Instrumentation and Algorithms for Electrostatic Inverse Problems

by

John Paul Strachan

Submitted to the Department of Physics

and

Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Physics

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2001

Author ....

Department of Physics

and

Department of Electrical Engineering and Computer Science

Aug 31, 2001

Certified by..

Neil A. Gershenfeld

Associate Professor

Thesis Supervisor

Accepted by ....

Arthur C. Smith

Chairman, Department Committee on Graduate Students

Instrumentation and Algorithms for Electrostatic Inverse Problems

by

John Paul Strachan

Submitted to the Department of Physics
and
Department of Electrical Engineering and Computer Science
on Aug 31, 2001, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Physics
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes tracking objects with low-level electric fields. A physical model is presented that describes the important interactions and the required mathematical inversions. Sophisticated hardware used to perform the measurements is described in detail. Finally, a discussion of the myriad applications for electric field sensing is described. The main application goal for this thesis is to make an efficient 3D mouse using electric field sensing technology.

Thesis Supervisor: Neil A. Gershenfeld
Title: Associate Professor

Acknowledgments

With immense gratitude, I would like to thank my advisor Neil Gershenfeld for finding a place for me in his group and keeping me on my toes with his restless intellect. I also thank Josh Smith for doing an amazing job in laying the groundwork for electric field sensing. Without his effort I would still be stumbling in the dark.

I would also like to thank the amazing students of the Physics and Media group. In particular, Rehmi Post and Matt Reynolds, who, between the two of them, patiently taught me everything I currently know about electronics and hardware development. And Jason Taylor, who lent his hands, voice, or wisdom to practically all aspects of this work. And Ben Recht, who provided steadfast mathematical and friendly companionship. I also wish to thank Derik Pridmore for helping me machine some of the crucial equipment I used, and Peter Russo for his work on resistive sheets.

In the keeping-me-sane category, I cannot fail to mention my great friends Ash Ahluwalia, Emily Chang, Danny Lai, Amy Lee, Spencer Liang, Ryota Matsuura, Aditya Prabhakar, Bhuwan Singh, and Tina Tyan. These wacky people frequently cause me to redefine my concept of appropriate social behavior.

Last, but certainly not least, I would like to thank my parents John and Damaris, and my sisters Monica and Erica, who have magnanimously tolerated my years of missing birthdays and spending winter breaks recharging on sleep. It is no understatement that I owe everything to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Purpose and Background

A critical component of any intelligent environment is the user interface. As computing becomes more and more pervasive in people's lives, we wish to make this interface as natural and efficient as possible. In this case, making it more natural implies a desire for human-computer interaction to mimic human-human interaction. For this to occur, the computing system needs more detailed information about the user's gestures and movements. Electric field sensing offers the possibility of supplying this type of information in a quick, inexpensive, and unobtrusive way.

The ultimate goal of electric field sensing is tomographic imaging of the local environment. This is a very difficult and ill-posed problem. Other forms of tomography, such as CAT (Computer-Aided Tomography) scans depend on a linear response from the material which is imaged, such as the absorption function when light is shined on the imaged body. More carefully, if the response of only object X is measured, as well as the response of only object Y, then the response with both X and Y is simply the superposition of the two responses. This is not the case for electric field sensing. When one "shines" electric fields at an object, the measured response (induced charge) cannot be superposed with other objects. Furthermore, the response itself is non-linear since position enters as a $\frac{1}{r^2}$. Maxwell's equations are linear only in source charges, not in any geometric parameters such as position or boundary surfaces. Thus, we have a doubly non-linear problem: we cannot superpose the responses of multiple objects since we need to account for interaction between them, and the individual response of an object is non-linear in position. To make the problem more tractable, we need to impose constraints, such as assuming specific shapes for the objects. This turns it into a tracking problem, rather than an imaging problem.

1.2 History

Most of the pioneering work in electric field sensing can be attributed to nature. Electric fields are used by various aquatic animals for sensing their environment, especially in dark, muddy waters where light is scarce [Bas94]. Using amplitude modulation, the fish can communicate with each other as well as determine parameters of nearby objects such as size, shape, distance away, and velocity. In deference to their pioneering work we have named our family of electric field sensing hardware Fish. Also, since one of the main goals of our work is to make a 3D user interface, this serves as a nice contrast to mice, which move practically in 2 dimensions, while fish navigate in 3. Some members of this hardware family (which will be described in more detail shortly) are the Classic fish, Smartfish, Lazyfish, and Taufish.

Among homo sapiens, electric field sensing was first seen with Leon Theremin's musical instrument in the early 20th century. In this device, capacitance to the user's hand was measured by two antennas controlling pitch and amplitude of a sound synthesizer. More recently, Neil Gershenfeld [Ger91], [Ger93] used electric field sensing in a collaboration with Yo-Yo Ma to detect movement of a cello bow with respect to the cello body. The goal here was to separate the musical response of a cello (which can be synthesized on a computer) from its physical embodiment. Since then, the Physics and Media group of the MIT Media Lab has further developed and used the sensing technology.

Electric field sensing is also used in applications to geophysical prospecting for finding oil or minerals, as well as impedance imaging of the human body for medical applications [Smi99].

1.3 Instrumentation

In this thesis I describe our current state of hardware development for electric field sensing. As mentioned above, we have named our family of instrumentation Fish. Figure 1-1 shows the ancestry of the hardware. The "Small Box" is the original, hand-wired, all analog implementation made by Neil Gershenfeld. Classic Fish was designed and created by Joe Paradiso, Tom Zimmerman, and Josh Smith and included an 8 bit Motorola microcontroller. SmartFish attempted to do most of the signal processing in software, but never quite worked.

Finally, Lazyfish was designed and implemented by Josh Smith. Using a synchronous undersampling trick, the hardware was able to perform signal demodulation using only a PIC microcontroller with an on-board A/D converter and a few op-amps for gain. The board has a nice design and a compact footprint. As described in Chapter 3, I basically used this design with a few modifications for my measurements.

Another member of the Fish family is the Taufish, with implementation done by Rehmi Post and firmware written by me. This board has the smallest footprint of all the Fish, measuring 3cm x 5cm. In contrast to the previous hardware which work in the frequency domain, the Taufish performs a time-domain measurement of RC charging cycles. The board is described in more detail in Chapter 3.

1.4 Algorithms

The instrumentation sketched out above provides measurements of coupling between sense electrodes and the body to be tracked. These set of measurements are a function of the geometric parameters of the body (position, size, orientation, etc.). To perform the inversion and get geometric parameters from sensor readings requires some knowledge of this forward function. As mentioned earlier, this forward function is non-linear which means there does not exist a provably optimal algorithm for capturing the internal parameters.

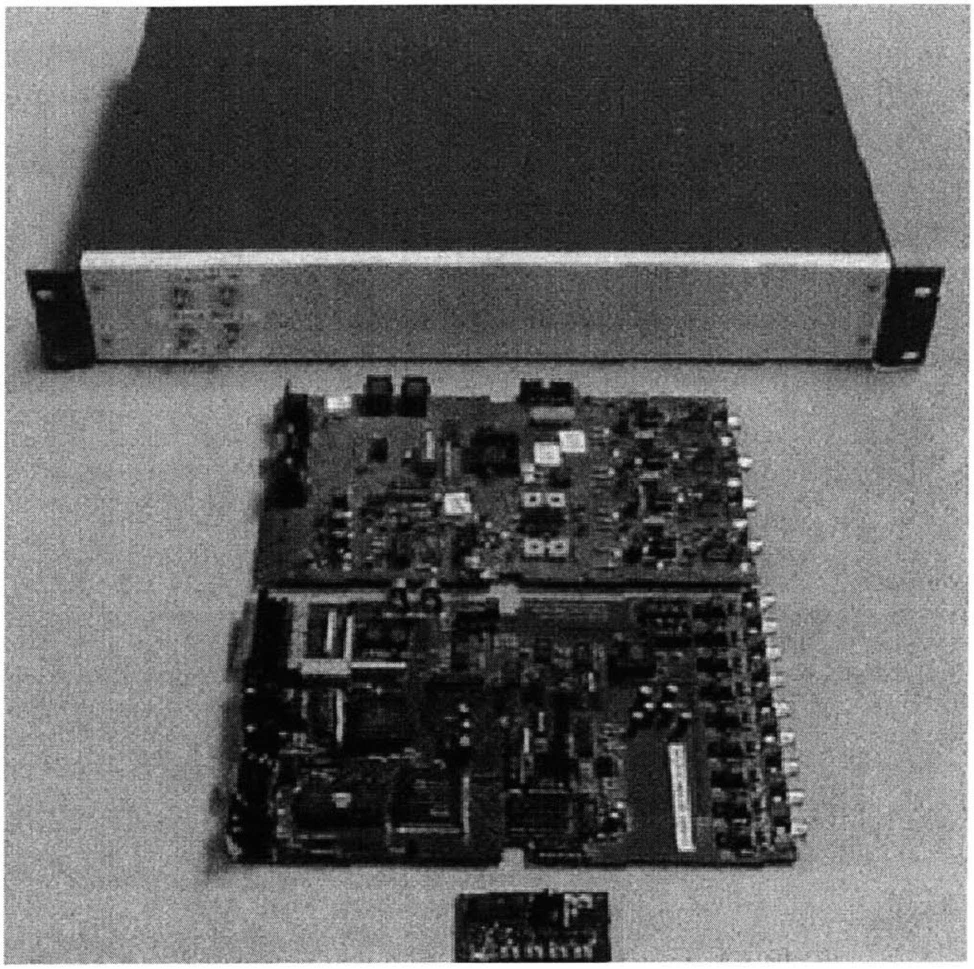

Figure 1-1: Evolution of Fish hardware. From top to bottom: "Small Box", Classic Fish, SmartFish, and LazyFish

Some of the algorithms discussed in this thesis include performing a local-linearization of our forward model and using standard, optimal (in the least-squares sense) linear techniques such as Kalman Filtering. This algorithm nicely allows for inverting noisy measurements using a noisy (i.e. possibly inaccurate) forward model, given that one has some prior knowledge of the internal dynamics of the system.

Another technique used in this thesis is decomposing the imaged object into its induced charges. The forward function in this parameterization is then linear and more tractable. Unfortunatley, arriving at and using the exact analytic form for the forward function is a challenging and computationally expensive problem. I describe a technique for finding a Green's function for the response of a point source. More exotic charge distributions can then be composed from this function.

Chapter 2

Physical Mechanisms

2.1 Introduction and Motivation

As with any kind of imaging problem, the information we really seek is the arrangement and density of the matter in our volume of interest. However, if we are to perform this imaging with electric fields, we are only "looking at" the world of charges, potentials, conductors, insulators, etc. In some sense this limits the amount of information we ultimately have of the actual configuration of mass in our system. It certainly suggests the need for a layer to reside between electric field sensing and actual tomography, to serve as an interpreter for the different language in which each speaks. On the other hand, this gets to the heart of electric field sensing's effectiveness: a freedom from many of the limitations present when one deals with masses. Looking at other forms of imaging such as video cameras, IR, and sonar reveals some of these constraints. Many of these techniques require an uninterrupted line of sight between the body and sensors, possess a sensitivity to surface textures and orientation, and require large information bandwidth and computation power to process. In the case of sensing with electric fields, the measurements are unaffected by dense or foggy air, clothing, makeup, sunlight, and other environmental background noise.

The bodies we wish to model and the bodies we wish to ignore conveniently polarize into materials of widely differing electrical properties, such as good conductors and good insulators. The computation required to process the measurements will be discussed later, but a useful observation is that to a large extent, the computation is proportional to the amount of information one wishes to extract out. This is a useful property of any general imaging scheme which allows implementations spanning the space from simple low-resolution feature recognition to more detailed imaging. An example is Josh Smith's work with "meta-balls" and image spheres [Smi99] in which a simple model (spheres) is used for the imaging body. This drops the problem into a much lower parameter space and thus simplifies computation.

Once again, electric field sensing concerns itself with measuring the electrical properties of the environment. The goal of this chapter is to provide a physical model for these measurements and their relation to the configuration and materials properties of the imaging bodies.

First off, any description of our physical theory must start with Maxwell's equations. Here they are for reference:

$$\nabla \times \mathbf{E} = \frac{\partial \mathbf{B}}{\partial t} \tag{2.1}$$

$$\nabla \times \mathbf{H} = \mathbf{J_{free}} + \frac{\partial \mathbf{D}}{\partial t} \tag{2.2}$$

$$\nabla \cdot \mathbf{D} = \rho_{free} \tag{2.3}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.4}$$

along with the continuity equation,

$$\nabla \cdot \mathbf{J_{free}} = -\frac{\partial \rho_{free}}{\partial t} \tag{2.5}$$

where, for isotropic and linear media,

$$\mathbf{D} = \epsilon \mathbf{E} \tag{2.6}$$

$$\mathbf{B} = \mu \mathbf{H} \tag{2.7}$$

$$\mathbf{J_{free}} = \sigma \mathbf{E} \tag{2.8}$$

2.2 Capacitive Measurements

Now, given the frequencies used in this project ($\leq 1MHz$) and the characteristic length scales ($\tilde{1}$ m), we can make a quasi-static approximation and use the familiar equations of electrostatics. More rigorously, this requires expanding our fields about a time rate parameter $\tau = \alpha t$ and taking just the first order terms (see, for example, [RFA60]).

For our applications, we can view the environment surrounding the sensing electrodes as a region of isotropic but inhomogeneous material. The basic configuration of our measurements is shown in Figure 2-1. Here we see all the couplings between the body to be imaged, two sense electrodes (although many more are typically used), and the ground which surrounds them. We can perform measurements at any sense electrode. These can be loading measurements of coupling to the outside world or direct measurements between two electrodes where one acts as a transmitter and the other acts as a receiver. More will be discussed on the different types of measurements. As shown, the basic interaction is of a capacitive nature, which stems from our quasi-static limit in which we can ignore electrodynamic interactions.

For the i^{th} electrode, held at potential V_i, the total charge induced on it by the other electrodes is:

$$Q_i = \sum_j^N C_{ij} V_j \tag{2.9}$$

(alternatively, we could say the N electrodes each have a total free charge, Q_i, and then find the potentials $V_j = \sum_i^N P_{ji} Q_i$. In this case, the matrix P is the inverse of the C matrix discussed below. Since we are applying voltages and measuring currents (charge), the C matrix is more relevant for our (and most other) applications.)

