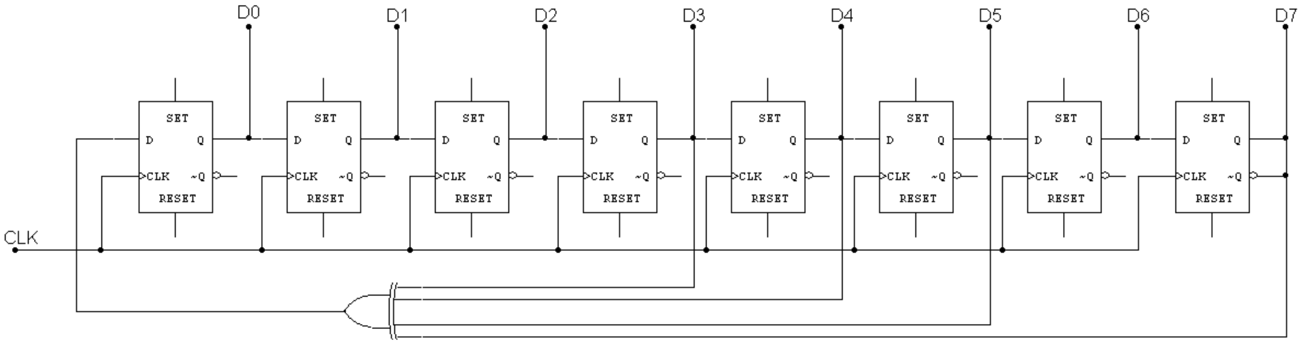

Figure 3. 8 bit maximal length sequence Fibonacci LFSR with taps at bits 7,5,4, and 3.

CHARACTERIZING EFFECTIVENESS

The contrast-to-noise ratio (CNR) will be an indicator of the effectiveness of channel isolation and noise rejection for the TDM and CDM channel isolation methods. The following steps should be taken to evaluate the CNR of this sensor given a particular multiplexing method (CDM/TDM) and signal vector length (number of samples per measurement). **Use MATLAB or MS Excel.** To make a fair comparison, give the two methods the same amount of real time. For two channels, give the CDM method twice as many samples as the TDM method.

1) Disable the IIR low-pass filter and pass accumulated sensor values directly to the PC.
2) Collect a large number of measurements (1000+) at the maximum sensor value (No hand present)
3) Collect a large number of measurements (1000+) at the minimum sensor value (Hand as close as possible to sensor)
4) Find the mean of the values produced in steps (2) and (3)
5) Find the std. deviation of the values produced in step (2)
6) $\text{CNR} = \dfrac{mean\ of\ step\ 2\ values - mean\ of\ step\ 3\ values}{standard\ deviation\ of\ step\ 2\ values}$

Question 3.1: What was the CNR for the TDM and CDM methods?

Question 3.2: How many samples per second were you able to process with the two techniques?

Question 3.3: How did the CNR vary given different signal vector lengths (number of samples per measurement)? Provide at least two additional data points for both TDM and CDM cases.

PART 4 (PHASE 2): PC-SIDE GRAPHICAL USER INTERFACE

A PC-side application, written in your language of choice, will be developed to interface with the MSP430 via your serial communication link and will visually display the data in a meaningful format. Java (using RXTX serial library), Python (using pySerial) and National Instruments' LabVIEW (using NI VISA library) are some possible environments for developing this GUI.

The requirements for this GUI are that it has some form of **graphical** display (not just numerical or textual) which somehow represents each sensor value.

HINTS:

- When using the GUI, you should send raw data over the UART instead of user-readable formatted text. This should make data parsing simpler on the receive end.
- Beware of framing errors, which occur when the PC's UART receiver gets "out-of-sync" with your UART transmitter.

Question 4.1: What programming language/environment did you use to develop the GUI? What were some advantages and disadvantages of this environment for this assignment?

Question 4.2: If you implemented an extra credit feature in this lab (Phase 1 or 2), describe it here.

EXTRA CREDIT SUGGESTIONS

For an additional challenge and some extra credit, increase the number of transmitters to three or four. To receive credit for the extra sensing channel(s), the GUI you develop in Part 4 must be capable of displaying the values produced.

Full credit for the GUI will be given for a graphical display as simple as a bar graph or meter, but extra credit points can be obtained if this minimum is exceeded. For instance, your graphical display could be used to represent the approximate location of the hand over the sensor plate.

Alternatively, implementing a PC -> MSP UART link for control over the sensor's operating parameters (such as data sampling size and low-pass filter coefficient) will also earn extra credit points.

Implement frequency division multiplexing (FDM).

Implement the "Galois" LFSR configuration instead of the "Fibonacci" configuration shown above. How many samples per second can you process using the alternative configuration? Does this affect the CNR you can achieve in a fixed time period? See http://en.wikipedia.org/wiki/Linear_feedback_shift_register for the Galois form of the LFSR.

If you have another idea for an extra feature or "selling point" of your electric field sensor, please propose this idea to the TA before implementing it to verify that you will receive credit.