The C terms form a capacitance matrix where C_{ij} is the capacitance between electrodes i and j. C_{ij} can be found by holding all other electrodes at zero potential and taking the ratio of charge to potential difference between i and j. The diagonal elements of the capacitance matrix, C_{ii}, are the "self-capacitances" which is the ratio of the charge on the conductor, Q_i, to the potential at which it is held, V_i. Another

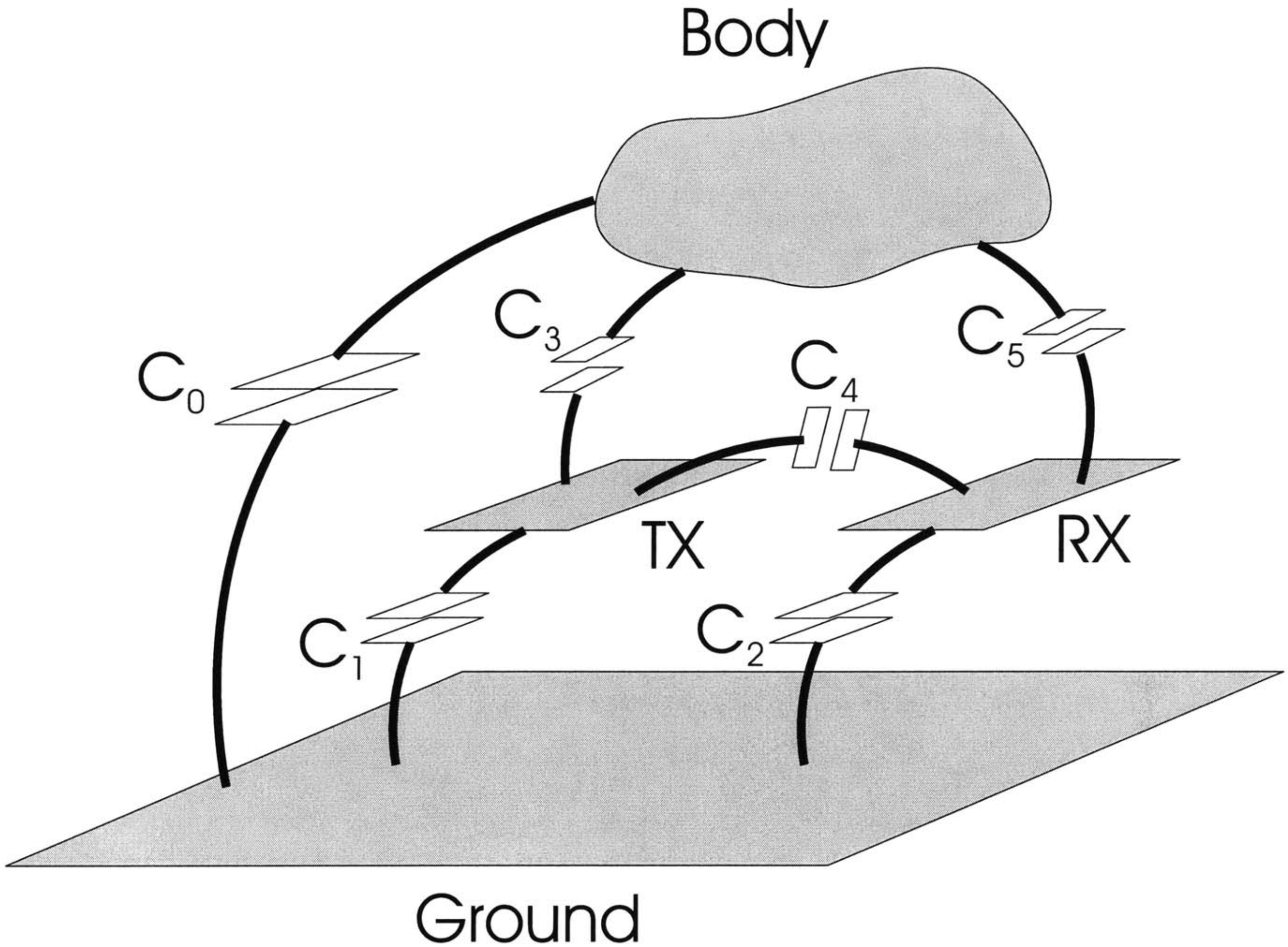

Figure 2-1: Configuration of sensors and imaging body used for electric field sensing

way of looking at self-capacitance is to assemble the free charge Q_i on the electrode by bringing it in from infinity (zero potential). In bringing in the charges, we work against the fields caused by all the other electrodes, and this defines our potential. Thus, the "self-capacitance" of a conductor is really a measurement of the fields due to everything besides that conductor. This defines how much work it takes to add charge to the electrode, or equivalently, the load one sees while driving the electrode.

Most previous forms of electric field sensing measure only the self-capacitance, thus working in "loading mode". In this case, a sense electrode is driven with a signal and the load it sees is measured. This load is dominated by the coupling to the immediate vicinity, e.g. a hand moving around above the electrode. In the case of loading mode measurements, one attains one data point per electrode per measurement. As pointed out by Joshua Smith [Smi96], we can extract more information by looking at the off-diagonal elements of the capacitance matrix.

2.3 Transmit-Receive

A loading mode measurement, in effect, measures an electrode's coupling to everything all at once. We would like to be able to isolate individual couplings and measure their value. This is complicated by the fact that we only have direct access to a few of the bodies involved. For example, we cannot directly measure the coupling between the body and the system ground. What we can do, however, is to drive one electrode at a particular frequency and look for signals of this frequency at other electrodes, thereby isolating from other couplings. This is simple to implement in hardware: one electrode is driven at a particular frequency, f, and the received signal at a different electrode is multiplied by the transmit signal, thus generating bands at frequencies 2f and 0. The product is then low-pass filtered and the resulting DC value is proportional to the amount of current traveling between the transmit and receive electrode.

There are further practical benefits of using AC measurements in electric field sensing. As long as our frequencies keep us in the quasi-static regime, our circuit model, Figure 2-1, still holds. Since the impedance between the bodies goes as $\frac{1}{j2\pi fC}$, we can decrease resistance by increasing frequency. This, however, does not change the relative values of our signals. All current pathways between our transmitter and receiver, including those not through the imaging body, are amplified equally, thus increasing the measured absolute signal. Practically, this is useful since it is easier in our hardware to measure a large signal swing. For example, at some point our signal is passed to an A/D converter. For a fixed number of bits, n, and an input voltage range, V, the resolution of our A/D is fixed at $\Delta V = \frac{V}{2^n}$. By increasing the absolute strength of our signals, we can resolve variations in this signal with better accuracy. And finally, using high frequency signals makes our measurements more robust to the pervasive 1/f noise.

The coupling one sees between a transmit-receive pair is a function of the imaging body. That is to say, there is a current pathway from the transmitter to receiver going through the body. This can be seen in what we call the "transmit" and "receive" measurement modes. The picture to keep in mind is that of Figure 2-1 where we only care about current paths between the transmitter and receiver–all others being eliminated since we measure synchronously. Thus, the appropriate model is that of a current divider consisting of those current paths directly between the transmit and receive electrodes, and those that travel through other bodies (including the one we are tracking). In the end we are only interested in C_3 and C_5 since this contains all geometric information regarding the imaging body.

2.3.1 Transmit Mode

As the body moves closer to the transmitter, the coupling shown as C_3 increases and the body becomes a large transmitter. This transmitted signal is measured at the receiver and varies with the distance between the body and the receiver. How does it vary? For very short distances away, we can approximate the body and receiver as a parallel plate capacitor and so the signal goes as 1/r. For large distances the body appears as a point source so the signal goes as $1/r^2$ as can be found using the method of images for a point charge above a ground plane.

2.3.2 Shunt Mode

Another interesting regime for electric field sensing is shunt mode. Once again we have a transmitter and receiver and a nearby body. As the body nears the electrodes, a portion of the displacement current between transmitter and receiver is "shunted" away to the body. This registers on the receiver as a decreased signal. When the body is out of view, the current flows between transmitter and receiver according to the coupling shown as C_4 and this registers as a maximal signal.

Shunt mode was the primary measuring mode used by Josh Smith in his work.

One of the big advantages of this mode is the amount of information that can be extracted. For a set of N electrodes we can get $\frac{N(N-1)}{2}$ data points. This is because any 2 electrodes can be considered a transmitter-receiver pair. The factor of $\frac{1}{2}$ stems from the indistinguishability of swapping the transmitter with the receiver. As suggested earlier, in loading mode only N data points could be extracted for the same geometry.

2.4 Comparison of various modes

The primary criteria by which to judge the different electric field sensing modes is: 1) amount of information per measurement, 2) sensitivity range, and 3) scalability. Other characteristics such as noise immunity and The first criterion we have already mentioned. Loading mode provides N measurements, shunt mode provides $\frac{N(N-1)}{2}$ measurements, and transmit mode provides RT measurements, where R is the number of receivers and T is the number of transmitters.

For sensitivity range, we are concerned with both signal resolution and signal isolation. That is to say, we care about our ability to resolve small changes in our signal (which corresponds to small movements in the environment) as well as the ability to block off unimportant portions of the environment. Clearly, in this latter criterion, transmit-receive mode is better suited.

For signal resolution we need to look at the functional fall off of the signal as the body moves away. Empirically speaking, we have found better sensitivity in transmit-receive measurements than in loading measurements. We even conjectured that this was due to a more rapid functional fall-off in loading mode. Figure 2-2 shows this not to be the case. The two sets of data were taken with the same geometric configuration for transmit-receive mode and loading mode. In the case of transmit-receive, the coupling between two bodies was measured as a function of separation. For loading mode, the same two bodies were used, with one body grounded and the other performing load measurements. To make the plots commensurate they were scaled and offset. It therefore seems likely that we get better sensitivity with transmit-receive measurements due to better isolation of the signals we are concerned with measuring. Loading mode sees small perturbations on a large signal while transmit-receive see much larger perturbations. Also, many of our loading mode hardware (such as taufish) use a DC measurement scheme rather than AC, thus diminishing the strength of capacitive coupling.

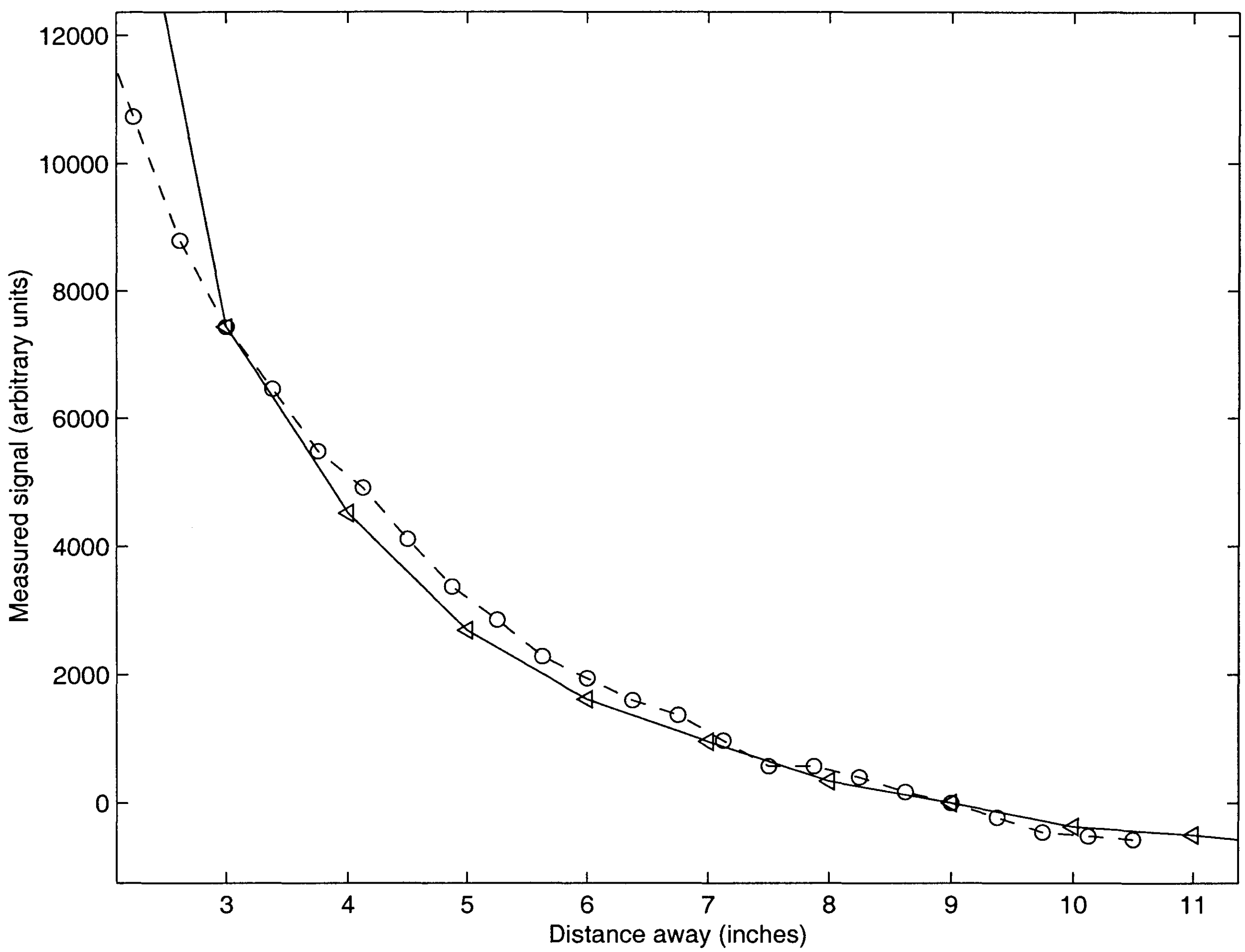

Figure 2-2: Plot of loading mode and transmit-receive mode fall-off with distance

Scalability issues become important as we try to extend our sense range to larger volumes or to greater resolution. Given the model I have provided, it is clear that in order to measure larger volumes with the same resolution, more sense electrodes will be required. Similarly, more sense electrodes are needed for measuring the same volume with greater resolution. We have hit these types of scaling limits in some of our past applications, most notably in the Museum of Modern Art exhibit. Here, we wished to measure areas of 0.3m x 0.3m with a resolution of 1cm. Accomplishing this required 100 electrodes of size 1cm x 1cm. The measurement modes discussed thus far all have the same horrible scaling performance. In order to scale well, we came up with a different measuring scheme using planar sheets of a fixed resistivity.

2.5 Resistive Sheets

The basic model for using resistive sheets is shown in Figure 2-3. Measurements are performed only on the perimeter of the sheet. The idea is that a capacitive coupling will exist between the sheet and imaging body, while resistive paths will exist within the sheet to the various sensor taps on the perimeter. Since the displacement current between the body and sheet will flow through the path of lower resistance, the taps closest in the x,y plane to the body will receive the largest signal. These taps are held at ground while the current sunk through them is measured. In essence, the capacitive coupling gives an estimate of the distance z between body and sheet, while the resistance in the sheet gives x and y.

Since one of our main goals is to make a 3D mouse operational in a large area, much of the work in this thesis involved using resistive sheets. These sheets were constructed by SCA out of cardboard and a coating of a duPont carbon film. A parameter which must be selected carefully is the resistivity of the sheet. We would like the currents in the sheet to be small enough as to make the sheet appear an equipotential for capacitive measurements. This points to using a large resistivity. However, as we scale up the resistivity in the sheet, it becomes more favorable for displacement current to travel directly from body to sensor taps, rather than going through the resistive sheets. This returns us to the standard geometry in which we have an array of isolated sense electrodes. Thus, we must walk a fine line between the regime in which the resistive sheet appears as one big electrode and the regime where it appears as an array of completely isolated electrodes.

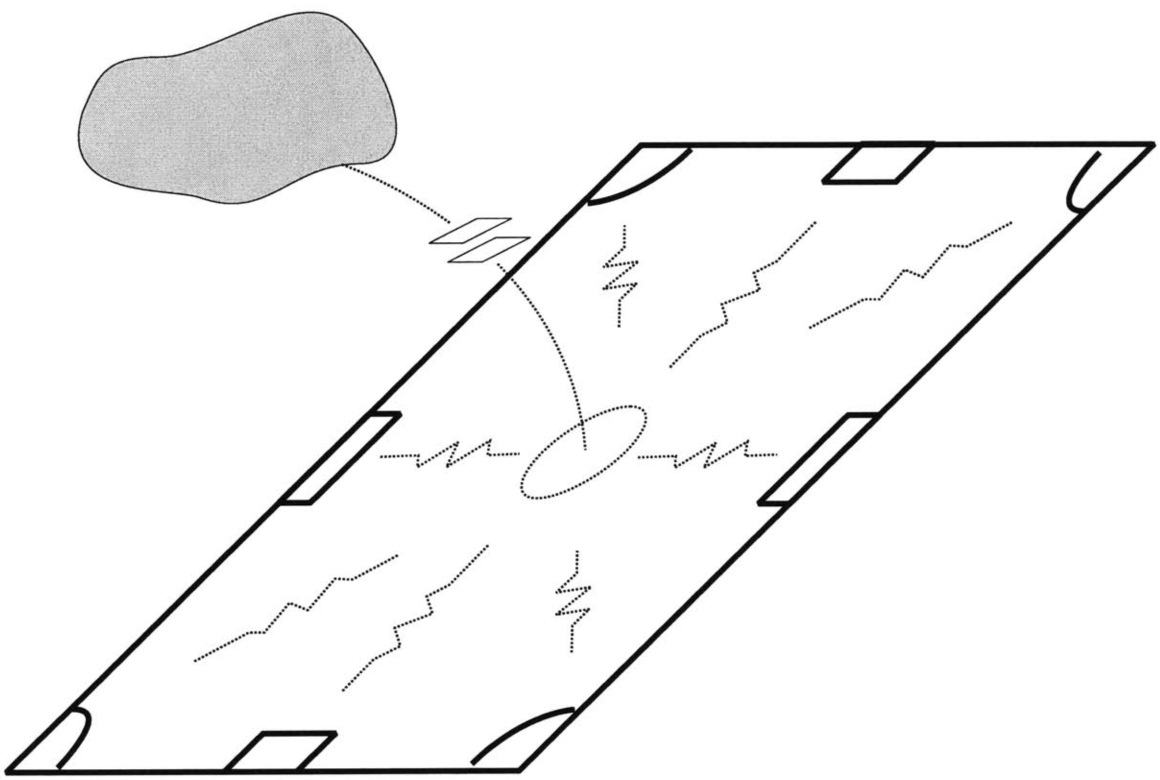

Figure 2-3: Circuit model of resistive sheet with 8 sensor taps on perimeter

Chapter 3

Measurement Hardware

Much of the work on electric field sensing has gone into the evolution of the measurement hardware. The Fish family has a long lineage which moves from complex, big, and do-all boards to simple, small, and do-one-thing-very-well boards. At the same time, we have explored using the various modes of electric field sensing (described earlier), and pushing the performance in that particular regime. The latest and greatest hardware is the Taufish, for loading mode measurements, and a revised LazyFish for transmit/receive measurements. Each will be described along with design figures and performance specifications. More details, including parts lists, can be found in the appendices.

3.1 Taufish

Taufish is designed around the idea that electric field sensing is based on a capacitive interaction and there are many known ways of measuring capaciance. In this case we measure the RC time constant of a step response to the circuit.

3.1.1 Design

Taufish is shown in Figure 3-1 and Figure 3-2. The effective circuit for measurements is an RC network where the resistore is internal to the Taufish and the capacitor is the effective coupling between the sense electrode and the environment. A voltage step is applied across the RC ciruit and the time it takes for the capacitor to charge up beyond a fixed threshold is measured. This gives an estimate of $/tau = RC$.

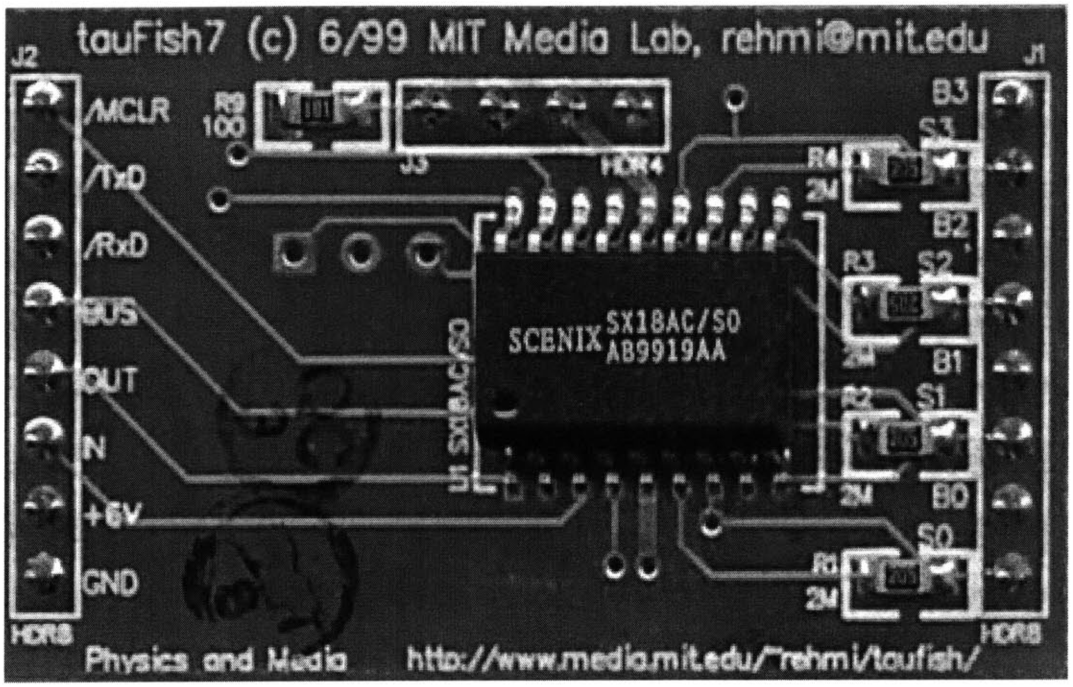

Figure 3-1: Taufish hardware (top view).

The exact measurement sequence consists of the following: a 5V voltage step is generated from a microcontroller and is applied through a resistor to the sense electrode. A timer is run on the MCU and the voltage on the electrode is tracked on an input pin, triggering when it rises above a fixed threshold. This measures the rise time, $\tau = RC$. The decay time is similarly measured by applying a zero potential through the resistor and counting until the electrode potential drops below a fixed threshold. Some measured waveforms are shown in Figure 3-3. The figures show the signals measured by Taufish with and without a hand present to increase the capacitance, and therefore the charge up time.

Included in the Taufish firmware code is a UART (Universal Asynchronous Receiever Transmitter) for receiving instructions from a controlling computer and sending back the measurements. In all our applications we used 115kbps RS-232.

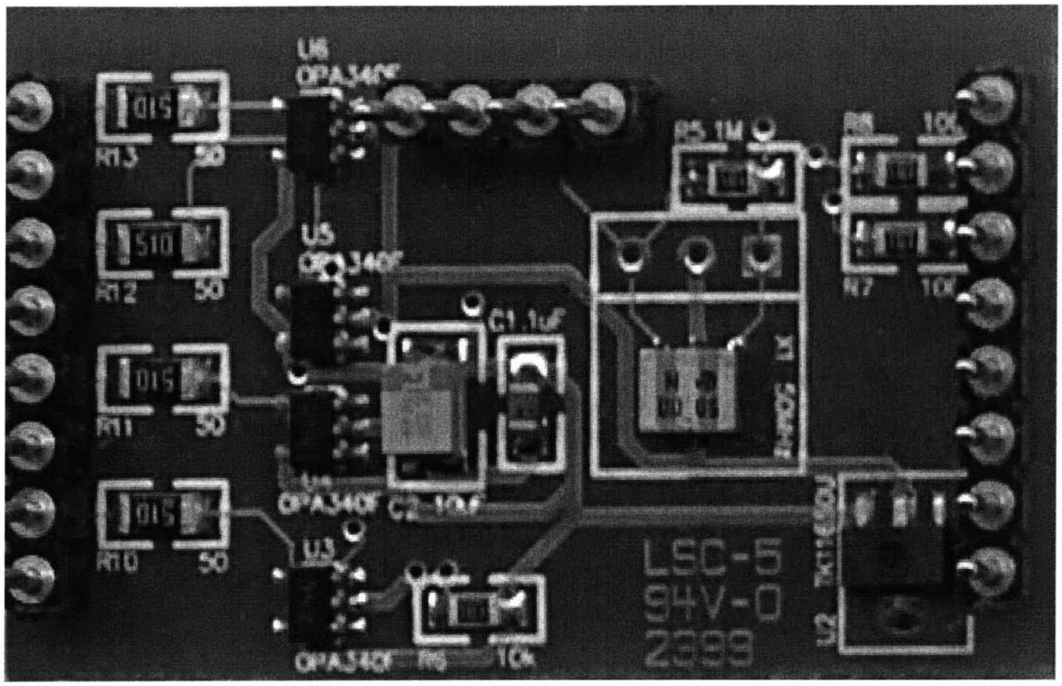

Figure 3-2: Taufish hardware (bottom view).

3.1.2 Shielding and Noise Immunity

The capacitance measured by Taufish will be the lumped sum of all environmental couplings, not just that of the intended body. Proper imaging requires extricating the intended body from everything else. This is primarily a data modeling problem to be dealt with using algorithms covered in a later section. However, there is still some room for good engineering design.

Shielding is one way to limit the area which is measured. In this case, large conducting planes are placed everywhere the environment needs to be blocked off. The sensing electrode voltage is buffered and fed into the conducting planes. This eliminates the capacitive coupling between the sense electrode and the shielding planes (more simply, there is no voltage difference and hence no electric field between the two). Furthermore, the environment behind the shield is blocked off since the conductor acts like a Faraday cage. In our implementations, we placed a shield immediately below the sensing plane to limit the imaging area to everything above the plane.

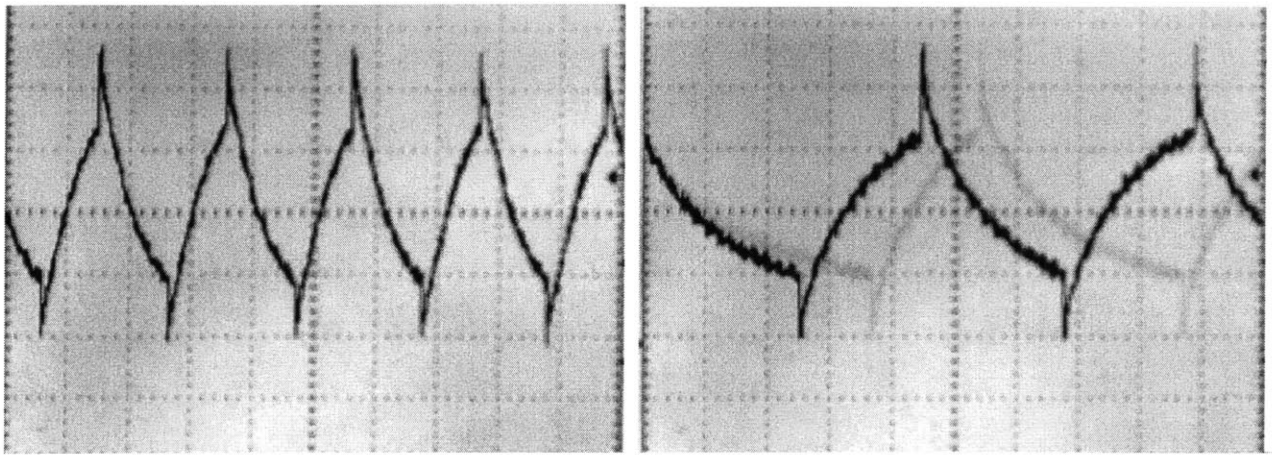

Figure 3-3: Waveform measured by taufish: without hand (left) and with hand (right).

One big advantage of the Taufish measuring scheme is that it has an inborn filtering mechanism for both high-frequency and low-frequency low-voltage noise. Noise with a period short compared to τ will be averaged out after many measurements, thus low-pass-filtering the signal. Also, by looking at both the charging and discharging curves of the RC network, we gain low-frequency immunity. Noise with a period much longer than τ will, for example, accelerate the charge cycle, while decelerating the discharge cycle by the same amount. So, once again, averaging through many measurements will make the system more robust to low frequency noise. For a 10 pF capacitance (typical for a 3x3 inch sense electrode in the presence of a hand) and a 2 MΩ resistance we get $\tau = 20 \times 10^{-6} sec$. This gives 50,000 measurements/second–plenty of time resolution for large-scale averaging.

The measurement scheme described above is still vulnerable to noise which is near the frequency of the charge and discharge cycles. Even if the noise is initially out of phase with the charging measurements, the Taufish will quickly lock onto the noise. We experienced exactly this effect when we placed a Taufish array next to some other sensing equipment in our lab. To combat it, we imposed a fixed period measurement, choosing a period much longer than what any actual measurement period might be. All charge-discharge cycles are now constrained to take the same amount of time. A separate timer runs in parallel with the measurement timer so that after the threshold is crossed the measurement time is stored but the charging continues until the timer elapses. The period is software configurable so in the rare case that the noise is exactly matched in frequency, this will be noticed and the period changed. The system remains vulnerable to any high-voltage noise in the environment.

3.1.3 Performance

Taufish was designed to be highly modular and scalable. Each individual board has four sensing channels and four voltage followers for corresponding shields. For larger applications (such as the MOMA exhibit described earlier) the boards can be tiled to arbitrary geometries and resolution.

One thing to note is that Taufish makes a purely loading mode measurement. Our attempts to measure off diagonal elements of the capacitance matrix (coupling between different sense electrodes) with this configuration have never yielded satisfying results.

3.2 Lazyfish Revisited

As described in the previous chapter, one of the lessons learned from the MOMA exhibit was that the Taufish measurement approach is not highly scalable. Place-mat sized measurement areas required a 5x6 array of Taufish, each driving 4 electrodes (120 electrodes in total!). To implement the resistive sheet trick mentioned in the previous chapter, I made a revised version of the Lazyfish hardware which was developed by Joshua Smith and well-documented, [Smi99]. This board works very effectively in transmit/receive mode, sending a large, 40V, 100 kHz signal from an LC tank circuit and then synchronously undersampling the received signal with an analog to digital converter to extract phase and amplitude information.

The revisions I made to Josh's board are very minor. A picture of the board is shown in Figure 3-4. I modified it to have 2 transmit channels and 8 receive channels. The increased number of receivers allows us to use a large square resistive sheet while measuring current at each corner and each side. Also, the output signal was increased in frequency from 100 kHz to 1 MHz. This improves performance in many respects.

Most importantly, the capacitive impedance ($Z = \frac{1}{j2\pi fC}$) decreases with frequency, providing a larger current pathway and thus a stronger coupling to the body to be imaged. Also, the LC tank, which is resonant at $f = \frac{1}{2\pi\sqrt{LC}}$, has a quality factor $Q = \frac{1}{R}\sqrt{\frac{L}{C}} = \frac{2\pi L}{R} f$ which is linearly dependent on the transmitted frequency. Thus, increasing the frequency increases the Q and the transmitted signal's amplitude. And finally, pushing our signals to higher frequencies decreases the effects of $\frac{1}{f}$ noise[1].

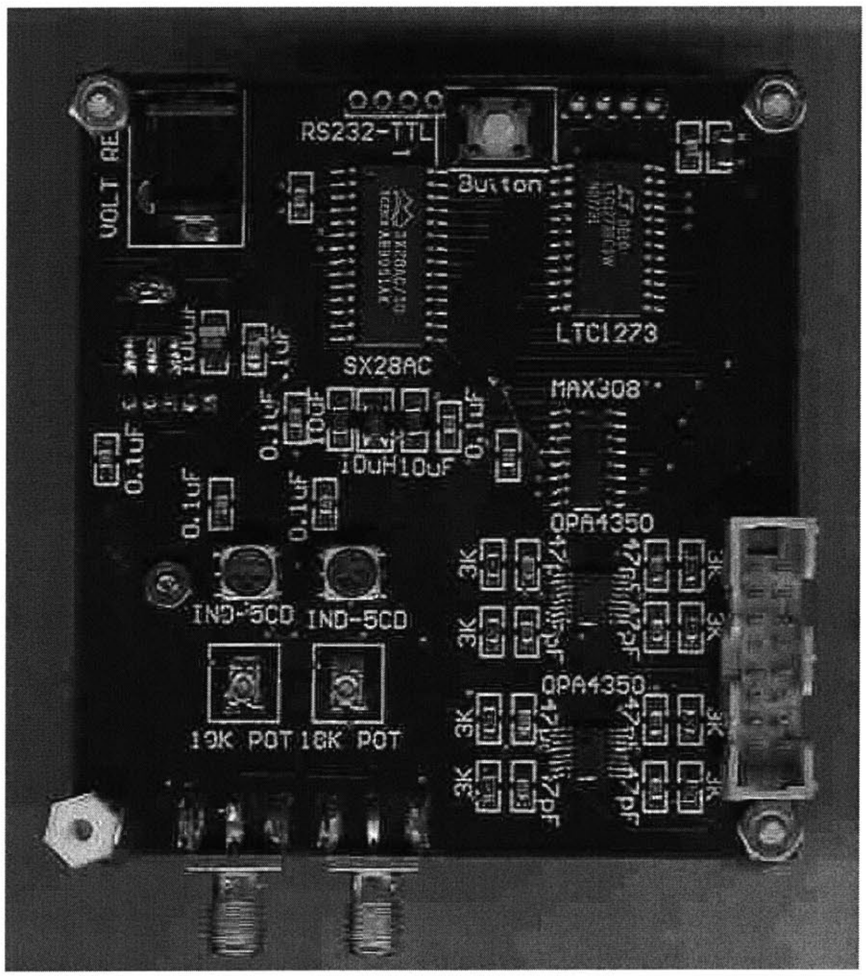

Figure 3-4: Schematic of revised lazyfish

Another change implemented on the revised Lazyfish was a buffer on the output stage. The purpose of this was to prevent de-tuning of the LC tank circuit if a

[1]One could argue that the increased parasitics on a high frequency board easily offset decreased $\frac{1}{f}$ noise.

large capacitive load was applied to the transmitted signal. This specific problem manifested itself when we attempted to use the sensing hardware on the dashboard of a car connected through a coaxial cable. It was discovered that the amplitude of the transmitted signal was greatly diminished as the capacitance was shifted by as much as 50

Chapter 4

Data Modeling and Inversion Algorithms

In this chapter I describe some of the algorithms used in this thesis to perform the inversion from electostatic measurements to geometric parameters. This includes recursive estimation with a Kalman filter as well as direct evaluation of an empirically derived inverse function.

4.1 Requirements

We wish to be able to map sensor readings to position and size rapidly enough to serve as a user interface (update approximately 30 times a second) and with enough resolution to track small movements on the order of 1/10th of an inch. Furthermore, we would like to be able to ignore what is happening in other parts of the room and focus on the immediate vicinity. If a person is using the device we would like to ignore the other person across the room, the elevator down the hall, and even portions of the user's body which we don't wish to track. And finally, we wish our algorithm to adapt to small changes in the behavior of the measurement hardware. That is to say, the forward function from body geometry to measurements on the Fish will change with time due to variations in gain stages, component values, and power lines. We would like the inversion algorithm to adapt to these changes.

The above four requirements have driven nearly all aspects of the development of this thesis, not just the algorithms. From the beginning we have wished to make an electric field user interface, or "gloveless data-glove". The need for rapid updates caused us to abandon the notion of performing detailed imaging and instead focus on tracking objects. The need for high resolution and sensitivity has driven the hardware development, arguably the most time-consuming portion of this thesis. To this end, we have evolved from loading mode to transmit-receive and gradually moved to higher frequencies and faster controllers.

I would like to describe how our inversion algorithms have developed in order to meet the above requirements. But first I will describe the appropriate formalism for function fitting and estimation problems.

evolution in this thesis To simplify, we can assume that only affine transformations will occur to the forward function: changing only a scaling factor or offset. important properties needed in our inversion algorithm are:

robustness in the presence of noisy measurements, rapid convergence, low computational cost, and

4.2 Data Modeling

4.2.1 Function Fitting

Function fitting involves going from a set of data points to an analytic function which can be evaluated at points not in the original data set. This is typically done by expanding the function using a basis set, such as polynomials, splines, gaussians, sinusoids, etc. The basis set is chosen to match the important properties of the underlying function such as smoothness, continuity, slow growth, boundedness, etc. Thus, some prior knowledge of the function's behavior is typically needed to describe it effectively. For a specific set of basis functions, one has the freedom to choose which parameters to vary in order to fit the data, and which to hold fixed. For example, one could hold fixed the order of the polynomials and only vary the coeeficients, or hold fixed the mean of the gaussians, while changing the variance.

More formally, given a model m with adjustable parameters ϕ and measured data d, we would like to find values for the parameters that give predicted data values in close agreement with the measured values. In other words, we seek a ϕ such that $P(\phi|d, m)$ is maximized.

Using Bayes' rule as well as applying a uniform prior and model, we get [Ger99]:

$$max_{\phi} P(\phi|d) = max_{\phi} P(d|\phi) \tag{4.1}$$

We further guess that the errors in our data measurements have a Gaussian distribution. By the central limit theorem this is a good guess if we know nothing else. This assumption is further validated by Josh Smith [Smi95, p.16], who showed that Gaussian quantization noise dominates over other forms of noise.

If we assume that the errors between samples are iid (independent and identically distributed) we can derive the least squares error measure,

$$min_{\phi} \Sigma_{i=1}^{N} [y_i - f(x_i, \phi)]^2 \tag{4.2}$$

Where we have N data points (x_i, y_i) which conform to some functional relationship with the paramaters ϕ:

$$y = f(x; \phi) \tag{4.3}$$

As mentioned earlier, by expanding f in a chosen basis, we can find the free parameters for this basis by solving 4.2 with our known data points.

4.2.2 Estimation with Kalman Filters

Thus far, we have neglected a discussion of time. If the system we are trying to model has some internal dynamics of which we have knowledge, this should be included in the prior. We should not have to re-solve the inversion problem from scratch each time. This is the realm of recursive estimators, of which the Kalman filter is a notable member. It provides the minimum mean square estimation for a linear model in the presence of zero mean noise [Cat89].

This discussion of Kalman filters closely parallels that in [Ger99]. Suppose we have a system, characterized by a vector of parameters, $\vec{x}_t$, and linear (discretized) internal dynamics:

$$\vec{x}_t = \mathbf{A_{t-1}} \cdot \vec{x}_{t-1} + \vec{\eta}_t \tag{4.4}$$

Meanwhile, we observe the internal parameters indirectly, through a linear forward model:

$$\vec{y}_t = \mathbf{B_t} \cdot \vec{x}_t + \vec{\epsilon}_t \tag{4.5}$$

If the noise, $\vec{\eta}$ and $\vec{\epsilon}$, are uncorrelated with zero mean, we can use a Kalman filter to obtain an estimate at time t for the internal state, $\vec{x}_{t|t}$, based only on the estimate at time (t-1), $\vec{x}_{t-1|t-1}$, the internal dynamics $\mathbf{A_t}$, and forward model $\mathbf{B_t}$. Also, since this is a recursive estimator, we need an initial guess for the internal state and the error, $\mathbf{E_{t|t-1}}$, of that guess, just to get us going. Given all this, we can find the new estimate:

$$\vec{x}_{t|t} = \mathbf{A_{t-1}} \cdot \vec{x}_{t-1|t-1} + \mathbf{K_t} \cdot (\vec{y}_t - \mathbf{B_t} \cdot \vec{x}_{t|t-1}) \tag{4.6}$$

where $\mathbf{K_t}$ is the Kalman gain matrix:

$$\mathbf{K_t} = \mathbf{E_{t|t-1}B_t^T(B_tE_{t|t-1}B_t^T + N_t^y)}^{-1} \tag{4.7}$$

and the error matrix is updated by:

$$\mathbf{E_{t|t-1}} = \mathbf{A_tE_{t|t}A_t^T + N_t^x} \tag{4.8}$$

One of the nice properties of the Kalman filter is that it is easy to compute. However, we have assumed from the beginning that the forward model is linear, while ours is definitely non-linear. The way around this is to locally linearize the forward model about the current estimate. If $\vec{f}$ is our actual forward model, then

$$\mathbf{B_t} = \frac{\partial \vec{f}}{\partial \vec{x}}|_{\vec{x}_{t|t-1}} \tag{4.9}$$

Kalman filters are often used in radar tracking problems where the objects being tracked have well-characterized dynamics, and the forward model is known. The biggest obstacle in using a Kalman filter in our problem of electrostatic tracking is the forward model, which is difficult to arrive at analytically, and also changes with humidity, background noise, and hardware aging. However, our attempt to use a Kalman filter was a key milestone in this thesis and motivated later development. I briefly describe the forward model we used and the conclusions from this work.

4.2.3 Image Charges

To arrive at the forward model mapping geometry to sensor measurements, we can focus on the effects free charges in the body have on the sense electrodes. We can model the sensing plane as a large ground plane. This is accurate because the electrodes are held at a virtual ground (through an op-amp) and the electrodes are spaced very closely. In the case of resistive sheets, the currents are small enough that we can view the entire sheet as an equipotential held at ground. This is a common electrostatics problem which can be solved by the method of images. For a charge q held at a position (0,0,z) above the plane, the induced charge distribution on the plane is:

$$\rho = \frac{2qz}{(x^2+y^2+z^2)^{\frac{3}{2}}} \tag{4.10}$$

The current measured on a particular electrode will be proportional to the total charge on that electrode, which can be found by integrating the density across the area.

$$\int\int \rho dxdy = qtan^{-1}(\frac{xy}{z(x^2+y^2+z^2)^{\frac{1}{2}}}) \tag{4.11}$$

Finally, if we follow the signal chain through the various gain stages in the hardware, we can appropriately scale the above model. This model is then locally linearized and fed into a Kalman filter to perform state estimation of the positions and sizes of induced charges.

The results in this work were not wholly effective. For tracking one individual charge (the effective center of charge for the object), the algorithm gave great performance for in plane x,y estimation, but had lousy performance as a z estimator. This was unacceptable since one can find x,y with much simpler algorithms. But it became clear from this work that the forward model, 4.10 had too weak a functional dependence on z to give much information. Changing z only serves to slightly alter the width of the induced charge distribution. Also, z enters into 4.10 almost linearly (if the denominator is dominated by the x and y terms), which means all electrodes become scaled by the same amount, which does not offer enough meaningful information. The goal, then, was to try an alternative model and/or algorithm.

4.3 Empirical Fitting in Measurement Space

If one was compelled (perhaps forcefully) to develop an electrostatic tracking device in under an hour, the simplest approach would be to take a set of data points, expand in some appropriate basis to get a complete forward model, and then invert the function to find geometric information from new data. The reason one does not do this is that the device is unlikely to still work the next day, or perhaps even the next hour: the forward model is no longer accurate when the slightest of perturbations are made to the environment and hardware. Or, so we thought. It turns out there is a simple mechanism to avoid these problems.

Let us consider the space of our measurements. For the sensing configurations described earlier (revised lazyfish used with resistive sheets or planar arrays of electrodes), each of our electrodes performs a current measurement. If we have 8 such electrodes, then a single measurement is a point in $\mathcal{R}^8$.

Now, let's imagine that our hardware is kicked, the humidity suddenly rises, the air between the object and sensors changes in permittivity, and the coupling of the object to the room ground suddenly changes. All of these environmental perturbations are what seem to doom any fixed database mapping measurements to geometry. But let's examine the situation more carefully. The key equation to keep in mind is:

$$I = C\frac{dV}{dt} \tag{4.12}$$

Where I is the current measured at an electrode, C is the coupling to the object, and V is the voltage difference between the two.

If the hardware is perturbed, but remains functional, then the only change we should expect is in the gain stages, thus affecting the scaling of all the measurements equally. If the ground coupling to the room changes, then the amount of current arriving at each electrode is altered, but once again scales all measurements approximately equally. In fact, nearly all sane perturbations we should be concerned about only have the effect of scaling all the measurements. In $\mathcal{R}^8$ space, this corresponds to a change in magnitude of the measurement vector, but the direction is left unperturbed. More concretely, if we normalize our measurements all of the above perturbations will have no affect at all! The fact that the geometry is contained completely in the orientation of vectors in measurement space is a trivial but surprising fact.

This revelation allowed us to experimentally acquire a data set mapping geometry to current measurements, normalize each vector, and use that as our inverse model. Now, when given a new measurement vector, we can determine what geometry it corresponds to by finding the closest matching angles in our data set. This constructs the neighborhood around our actual point. Interpolating in this neighborhood gives a close estimate of the geometry.

Another way of accomplishing the same thing is to take a set of data and perform a function fitting to derive an analytic function mapping current measurements to geometry. The challenge here is to choose the appropriate basis for the function fitting. In our case, we chose to use radial basis functions (RBFs), which are a set of functions which depend only on the distance from the data point to a set of representative locations, $\vec{c}_i$. Specifially, we used a r^3 function. This constructs a mapping of the form:

$$\vec{y} = \sum_{i}^{M} a_i f(|\vec{x} - \vec{c}_i|^3) \tag{4.13}$$

This fitting has proven to effectively capture the behavior of our system. We started by taking a sample set of data moving around a small test body and recording the measured readings. Using an RBF fit, we were able to accurately find out-of-sample data points.

4.4 Conclusions and Future Work

With the lazyfish hardware and an RBF fit over experimentally acquired data, we have shown that we are able to get better than 1/10 inch tracking resolution for an individual test charge. We were able to get this performance using resistive sheets which have very nice scaling features with area. Also, a simple model has emerged for how best to go about performing the inversion from measurement space to geometric parameter space. Because this model works on charges, which are superposable, this provides a generalizable mechanism for inverting more complex distributions, beyond individual charges. In fact, this is a critical next step. If one assumes only one charge above the resistive sheet, then the above model arrives at the effective center of charge for the system around the sense plane. Unfortunately, portions of the user's body other than the hand contribute greatly to the measured signal, almost drowning out what we wish to image. By separating the body into several charges, we can ignore the larger, more stationary charge and focus on the smaller perturbation.

Another related next step is to use the RBF fit as the forward model in a Kalman filter. As described before, this would correctly include the internal dynamics of the system. Also, the Kalman filter could easily be extended for multiple charges, each evolving with different dynamics, and thus would quite naturally be able to handle a large static charge along with a smaller more dynamic charge.

The results thus far are encouraging for future work. Electrostatic tracking has many nice properties other tracking mechanisms lack. These include high scalability (with the resistive sheets), high sensitivity (with the revised hardware), an independence from line of sight measurements, and a low cost, unobtrusive form factor.

Appendix A

Taufish

The Taufish hardware was implemented by Rehmi Post, while I worked on firmware. All work described here is copyrighted MIT Media Lab, 2001.

A.1 Description

Taufish performs a loading mode measurement by timing an RC charge and discharge cycles. The number of cycles measured is configurable. If greater than 1, the times are added together.

A.2 Commands

All commands are issued through RS-232 at a 115200 baud rate, 8 bits, 1 stop bit, no parity.

C # Command - Change number of samples to #, which is one byte value
P # Command - Change period length for fixed period mode
S # Command - Perform a loading mode measurement on channel #, which is between 0 and 3, and then send out measured values which are 6 bytes giving the value in ascii.
s # Command - Same as above, but do not send out measured values

F # Command - Perform a fixed period loading mode measurement on channel #, which is between 0 and 3.

f # Command - Same as above, but do not send out measured values

R # Command - Return measured values for channel #. 6 bytes

r Command - Return measured values for all channels. 6x4=24 bytes

A.3 Firmware Code

```
;;; -*- Mode: asm; mode: font-lock -*-
;;;
;;; tauFish7.src tauFish embeddable electrostatic sensing node
;;; (C) 1998,1999 MIT Media Lab, Rehmi Post, John-Paul Strachan
;;;
;;; History:
;;;
;;; 1/99 Rehmi Post <rehmi@mit.edu>
;;; John-Paul Strachan <jpstrach@mit.edu>
;;; Created this file based on tauFish code for the PIC16F84
;;;
;;; 6/19/99
;;;  tau7_10d is just like 7_10c (no fixed period) except
;;;  all channels are clamped low after sampling.
;;; also modularizes the sampling code in 1 loop for all 4 electrodes.
;;;
;;;
;;; tau7_11d changes the sampling method to have all other
;;; channels charging as well this should increase sensitivity
;;; by minimizing capacitive interaction between the electrodes

;;; tau7_11b fixes problems with the fixed period. seems to work ok now.
;;; tau7_11a adds the clamp high and low features. The command is:
;;;     Board# + H or L
;;; tau7_11 combines the fixed period code and non-fixed period code
;;; the 2 are available as separate routines
;;;   non-fixed is called with S or s (then electrode #)
;;; fixed period is called with F or f (then electrode #)
;;; this modularizes the sampling code in 1 loop for all 4 electrodes
;;;

MY_ID = $01
BCAST_ID = $1f

; Device

device pins18,pages4,banks8,stackx
device turbo,optionx
device oschs
id 'tau7_11d'
reset reset_entry
freq 50_000_000
```

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;EQUATES;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;

_cap_bits = %10101010
_res_bits = %01010101

_chan1 = %00000011
_cap_loc1 = %00000010
_res_loc1 = %00000001

_chan2 = %00001100
_cap_loc2 = %00001000
_res_loc2 = %00000100

_chan3 = %00110000
_cap_loc3 = %00100000
_res_loc3 = %00010000

_chan4 = %11000000
_cap_loc4 = %10000000
_res_loc4 = %01000000

tx_bit = 0
rx_bit = 1
out_bit = 2
busy_bit = 3
in_bit = 4

TX = (1<<tx_bit)
RX = (1<<rx_bit)
BUSY = (1<<busy_bit)
OUT = (1<<out_bit)
IN = (1<<in_bit)

rx_pin = ra.rx_bit
tx_pin = ra.tx_bit
busy_pin = ra.busy_bit

_res0 = rb.0
_res1 = rb.2
_res2 = rb.4
_res3 = rb.6
_cap0 = rb.1
_cap1 = rb.3
_cap2 = rb.5
_cap3 = rb.7


;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;VARIABLES;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;

org 8

byte ds 1
j ds 1
nsamp ds 1
txreg ds 1
reply ds 1
chargel ds 1
```

```
chargem ds 1
chargeh ds 1

org $10
normal = $

delay_rate ds 1
k ds 1
i ds 1
chan_bits ds 1
cap_location ds 1
res_location ds 1
period ds 1
periodcount ds 1
periodtemp ds 1

org $30
charge = $

chargel0   ds 1
chargel1 ds 1
chargel2  ds 1
chargel3 ds 1
chargem0 ds 1
chargem1 ds 1
chargem2 ds 1
chargem3 ds 1
chargeh0 ds 1
chargeh1 ds 1
chargeh2 ds 1
chargeh3 ds 1

org $50
serial = $

tx_high ds 1 ;tx
tx_low ds 1
tx_count ds 1
tx_divide ds 1
rx_count ds 1 ;rx
rx_divide ds 1
rx_byte ds 1
rx_flag ds 1
string ds 1

reply_bit = reply.0
dontcount_bit = reply.1
drive_bit = reply.2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;MACROS;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

clamp_high macro
mov rb, #$FF
endm

clamp_low macro
clr rb
endm
```

```
float_all macro
mov !rb, #$FF
endm

set_chan macro
mov w, chan_bits
or rb, w
endm

clr_chan  macro
mov w, /chan_bits
and rb, w
endm

highz_all macro
mov !rb, #$FF
endm

highz_chan macro
mov !rb, chan_bits
endm

pulse_all_res_high macro
mov w, #_res_bits ;mask out the capacitors
or rb, w ;and set all resistor pins high
mov !rb, #_cap_bits ;set all except caps as output
mov !rb, #$FF ;then tristate everything
endm

pulse_res_high  macro
; sb rx_pin
; jmp interrupted

mov w, res_location ;these 2 lines replace setb _res
or rb, w ;now a mask is used to not alter the other channels
mov !rb, cap_location ;set all except cap as output
mov !rb, chan_bits ;set all except cap & res as output
endm

pulse_all_res_low macro
mov w, #_cap_bits ;mask out the capacitors
and rb, w ;and set all resistor pins low
mov !rb, #_cap_bits ;set all except caps as output
mov !rb, #$FF ;then tristate everything
endm


pulse_res_low macro
; sb rx_pin
; jmp interrupted

mov w, /res_location ;these 2 lines replace clrb _res
and rb, w
mov !rb, cap_location ;set all except cap as output
mov !rb, chan_bits ;set all except cap & res as output
endm

clear_charge_vals macro
clr chargel
clr chargem
clr chargeh
```

```
endm

drive_tx macro
mov m, #$0F
mov !RA, #(RX|IN|OUT) ; make TX an output (as well as BUSY)
endm

tristate_tx macro
mov m, #$0F
mov !RA,#(RX|TX|IN|OUT) ; stop driving the TX bus
endm

enable_rtcc macro
mov !option,#%10011111 ;enable rtcc interrupt
endm

disable_rtcc macro
mov !option,#%11011111 ;disable rtcc interrupt
endm


ten_nops macro
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
endm

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;END MACROS;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; *** 2400 baud (for slower baud rates, increase the RTCC prescaler)
;baud_bit       = 7                        ;for 2400 baud
;start_delay    =       128+64+1                 ; "    "    "
;int_period     =       163                      ; "    "    "
;
; *** 9600 baud
;baud_bit        = 5 ;for 9600 baud
;start_delay     =  16+8+1  ; "    "     "
;int_period      = 163 ; "    "     "
;
; *** 19200 baud
;baud_bit = 4                         ;for 19200 baud
;start_delay = 16+8+1                   ; "    "     "
;int_period = 163                      ; "    "     "
;
; *** 38400 baud
;baud_bit = 3                         ;for 38400 baud
;start_delay = 8+4+1 ; "    "     "
;int_period =  163                      ; "    "     "
;
; *** 57600 baud
;baud_bit = 2 ;for 57600 baud
;start_delay = 4+2+1  ; "    "     "
```

```
;int_period = 217 ; "    "     "
;
; *** 115.2k baud
;baud_bit = 1                        ;for 115.2K baud
;start_delay =       2+0+0                    ; "    "     "
;int_period =  217                         ; "    "     "
;
; *** 115.2k baud with improved sampling (rehmi)
baud_bit = 2                        ;for 115.2K baud
start_delay =       4+2+0                    ; "    "     "
int_period =  109                         ; "    "     "
;
; *** 230.4k baud (for faster rates, reduce int_period - see above*)
;baud_bit =  0                        ;for 230.4K baud
;start_delay =  1+0+0                    ; "    "     "
;int_period  =  217                       ; "    "     "
;


org 0

;
; Interrupt routine - virtual peripherals
;
interrupt ;3  ; interrupt overhead
mov w,#MY_ID
bank    serial                  ;switch to serial register bank

:transmit       clrb    tx_divide.baud_bit      ;clear xmit timing count flag
inc     tx_divide               ;only execute transmit routine
STZ                             ;set zero flag for test
SNB     tx_divide.baud_bit      ; every 2^baud_bit interrupt
test    tx_count                ;are we sending?
JZ      :receive                ;if not, go to :receive
clc                             ;yes, ready stop bit
rr      tx_high                 ; and shift to next bit
rr      tx_low                  ;
dec     tx_count                ;decrement bit counter
movb    tx_pin,/tx_low.5        ;output next bit
;
:receive        movb    c,rx_pin                ;get current rx bit
test    rx_count                ;currently receiving byte?
jnz     :rxbit                  ;if so, jump ahead
mov     w,#9                    ;in case start, ready 9 bits
sc                              ;skip ahead if not start bit
mov     rx_count,w              ;it is, so renew bit count
mov     rx_divide,#start_delay  ;ready 1.5 bit periods
:rxbit          djnz    rx_divide,:rxdone       ;middle of next bit?
setb    rx_divide.baud_bit      ;yes, ready 1 bit period
dec     rx_count                ;last bit?
sz                              ;if not
rr      rx_byte                 ;  then save bit
snz                             ;if so
setb    rx_flag                 ;  then set flag
:rxdone
bank normal
mov     w,#-int_period          ;interrupt every int_period
:end_int retiw                          ;exit interrupt
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;****** End of interrupt sequence********
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

_hello dw 'tau7_11d',13,10, 0

;;;;;;;;;;;;;;;;;;;;;;;
;RESET ENTRY;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;

reset_entry
; set up the ports to be all inputs
mov m, #$0F
mov !RA, #(RX|TX|BUSY|IN|OUT)
; _tx  initially an input (not driving the bus)
mov !RB, #%11111111 ; _cap(0-3) are inputs (initially)
        ; _res(0-3) are inputs

mov m, #$0C ;make sensor pins schmitt trigger
mov !RB,#$00
mov m,#$0F ;set mode back to direction control

bank normal

clr fsr ;reset all ram banks
:loop setb fsr.4
clr ind
ijnz fsr,:loop

serve
mov w, #80
mov nsamp,w
mov w, #1
mov period, w
mov w, #1
mov delay_rate, w
serve1
sb drive_bit
clr rb
snb drive_bit
mov w, #$FF
snb drive_bit
mov rb,w

mov !rb, #$00
clrb reply_bit
call @get_byte
mov w, byte
mov  j, nsamp
cje byte, #'b', binary_respond_all
cje byte, #'D', change_delay_rate
cje byte, #'S', loading1
cje byte, #'s', loading2
cje byte, #'F', fixed_loading1
cje byte, #'f', fixed_loading2
cje byte, #'H', clamp_all_high
cje byte, #'L', clamp_all_low
cje byte, #'R', return_value
cje byte, #'r', respond_all
cje byte, #'C', do_setN
cje byte, #'P', change_period
cjne byte, #'I', serve1
```

```
;send id string
bank serial
mov string, #_hello
call @send_string
jmp servel

;get loading measurement for a specified channel
loading1
setb reply_bit
loading2
call @get_byte
clear_charge_vals
bank normal
mov w,#$0F
and byte,w
cje byte, #0, get_chan1
cje byte, #1, get_chan2
cje byte, #2, get_chan3
cje byte, #3, get_chan4
jmp servel

;get loading measurement for a specified channel
fixed_loading1
setb reply_bit
fixed_loading2
call @get_byte
clear_charge_vals
bank normal
mov w,#$0F
and byte,w
cje byte, #0, fixed_get_chan1
cje byte, #1, fixed_get_chan2
cje byte, #2, fixed_get_chan3
cje byte, #3, fixed_get_chan4
jmp servel

clamp_all_high
setb drive_bit
jmp @servel

clamp_all_low
clrb drive_bit
jmp @servel

change_delay_rate
call @get_byte
mov w, byte
mov delay_rate, w
jmp @servel

change_period
call @get_byte
mov w, byte
mov  period, w
jmp @servel

return_value
call @get_byte
mov w,#$0F
and byte,w
cje byte, #0, repl1
```

```
cje byte, #1, repl2
cje byte, #2, repl3
cje byte, #3, repl4
jmp servel

do_setN
call @get_byte
mov w, byte
mov  nsamp, w
jmp @servel

get_chan1
mov chan_bits, #_chan1
mov cap_location, #_cap_loc1
mov res_location, #_res_loc1
call @sample
bank charge
mov chargel0, chargel
mov chargem0, chargem
mov chargeh0, chargeh
bank normal
jnb reply_bit, servel
repl1
call @respond1
call @send_cr
jmp @servel
get_chan2
mov chan_bits, #_chan2
mov cap_location, #_cap_loc2
mov res_location, #_res_loc2
call @sample
bank charge
mov chargel1, chargel
mov chargem1, chargem
mov chargeh1, chargeh
bank normal

jnb reply_bit, servel
repl2
call @respond2
call @send_cr
jmp @servel
get_chan3
mov chan_bits, #_chan3
mov cap_location, #_cap_loc3
mov res_location, #_res_loc3
call @sample
bank charge
mov chargel2, chargel
mov chargem2, chargem
mov chargeh2, chargeh
bank normal

jnb reply_bit, servel
repl3
call @respond3
call @send_cr
jmp @servel
get_chan4
mov chan_bits, #_chan4
mov cap_location, #_cap_loc4
```

```
mov res_location, #_res_loc4
call @sample
bank charge
mov chargel3, chargel
mov chargem3, chargem
mov chargeh3, chargeh
bank normal
jnb reply_bit, servel
repl4
call @respond4
call @send_cr
jmp @servel

fixed_get_chan1
mov chan_bits, #_chan1
mov cap_location, #_cap_loc1
mov res_location, #_res_loc1
call @fixed_sample
bank charge
mov chargel0, chargel
mov chargem0, chargem
mov chargeh0, chargeh
bank normal
jnb reply_bit, servel
call @respond1
call @send_cr
jmp @servel
fixed_get_chan2
mov chan_bits, #_chan2
mov cap_location, #_cap_loc2
mov res_location, #_res_loc2
call @fixed_sample
bank charge
mov chargel1, chargel
mov chargem1, chargem
mov chargeh1, chargeh
bank normal
jnb reply_bit, servel
call @respond2
call @send_cr
jmp @servel
fixed_get_chan3
mov chan_bits, #_chan3
mov cap_location, #_cap_loc3
mov res_location, #_res_loc3
call @fixed_sample
bank charge
mov chargel2, chargel
mov chargem2, chargem
mov chargeh2, chargeh
bank normal
jnb reply_bit, servel
call @respond3
call @send_cr
jmp @servel
fixed_get_chan4
mov chan_bits, #_chan4
mov cap_location, #_cap_loc4
mov res_location, #_res_loc4
call @fixed_sample
bank charge
```

```
mov chargel3, chargel
mov chargem3, chargem
mov chargeh3, chargeh
bank normal
jnb reply_bit, servel
call @respond4
call @send_cr
jmp @servel

respond_sub1
; send all the charge and discharge values
bank charge
mov chargel, chargel0
mov chargem, chargem0
mov chargeh, chargeh0
jmp @send_data

respond_sub2
; send all the charge and discharge values
bank charge
mov chargel, chargel1
mov chargem, chargem1
mov chargeh, chargeh1
jmp @send_data

respond_sub3
; send all the charge and discharge values
bank charge
mov chargel, chargel2
mov chargem, chargem2
mov chargeh, chargeh2
jmp @send_data

respond_sub4
; send all the charge and discharge values
bank charge
mov chargel, chargel3
mov chargem, chargem3
mov chargeh, chargeh3
jmp @send_data

respond_all
call @respond1
call @respond2
call @respond3
call @respond4
call @send_cr
jmp @servel

binary_respond_all
bank charge
mov chargel, chargel0
mov chargem, chargem0
mov chargeh, chargeh0
mov w,#0
call @send_binary_data
bank charge
mov chargel, chargel1
mov chargem, chargem1
mov chargeh, chargeh1
mov w,#1
```

```
call @send_binary_data
bank charge
mov chargel, chargel2
mov chargem, chargem2
mov chargeh, chargeh2
mov w,#2
call @send_binary_data
bank charge
mov chargel, chargel3
mov chargem, chargem3
mov chargeh, chargeh3
mov w,#3
call @send_binary_data
jmp @servel

;***************
;* Subroutines *
;***************

; These routines are meant to be at the beginning of a page.
; If they aren't, set an org here

;org $200

sample jmp @_sample
load_cycle jmp @_load_cycle
fixed_sample jmp @_fixed_sample
fixed_load_cycle jmp @_fixed_load_cycle

respond1 jmp @respond_sub1
respond2 jmp @respond_sub2
respond3 jmp @respond_sub3
respond4 jmp @respond_sub4
send_cr jmp @send_cr_sub
send_binary_data jmp send_binary_data_sub

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; delays by (number=10)*5 + 3 cycles in turbo mode

delay5 bank normal
mov i, delay_rate
outer_loop
dec i
snz
retp
mov     w, #$FF  ; number
mov  k,w
dl5 nop
decsz k
jmp dl5
jmp outer_loop


putc bank normal
mov txreg,w
jmp @send_byte

puthexn
call @tohex
jmp @putc
```

```
tohex and     w, #$0F
add  PC,W
RETW '0123456789ABCDEF'


;
; Get byte via serial port
;
get_byte bank serial
enable_rtcc

jnb rx_flag,$
clrb rx_flag
bank serial
mov byte,rx_byte
disable_rtcc
bank normal
retp

;
; Send byte via serial port
;

send_byte bank serial
setb tx_pin
drive_tx
mov w, txreg
not w ;ready bits
mov tx_high,w
clr tx_low
setb tx_low.7
mov tx_count,#12 ;1 start + 8 data + 1 stop bits
enable_rtcc
:wait
bank serial
test tx_count
jnz :wait
bank normal
disable_rtcc
tristate_tx
retp

; Send string of ID data

send_string

bank serial
:loop
mov     w,string                ;read next string character
mov     m,#0                    ; with indirect addressing
iread                           ; using the mode register
mov     m,#$F                   ;reset the mode register
test    w                       ;are we at the last char?
jnz :next_char               ;if not=0, skip ahead
bank normal
RETP                            ;yes, leave & fix page bits
:next_char
mov txreg,w
call    send_byte               ;not 0, so send character
bank serial
```

```
inc     string                  ;point to next character
jmp     :loop                   ;loop until done


;returns the loading values
send_data
mov w,<>chargeh
call @puthexn
bank charge
mov w,chargeh
call @puthexn
bank charge
mov w,<>chargem
call @puthexn
bank charge
mov w,chargem
call @puthexn
bank charge
mov w,<>chargel
call @puthexn
bank charge
mov w,chargel
call @puthexn
mov w, #' '
jmp @putc

;returns the loading values
send_binary_data_sub
and w,#3
or w,#(MY_ID<<2)
call @putc
mov w,chargeh
call @putc
mov w,chargem
call @putc
mov w,chargel
jmp @putc

;sends a carriage return
send_cr_sub
mov     w, #13
call @putc
mov     w, #10
call @putc
mov w, #0
jmp @putc


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Sample one channel without fixed period;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;The pins for the channel to be sampled will
;be in chan_bits, res_location, and cap_location

_sample
call @load_cycle
decsz j
jmp _sample
retp
```

```
_load_cycle
clr_chan ; set only this channel low
mov !rb,#$00 ; and drive *everything*
call @delay5 ; for 5 us
; highz_chan ; then tristate the sample chan.
highz_all ; tristate all
loop1
inc chargel  ; increment the counter --
snb STATUS.2 ; all 24 bits
inc chargem  ; worth...
snb STATUS.2
inc chargeh
; pulse_res_high ; source a pulse of current through R,
pulse_all_res_high
mov w, cap_location
and w, rb
snz
jmp loop1 ;
set_chan ; set the cap and res high
mov !rb,#$00 ; and drive *everything*
call @delay5 ; for 5 us
; highz_chan ; then tristate the sample chan.
highz_all
loop2
inc chargel  ; increment the counter --
snb STATUS.2 ; all 24 bits
inc chargem  ; worth...
snb STATUS.2
inc chargeh
; pulse_res_low
pulse_all_res_low
mov w, cap_location
and w, rb
sz
jmp loop2
retp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Sample one channel with fixed period;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;The pins for the channel to be sampled will
;be in chan_bits, res_location, and cap_location

_fixed_sample
call @fixed_load_cycle
decsz j
jmp _fixed_sample
retp
_fixed_load_cycle
clr_chan ; set only this channel low
mov !rb,#$00 ; and drive *everything*
call @delay5 ; for 5 us
; highz_chan ; then tristate the sample chan.
highz_all ; tristate all pins (7/11/99)

clr periodcount ; clear the period counters
clr periodtemp
clrb dontcount_bit
f_loop1
inc chargel  ; increment the counter --
```

```
snb STATUS.2 ; all 24 bits
inc chargem  ; worth...
snb STATUS.2
inc chargeh
dontcount1
; pulse_res_high ; source a pulse of current through R
pulse_all_res_high ; (7/11/99 all channels are charged as well)
inc periodcount
snz ; when periodcount wraps around
inc periodtemp ; increment periodtemp
mov w,periodtemp ; and compare to period
mov w,period - w ; if equal
jz done1 ; we're done
jnb dontcount_bit, next1
ten_nops
jmp dontcount1

next1
mov w, cap_location
and w, rb

snz
jmp f_loop1
setb dontcount_bit
jmp dontcount1
done1
clr periodcount
clr periodtemp
clrb dontcount_bit
set_chan ; set the cap and res high
mov !rb,#$00 ; and drive *everything*
call @delay5 ; for 5 us
; highz_chan ; then tristate the sample chan.
highz_all ; tristate all pins (7/11/99)
f_loop2
inc chargel  ; increment the counter --
snb STATUS.2 ; all 24 bits
inc chargem  ; worth...
snb STATUS.2
inc chargeh
dontcount2
; pulse_res_low ; sink a pulse of current through R,
pulse_all_res_low ; (7/11/99 all channels are charged as well)
inc periodcount
snz
inc periodtemp
mov w,periodtemp
mov w,period - w
jz done2
jnb dontcount_bit, next2
ten_nops
jmp dontcount2
next2
mov w, cap_location
and w, rb

sz
jmp f_loop2
setb dontcount_bit
jmp dontcount2
done2
```

```
;; clamp_low
;; mov !rb, #$00
retp

interrupted
mov rtcc, #245
jmp @servel

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Appendix B

Lazyfish Revisited

I provide a breif description, command summary, parts list, and schematic for the new Lazyfish hardware. All work described here is copyrighted MIT Media Lab, 2001.

B.1 Description

Lazyfish is well documented in Josh Smith's thesis [Smi99]. I quickly review my revision of this board. The transmit stage is simply a 1 MHz square wave generated by the SX microcontroller, which is filtered by a tuned LC tank circuit, and then buffered through a high-voltage op-amp. There are two transmitters. The receive stage consists of 8 transimpedance amplifiers to measure current in each channel. This signal is then multiplexed, amplified once more, and then fed into an analog-to-digital converter. The A/D is controlled by the SX, which also stores the measured voltage value fromt the A/D and communicates through RS-232.

B.2 Parts

The following table describes the components of this board, along with vendor or supplier information.

Description of Part	Manufacturer	Part # and Package	Vendor
A/D Converter	Linear Technologies	LTC1273 SOIC24	Digikey
Microcontroller	Scenix	SX28AC/SOIC	Scenix
Analog Multiplexor	Maxim	MAX308 SO16	Digikey
Op-Amp, Receiver, Quad	Burr-Brown	OPA4350 SSOP	Digikey
Op-Amp, Gain, Single	Burr-Brown	OPA350	Digikey
Voltage Regulator	Linear Technologies	LT1129 SOT223	Digikey
High Voltage Switching Regulator	Linear Technologies	LT1082 TO220-5	Digikey
High Voltage Op-Amp	Texas Instruments	THS4082 SO8	Digikey

B.3 Commands

All commands are issued through RS-232 at a 115200 baud rate, 8 bits, 1 stop bit, no parity.

I Command - Print Identification

L # Command - Change number of samples taken to #. # is one byte between 1 and 255. Default value is 01.

W # Command - Perform measurement on channel # and return the values measured. # must be between 1 and 8. The returned value is of form "123456 123456 123456 123456 123456 123456 \cr\lf" where each number is an ascii character. These correspond to the measured values at 0, 90, 180, and 270 degrees relative to the transmitted signal. If the number of samples is set (by the **L** command) to be greater than one, then the measurements are accumulated for each quadrature value.

C # Command - Change number of ring-up cycles before measurements are taken. This is used to warm-up the LC tank circuit. Default value is 01.

B.4 Firmware Code

```
;;; -*- Mode: asm; mode: font-lock -*-
;;;
;;; Lazy-I fish firmware. Based on techniques in Lazyfish designed by Josh Smith.
;;; Uses transmit mode on resitive sheet and measures the current at various sites on the
;;; sheet.  This is then demodulated with transmitted signal.
;;; Interfaces with the LTC1273 analog-digital converter

;;; John Paul Strachan <jpstrach@mit.edu> 10/6/00
;;; MIT Media Lab (c) 2000
;;; Physics and Media Group


; Device
   device sx281,stackx\_optionx
   device turbo
   device oschs3

id 'lzyifish'
reset reset\_entry
freq 50\_000\_000



;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;EQUATES;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;
rx\_pin = ra.1
tx\_pin = ra.0
LC\_pin1 = ra.2
LC\_pin2 = ra.3


HBEN\_pin = rb.0
Busy\_bar\_pin = rb.1
CS\_bar\_pin = rb.2
RD\_bar\_pin = rb.3

select0\_pin = rb.4
select1\_pin = rb.5
select2\_pin = rb.6

adc\_data\_reg = rc



;Schmitt Trigger, 0 = enabled, 1 = disable
;Level:  CMOS or TTL levels, 0 = TTL, 1 = CMOS
;Weak Pull-up, 0 = enabled, 1 = disabled
;Direction register, 0 = output, 1 = input

RA\_latch equ \%00000000 ;SX18/20/28/48/52 port A latch init
RA\_DDIR equ \%11110010 ;SX18/20/28/48/52 port A DDIR value
RA\_LVL equ \%11111111 ;SX18/20/28/48/52 port A LVL value
RA\_PLP equ \%11111111 ;SX18/20/28/48/52 port A PLP value

RB\_latch equ \%00001101 ;SX18/20/28/48/52 port B latch init
RB\_DDIR equ \%00000010 ;SX18/20/28/48/52 port B DDIR value
```

```
;; set rb.3 as input due to possible short-circuit on board

RB\_ST equ \%11111111 ;SX18/20/28/48/52 port B ST value
RB\_LVL equ \%11111111 ;SX18/20/28/48/52 port B LVL value
RB\_PLP equ \%11111111 ;SX18/20/28/48/52 port B PLP value

RC\_latch equ \%00000000 ;SX18/20/28/48/52 port C latch init
RC\_DDIR equ \%11111111 ;SX18/20/28/48/52 port C DDIR value
RC\_ST equ \%11111111 ;SX18/20/28/48/52 port C ST value
RC\_LVL equ \%11111111 ;SX18/20/28/48/52 port C LVL value
RC\_PLP equ \%11111111 ;SX18/20/28/48/52 port C PLP value

ST\_W equ \$0C ;Write Port Schmitt Trigger setup, 0 = enabled, 1 = disabled
LVL\_W equ \$0D ;Write Port Schmitt Trigger setup, 0 = enabled, 1 = disabled
PLP\_W equ \$0E ;Write Port Schmitt Trigger setup, 0 = enabled, 1 = disabled
DDIR\_W equ \$0F ;Write Port Direction

;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;VARIABLES;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;

org 8

byte ds 1
j ds 1
txreg ds 1
tx\_timer ds 1
repeat\_time ds 1
k ds 1
i ds 1
org \$10
normal = \$

tx\_time ds 1
delay\_rate ds 1
adc\_data ds 1
i\_acc\_l ds 1
i\_acc\_h ds 1
q\_acc\_l ds 1
q\_acc\_h ds 1

org \$30
data = \$

data\_0\_l ds 1
data\_0\_m ds 1
data\_0\_h ds 1
data\_90\_l ds 1
data\_90\_m ds 1
data\_90\_h ds 1
data\_180\_l ds 1
data\_180\_m ds 1
data\_180\_h ds 1
data\_270\_l ds 1
data\_270\_m ds 1
data\_270\_h ds 1
temp ds 1

org \$50
serial = \$
```

```
tx\_high ds 1 ;tx
tx\_low ds 1
tx\_count ds 1
tx\_divide ds 1
rx\_count ds 1 ;rx
rx\_divide ds 1
rx\_byte ds 1
rx\_flag ds 1
string ds 1

; *** 115.2k baud with improved sampling (rehmi)
baud\_bit = 2                    ;for 115.2K baud
start\_delay =      4+2+0                ; "    "     "
int\_period =  109                    ; "    "     "

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;MACROS;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

enable\_rtcc macro
mov !option,\#\%10011111 ;enable rtcc interrupt
endm

disable\_rtcc macro
mov !option,\#\%11011111 ;disable rtcc interrupt
endm

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;END MACROS;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Interrupt routine - virtual peripherals

org 0

interrupt ;3  ; interrupt overhead
bank    serial                  ;switch to serial register bank

:transmit       clrb    tx\_divide.baud\_bit      ;clear xmit timing count flag
inc     tx\_divide               ;only execute transmit routine
STZ                             ;set zero flag for test
SNB     tx\_divide.baud\_bit       ; every $2^{baud\_bit}$ interrupt
test    tx\_count                ;are we sending?
JZ      :receive                ;if not, go to :receive
clc                             ;yes, ready stop bit
rr      tx\_high                 ; and shift to next bit
rr      tx\_low                  ;
dec     tx\_count                ;decrement bit counter
movb    tx\_pin,/tx\_low.5        ;output next bit
;
:receive        movb    c,rx\_pin                ;get current rx bit
test    rx\_count                ;currently receiving byte?
jnz     :rxbit                  ;if so, jump ahead
mov     w,\#9                    ;in case start, ready 9 bits
sc                              ;skip ahead if not start bit
mov     rx\_count,w              ;it is, so renew bit count
mov     rx\_divide,\#start\_delay  ;ready 1.5 bit periods
:rxbit          djnz    rx\_divide,:rxdone       ;middle of next bit?
setb    rx\_divide.baud\_bit      ;yes, ready 1 bit period
dec     rx\_count                ;last bit?
sz                              ;if not
```

```
rr      rx\_byte                ;  then save bit
snz                             ;if so
setb    rx\_flag                ;  then set flag
:rxdone
bank normal
mov     w,\#-int\_period          ;interrupt every int\_period
:end\_int retiw                           ;exit interrupt
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;****** End of interrupt sequence********
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

\_hello dw 'lazy-i-fish',13,10, 0

;;;;;;;;;;;;;;;;;;;;;;;
;RESET ENTRY;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;

reset\_entry

;;; set up ports
disable\_rtcc
call @configure\_ports
bank normal
;;; reset all ram banks
clr fsr
:loop setb fsr.4
clr ind
ijnz fsr,:loop

;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;MAIN CODE STARTS HERE;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;
bank normal
mov tx\_time,\#\$01
mov repeat\_time, \#\$01

start
call @get\_byte
mov w, byte
cje byte, \#'R', drive\_lc
cje byte, \#'C', change\_tx\_time
cje byte, \#'L', change\_number\_measurements
; cje byte, \#'T', transmit\_measure; measure only default channel
cje byte, \#'W', transmit\_measure2; measure one specified channel
cje byte, \#'X', transmit\_default
cje byte, \#'S', do\_adc\_sample
cje byte, \#'A', go\_endless
cjne byte, \#'I', start


;send id string
bank serial
mov string, \#\_hello
call @send\_string
jmp @start
go\_endless
clrb select0\_pin
clrb select1\_pin
call @sample2
call @send\_data
```

```
call @send\_cr
jmp @go\_endless
drive\_lc
bank normal
mov tx\_time,\#30
mov repeat\_time, \#\$01
call @driver
jmp @start
change\_number\_measurements
call @get\_byte
mov w,byte
mov repeat\_time,w
jmp @start
change\_tx\_time
call @get\_byte
mov w,byte
bank normal
mov tx\_time,w
jmp @start
transmit\_measure
;j is the number of measurements performed
;tx\_time measures how long the LC is driven
call @sample
call @send\_data
call @send\_cr
jmp @start

transmit\_default
clrb select0\_pin
clrb select1\_pin
clrb select2\_pin
call @sample2
call @send\_data
call @send\_cr
jmp @start

transmit\_measure2
;j is the number of measurements performed
;tx\_time measures how long the LC is driven
call @get\_byte
mov w,\#\$0F
and byte,w
cje byte, \#0, chan0
cje byte, \#1, chan1
cje byte, \#2, chan2
cje byte, \#3, chan3
cje byte, \#4, chan4
cje byte, \#5, chan5
cje byte, \#6, chan6
cje byte, \#7, chan7
jmp @start

chan0
clrb select0\_pin
clrb select1\_pin
clrb select2\_pin
call @sample2
call @send\_data
call @send\_cr
jmp @start
```

```
chan1
setb select0\_pin
clrb select1\_pin
clrb select2\_pin
call @sample2
call @send\_data
call @send\_cr
jmp @start

chan2
clrb select0\_pin
setb select1\_pin
clrb select2\_pin
call @sample2
call @send\_data
call @send\_cr
jmp @start

chan3
setb select0\_pin
setb select1\_pin
clrb select2\_pin
call @sample2
call @send\_data
call @send\_cr
jmp @start
chan4
clrb select0\_pin
clrb select1\_pin
setb select2\_pin
call @sample2
call @send\_data
call @send\_cr
jmp @start

chan5
setb select0\_pin
clrb select1\_pin
setb select2\_pin
call @sample2
call @send\_data
call @send\_cr
jmp @start
chan6
clrb select0\_pin
setb select1\_pin
setb select2\_pin
call @sample2
call @send\_data
call @send\_cr
jmp @start
chan7
setb select0\_pin
setb select1\_pin
setb select2\_pin
call @sample2
call @send\_data
call @send\_cr
jmp @start

do\_adc\_sample
```

```
clrb select0\_pin
clrb select1\_pin
clrb select2\_pin
call @adc\_sample
bank data
mov w,<>data\_0\_h
call @puthexn
bank data
mov w,data\_0\_h
call @puthexn
bank data
mov w,<>data\_0\_l
call @puthexn
bank data
mov w,data\_0\_l
call @puthexn
call @send\_cr
jmp @start


;***************
;* Jumps
;***************

; These routines are meant to be at the beginning of a page.
; If they aren't, set an org here

org \$200

sample jmp @\_sample
load jmp @\_load
sample2 jmp @\_sample2
send\_cr jmp @\_send\_cr
configure\_ports jmp @\_configure\_ports
get\_byte jmp @\_get\_byte
send\_byte jmp @\_send\_byte
send\_data jmp @\_send\_data
send\_string jmp @\_send\_string
driver jmp @\_driver
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;*********************************************
;Subroutines--Can be called in above code
;*********************************************


;;;

adc\_sample
bank data
clr data\_0\_l
clr data\_0\_h
;set HBEN,CS\_bar,RD\_bar all low to initiate conversion
mov w, rb
and w, \#\%11110010
mov rb, w
call delay\_11\_cycles
:loop
sb busy\_bar\_pin
jmp :loop
```

```
mov w,adc\_data\_reg
bank data
mov data\_0\_l,w

;set HBEN, CS\_bar, RD\_bar all high to finish conversion
mov w, rb
or w, \#\%00001101
mov rb, w
nop
nop
nop

;set CS\_bar, RD\_bar low to get next byte
and w, \#\%11110011
mov rb, w

call delay\_9\_cycles

;get MSB and add to accumulater
mov w,adc\_data\_reg
mov data\_0\_h,w
;set CS\_bar and RD\_bar high and set HBEN low
mov w, rb
or w, \#\%00001100
and w, \#\%11111110
mov rb, w
call delay\_11\_cycles
retp

;;;

delay\_118\_cycles
;3
mov k,\#0 ;1
:loop
inc k ;1
nop ;1
cjne k,\#14, :loop ;6 (4 when not jmp)

nop ;1
retp ;3

delay\_62\_cycles
;3
mov k,\#0 ;1
:loop
inc k ;1
nop ;1
cjne k,\#7, :loop ;6 (4 when not jmp)

nop ;1
retp ;3

delay\_9\_cycles
;3
nop
nop
nop
retp ;3

delay\_11\_cycles
```

```
;3
nop
nop
nop
nop
nop
retp ;3

delay\_12\_cycles
;3
nop
nop
nop
nop
nop
nop
retp ;3

delay\_14\_cycles
;3
nop
nop
nop
nop
nop
nop
nop
nop
retp ;3

delay\_16\_cycles
;3
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
retp ;3

delay\_17\_cycles
;3
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
retp ;3

delay\_19\_cycles
;3
```

```
call delay\_12\_cycles ;12
nop ;1
retp ;3

delay\_20\_cycles
;3
call delay\_14\_cycles ;14
retp ;3

delay\_22\_cycles
;3
call delay\_16\_cycles ;16
retp ;3

delay\_23\_cycles
;3
call delay\_16\_cycles ;16
nop
retp ;3

;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;

\_sample2
bank data
clr data\_0\_l
clr data\_0\_m
clr data\_0\_h
clr data\_90\_l
clr data\_90\_m
clr data\_90\_h
clr data\_180\_l
clr data\_180\_m
clr data\_180\_h
clr data\_270\_l
clr data\_270\_m
clr data\_270\_h
mov j,repeat\_time
again2
call @load2
decsz j
jmp again2
retp
load2
bank normal
;ring up the LC Tank circuit, just to get it going
mov tx\_timer,\#05
mov !ra, \#RA\_DDIR
:loop cjbe tx\_timer, \#0, :done ;4 (6 if jmp)
nop
nop
nop ; added for correction 8/15/01
setb LC\_pin1
dec tx\_timer
call delay\_22\_cycles
nop ; added for correction 8/15/01
clrb LC\_pin1
call delay\_14\_cycles
jmp :loop     ;3
:done
```

```
In\_Phase\_Measurement1

;;; Get in-phase measurements
nop ; added for correction 8/15/01
setb LC\_pin1

;set HBEN,CS\_bar,RD\_bar all low to initiate conversion
mov w, rb
and w, \#\%11110010
mov rb, w

call delay\_20\_cycles
nop ; added for correction 8/15/01
clrb LC\_pin1

; call delay\_19\_cycles
;Changed 8/16/01
;Wait for conversion to finish

call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
:finished\_conv
nop ; added for correction 8/15/01
setb LC\_pin1

mov w,adc\_data\_reg
bank data

add data\_0\_l,w
snc
mov w,\#\$01
snc
```

```
add data\_0\_m,w
snc
inc data\_0\_h

;set HBEN, CS\_bar, RD\_bar all high to finish conversion
mov w, rb
or w, \#\%00001101
mov rb, w
nop
nop
nop

;set CS\_bar, RD\_bar low to get next byte
and w, \#\%11110011
mov rb, w

nop
nop
nop
nop
nop
nop
nop ; added for correction 8/15/01
clrb LC\_pin1
nop
;get MSB and add to accumulater
mov w,adc\_data\_reg

and w,\#\$0F


bank data
nop

add data\_0\_m,w
snc
inc data\_0\_h

;set CS\_bar and RD\_bar high and set HBEN low
mov w, rb
or w, \#\%00001100
and w, \#\%11111110
mov rb, w

call delay\_11\_cycles
nop ; added for correction 8/15/01
setb LC\_pin1
call delay\_23\_cycles
nop ; added for correction 8/15/01
clrb LC\_pin1

call delay\_23\_cycles
nop ; added for correction 8/15/01
setb LC\_pin1

Quad\_Phase\_Measurement1
call delay\_12\_cycles

;set CS\_bar, RD\_bar, and HBEN low to start a conversion
mov w, rb
and w, \#\%11110010
```

```
mov rb, w

call delay\_9\_cycles
clrb LC\_pin1

call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
:finished\_conv
nop ; added for correction 8/15/01
setb LC\_pin1

mov w,adc\_data\_reg
bank data

add data\_90\_l,w
snc
mov w,\#\$01
snc
add data\_90\_m,w
snc
inc data\_90\_h

;set HBEN, CS\_bar, RD\_bar all high to finish conversion
mov w, rb
or w, \#\%00001101
mov rb, w
nop
nop
nop

;set CS\_bar, RD\_bar low to get next byte
and w, \#\%11110011
```

```
mov rb, w

nop
nop
nop
nop
nop
nop
nop ; added for correction 8/15/01
clrb LC\_pin1
nop
;get MSB and add to accumulater
mov w,adc\_data\_reg

bank data
and w,\#\$0F
nop

add data\_90\_m,w
snc
inc data\_90\_h

;set CS\_bar and RD\_bar high and set HBEN low
mov w, rb
or w, \#\%00001100
and w, \#\%11111110
mov rb, w

call delay\_11\_cycles
nop ; added for correction 8/15/01
setb LC\_pin1
call delay\_23\_cycles
nop ; added for correction 8/15/01
clrb LC\_pin1

In\_Phase\_Measurement2

;set CS\_bar, RD\_bar, and HBEN low to start a conversion
mov w, rb
and w, \#\%11110010
mov rb, w

call delay\_20\_cycles
nop ; added for correction 8/15/01
setb LC\_pin1

call delay\_23\_cycles
nop
clrb LC\_pin1

call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
```

```
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles

:finished\_conv
nop ; added for correction 8/15/01
clrb LC\_pin1

mov w,adc\_data\_reg
bank data

add data\_180\_l,w
snc
mov w,\#\$01
snc
add data\_180\_m,w
snc
inc data\_180\_h

;set HBEN, CS\_bar, RD\_bar all high to finish conversion
mov w, rb
or w, \#\%00001101
mov rb, w
nop
nop
nop

;set CS\_bar, RD\_bar low to get next byte
and w, \#\%11110011
mov rb, w

nop
nop
nop
nop
nop
nop
nop ; added for correction 8/15/01
setb LC\_pin1
nop
;get MSB and add to accumulater
mov w,adc\_data\_reg

;Changed so that we use 3 bytes to store data.  Removed the swap instructions and replaced with nops
; mov temp,w
; swap temp
; mov w, temp
```

```
; nop
bank data
and w,\#\$0F
nop

add data\_180\_m,w
snc
inc data\_180\_h

;set CS\_bar and RD\_bar high and set HBEN low
mov w, rb
or w, \#\%00001100
and w, \#\%11111110
mov rb, w

call delay\_11\_cycles
nop ; added for correction 8/15/01
clrb LC\_pin1
call delay\_23\_cycles
nop ; added for correction 8/15/01
setb LC\_pin1
call delay\_23\_cycles
nop ; added for correction 8/15/01
clrb LC\_pin1

Quad\_Phase\_Measurement2
call delay\_12\_cycles
;set CS\_bar, RD\_bar, and HBEN low to start a conversion
mov w, rb
and w, \#\%11110010
mov rb, w

call delay\_9\_cycles
setb LC\_pin1

call delay\_23\_cycles
nop
clrb LC\_pin1

call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles
```

```
nop
clrb LC\_pin1
call delay\_23\_cycles
nop
setb LC\_pin1
call delay\_23\_cycles

:finished\_conv
nop ; added for correction 8/15/01
clrb LC\_pin1

mov w,adc\_data\_reg
bank data

add data\_270\_l,w
snc
mov w,\#\$01
snc
add data\_270\_m,w
snc
inc data\_270\_h

;set HBEN, CS\_bar, RD\_bar all high to finish conversion
mov w, rb
or w, \#\%00001101
mov rb, w
nop
nop
nop

;set CS\_bar, RD\_bar low to get next byte
and w, \#\%11110011
mov rb, w

nop
nop
nop
nop
nop
nop
nop ; added for correction 8/15/01
setb LC\_pin1
nop
;get MSB and add to accumulater
mov w,adc\_data\_reg


bank data
and w,\#\$0F
nop

add data\_270\_m,w
snc
inc data\_270\_h

;set CS\_bar and RD\_bar high and set HBEN low
mov w, rb
or w, \#\%00001100
and w, \#\%11111110
mov rb, w
```

```
call delay\_11\_cycles
nop ; added for correction 8/15/01
clrb LC\_pin1
call delay\_23\_cycles
nop ; added for correction 8/15/01
setb LC\_pin1
call delay\_23\_cycles
nop ; added for correction 8/15/01
clrb LC\_pin1

;;; Let tank die off
mov w,\#RA\_DDIR
or w,\#\%00001100
mov !ra,w
call delay\_62\_cycles
retp

;;;;;;;;;;;;;;;;;;;;;;;;

\_driver
bank normal
mov tx\_timer, tx\_time
mov !ra, \#RA\_DDIR
:loop cjbe tx\_timer, \#0, :done
; mov !rb, \#RB\_DDIR
setb LC\_pin1
dec tx\_timer
call delay\_22\_cycles
clrb LC\_pin1
call delay\_16\_cycles
jmp :loop
:done
mov w,\#RA\_DDIR
or w,\#\%00001100
mov !ra,w
retp

;;;;;;;;;;;;;;;;;;;;;;;;

\_sample
bank normal
clr i\_acc\_l
clr i\_acc\_h
clr q\_acc\_l
clr q\_acc\_h
mov j,repeat\_time
again
call @load
decsz j
jmp again
retp
\_load
retp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; org \$400

\_configure\_ports
mode ST\_W ;point MODE to write ST register
mov     w,\#RB\_ST             ;Setup RB Schmitt Trigger, 0 = enabled, 1 = disabled
mov !rb,w
```

```
mov     w,\#RC\_ST              ;Setup RC Schmitt Trigger, 0 = enabled, 1 = disabled
mov !rc,w

mode LVL\_W ;point MODE to write LVL register
mov     w,\#RA\_LVL             ;Setup RA CMOS or TTL levels, 0 = TTL, 1 = CMOS
mov !ra,w
mov     w,\#RB\_LVL             ;Setup RB CMOS or TTL levels, 0 = TTL, 1 = CMOS
mov !rb,w
mov     w,\#RC\_LVL             ;Setup RC CMOS or TTL levels, 0 = TTL, 1 = CMOS
mov !rc,w

mode PLP\_W ;point MODE to write PLP register
mov     w,\#RA\_PLP             ;Setup RA Weak Pull-up, 0 = enabled, 1 = disabled
mov !ra,w


mov     w,\#RB\_PLP             ;Setup RB Weak Pull-up, 0 = enabled, 1 = disabled
mov !rb,w
mov     w,\#RC\_PLP             ;Setup RC Weak Pull-up, 0 = enabled, 1 = disabled
mov !rc,w

mode DDIR\_W ;point MODE to write DDIR register
mov w,\#RA\_DDIR ;Setup RA Direction register, 0 = output, 1 = input
mov !ra,w
mov w,\#RB\_DDIR ;Setup RB Direction register, 0 = output, 1 = input
mov !rb,w
mov w,\#RC\_DDIR ;Setup RC Direction register, 0 = output, 1 = input
mov !rc,w

mov     w,\#RA\_latch           ;Initialize RA data latch
mov     ra,w
mov     w,\#RB\_latch          ;Initialize RB data latch
mov     rb,w
mov     w,\#RC\_latch           ;Initialize RC data latch
mov     rc,w
retp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Get byte via serial port
;
\_get\_byte bank serial
enable\_rtcc

jnb rx\_flag,\$
clrb rx\_flag
bank serial
mov byte,rx\_byte
disable\_rtcc
bank normal
retp

; Send byte via serial port
call @puthexn


\_send\_byte bank serial
setb tx\_pin
drive\_tx
mov w,txreg
not w ;ready bits
mov tx\_high,w
clr tx\_low
```

```
setb tx\_low.7
mov tx\_count,\#12 ;1 start + 8 data + 1 stop bits
enable\_rtcc
:wait test tx\_count
jnz :wait
bank normal
disable\_rtcc
tristate\_tx
retp

; Send string of ID data

\_send\_string

bank serial
:loop
mov     w,string                ;read next string character
mov     m,\#0                    ; with indirect addressing
iread                           ; using the mode register
mov     m,\#\$F                  ;reset the mode register
test    w                       ;are we at the last char?
jnz :next\_char                 ;if not=0, skip ahead
bank normal
RETP                            ;yes, leave \& fix page bits
:next\_char
mov txreg,w
call    @send\_byte              ;not 0, so send character
bank serial
inc     string                  ;point to next character
jmp     :loop                   ;loop until done


;returns the loading values

\_send\_data
bank data
mov w,<>data\_0\_h
call @puthexn
bank data
mov w,data\_0\_h
call @puthexn
bank data
mov w,<>data\_0\_m
call @puthexn
bank data
mov w,data\_0\_m
call @puthexn
bank data
mov w,<>data\_0\_l
call @puthexn
bank data
mov w,data\_0\_l
call @puthexn
mov w, \#' '
call @putc

bank data
mov w,<>data\_90\_h
call @puthexn
bank data
mov w,data\_90\_h
```

```
call @puthexn
bank data
mov w,<>data\_90\_m
call @puthexn
bank data
mov w,data\_90\_m
call @puthexn
bank data
mov w,<>data\_90\_l
call @puthexn
bank data
mov w,data\_90\_l
call @puthexn
mov w, \#' '
call @putc

bank data
mov w,<>data\_180\_h
call @puthexn
bank data
mov w,data\_180\_h
call @puthexn
bank data
mov w,<>data\_180\_m
call @puthexn
bank data
mov w,data\_180\_m
call @puthexn
bank data
mov w,<>data\_180\_l
call @puthexn
bank data
mov w,data\_180\_l
call @puthexn
mov w, \#' '
call @putc

bank data
mov w,<>data\_270\_h
call @puthexn
bank data
mov w,data\_270\_h
call @puthexn
bank data
mov w,<>data\_270\_m
call @puthexn
bank data
mov w,data\_270\_m
call @puthexn
bank data
mov w,<>data\_270\_l
call @puthexn
bank data
mov w,data\_270\_l
jmp @puthexn


;sends a carriage return
\_send\_cr
mov     w, \#13
call @putc
```

```
mov     w, \#10
call @putc
mov w, \#0
jmp @putc

;;;
;;;Used to send a character--use puthexn if it is in hex
putc bank normal
mov txreg,w
jmp @send\_byte

puthexn
call @tohex
jmp @putc

tohex and     w, \#\$0F
add  PC,W
RETW '0123456789ABCDEF'
```

Bibliography

[Bas94] J. Bastian. Electrosensory organisms. *Physics Today*, pages 30–37, February 1994.

[Cat89] Donald E. Catlin. *Estimation, Control, and the Discrete Kalman Filter*. Springer Verlag, New York, NY, 1989.

[Ger91] N. Gershenfeld. Sensors for real-time cello analysis and interpretation. *Proceedings of the ICMC*, 1991.

[Ger93] N. Gershenfeld. Method and apparatus for electromagnetic non-contact position measurement with respect to one or more axes. *U.S. Patent No. 5,247,261*, 1993.

[Ger99] N. Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, New York, NY, 1999.

[RFA60] L.J. Chu R.M. Fano and R.B. Adler. *Electromagnetic Fields, Energy, and Forces*, section 1.2, pages 10–119. John Wiley and Sons, New York, 1960.

[Smi95] J.R. Smith. Toward electric field tomography. Master's thesis, MIT, August 1995.

[Smi96] J.R. Smith. Field mice: extracting hand geometry from electric field measurements. *IBM Systems Journal*, 35(3,4), 1996.

[Smi99] J.R. Smith. *Electric Field Imaging*. PhD dissertation, MIT, Media Arts and Sciences, February 1999.