Silicon Integration of "Lab-on-a-Chip" Dielectrophoresis Devices

by

Nusraat Fowjia Masood

A Thesis submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Electrical & Computer Engineering University of Manitoba

Winnipeg, Manitoba, Canada

Abstract

To harness the wealth of success and computational power from the microelectronics industry, lab-on-a-chip (LOAC) applications should be fully integrated with silicon platforms. This works demonstrates a dielectrophoresis-based LOAC device built entirely on silicon using standard CMOS (complementary metal oxide semiconductor) processing techniques. The signal phases on multiple electrodes were controlled with only four electrical contacts, which connected to the device using three metal layers separated with interlayer dielectric. Indium tin oxide was deposited on a milled plastic lid to provide the conductivity and optical clarity necessary to electrically actuate the particles and observe them. The particles and medium were in the microfluidic chamber formed by using conductive glue to bond the plastic milled lid to the patterned silicon substrate. A correlation between the particle velocities and the electric field gradients was made using video microscopy and COMSOL Multiphysics ® simulations.

Acknowledgements

I would like to thank Dr. Buchanan for his patience with me. He has provided me much needed guidance personally and professionally. He tirelessly pushed me to reach goals I easily gave up on. I sincerely hope his mentorship continues. I would also like to thank Dr. Thomson who was so approachable and willing to help the many many times I knocked on his office door. I would also like to thank my other committee member, Dr. Lin, for graciously agreeing to be my external. I would like to thank the entire technical staff but in particular Dwayne Chrusch, Daryl Hamelin, Cory Smit and Allen Symmons. I kept them fairly busy but they only occasionally shunned me.

I would like to thank my friends and family. My parents have worked very hard so I could dream and achieve. Many colleagues became more than colleagues. I especially need to thank Graham Ferrier for helping me communicate at an academic level. Without his help this thesis and various abstracts would have read quite poorly. My most sincere thanks go to my husband, Mohammad Ullah. He has withstood various shades of my panic and craze and looks even more handsome for it. He has provided me comfort and glee in the very darkest of sinkholes in my life.

This research was supported by Natural Sciences and Engineering Research Council of Canada (NSERC), the Canadian Foundation for Innovation (CFI) and the Manitoba Research & Innovation Fund (MRIF).

Contents

List of Tables

List of Figures

List of Copyrighted Material for which Permission was Obtained

Figure 2-9 on page 21:

G. Medoro, N. Manaresi, A. Leonardi, L. Altomare, M. Tartagni, R. Guerrieri, “A Lab-on-a-Chip for Cell Detection and Manipulation”, IEEE Sensors Journal, Vol 3, No 3, pg 317-325 June 2003

Figure 2-10 on page 22:

A. Vulto, G. Medoro, L. Altomare, G A Urban, M. Tartagni, R. Guerrieri, N. Manaresi, “Selective Sample Recovery of DEP-Separated Cells and Particles by Phaseguide-Controlled Laminar Flow”, Journal of Micromechanics and Microengineering, Vol 16, pp 1847-1853, 2006

Chapter 1: Introduction

1.1: Motivation

In 1959, the physicist Richard Feynman from the California Institute of Technology gave a lecture entitled "There's Plenty of Room at the Bottom" which would spark the imagination of scientists everywhere even until now. He contemplated out loud the potentials for the new field of manipulating and controlling things on small scale. He shared his insights about building things of the smallest magnitude and examining nature's tiniest wonders. He whimsically looked forward to the year 2000 and how all of human knowledge would conveniently be kept in one's pocket. The man was a visionary; eventually earning a Nobel Peace prize for his work in quantum mechanics. He foresaw the miniaturization of computers from room sized pieces of equipment in the 1960s to the sophisticated Angstrom sized components of transistors. He even anticipated facial recognition software. Some of the things he talked about came into fruition and others did not. This thesis focuses on Feynman's goal of manufacturing something very small and manoeuvring objects at the same very small scale.

Silicon is the workhorse of the microelectronics industry. It's the most commonly used semiconductor used to make integrated circuitry. The primary appeal of silicon is the silicon-dioxide insulator that is grown at high temperatures to form an excellent insulator with an impressive 9eV electrical bandgap. During the oxidation process, silicon is consumed as the oxide grows into and on the surface of

the wafer. The interface between the silicon and grown silicon-dioxide is excellent as well. For these reasons, silicon has largely been used for Complementary Metal Oxide Semiconductor (CMOS) devices for decades now.

The motivation of this thesis was to develop a fabrication process for a silicon-based dielectrophoretic lab-on-a-chip device and then successfully test the device. Silicon substrate design was sought after because although preliminary prototypes can be developed using other materials, ultimately to achieve miniaturization, integration and increase computational strength of lab-on-a-chip devices silicon is the best choice.

1.1: Thesis Outline

This thesis is broken down into six chapters including this introduction chapter. Chapter 2 is a very good place for the reader to begin. Chapter 2 provides an interesting background of many concepts of this thesis including lab-on-a-chip and dielectrophoresis. Chapter 3 includes the design and fabrication details, which are presented such that the reader appreciates the complexities of building a device using microfabrication techniques. Many of the setbacks are documented in chapter 3 along with the solutions implemented in order to overcome them. In chapter 4, the experimental set-up is documented which includes the software control used for test automation. The results are shared in chapter 5 where data suggests the particles were successfully actuated by the electric fields generated by the electrically driven microfabricated electrodes in accordance to dielectrophoresis

principles. Finally, the conclusion in chapter 6 summarizes the body of work and offers direction for future research using the device.

Chapter 2: Background

In order to fully integrate the reader into the area of dielectrophoresis and lab-on-a-chip applications, a brief background is given. First, the concept of lab-on-a-chip is introduced and its potential benefits are presented. Second, the principle of dielectrophoresis is introduced and discussed in some detail. Finally, some dielectrophoresis-based lab-on-a-chip designs are reviewed including a design that has greatly influenced this thesis.

2.1: Lab-on-a-Chip

The Lab-on-a-chip (LOAC) concept emerged in the 1990s [1]. The term lab-on-a-chip is used to describe devices that perform modern laboratory functions such as fluid handling, separation and sensing of analytes on a chip instead of using large bulky equipment and/or complicated and time consuming chemical analysis. Current approaches for manipulating biological cells include optical tweezers, fluorescence or magnetic activated cell sorting, centrifugation, filtration and electric-field based mechanisms. At present, most biological or chemical analysis can only be conducted in centralized labs. Lab-on-a-chip technology, which can be applied in the fields of biology, pharmaceutics, medicine, environmental science and food safety, to name a few, is providing significant opportunities for point-of-care and onsite testing. It is estimated that 90% of the cost and 95% of the time related to molecular diagnostics is lost in sample collection, preparation and transportation

[1-5]. Onsite LOAC approaches offer efficient solutions. The miniaturization of devices onto a microfluidic chip environment reduces the sample sizes and reagents needed for analysis. Samples can be analyzed and necessary precautions can be taken in minutes instead of days. There is no need to transport samples and the likelihood of mislabelled samples in a centralized lab is minimal. A lab-on-a-chip device can act as a detector, counter or filter of chosen cells, organisms, bacteria, DNA or other particles.

Lab-on-a-chip technology is set for commercial development, which will have a profound effect on all aspects of diagnostic testing by enabling testing in non-laboratory settings. Relatively unskilled operators will be able to perform complex clinical tests. The integration of lab-on-a-chip technology into society will be determined by regulations, cost-benefit studies and the interest of the general public.

2.2: Dielectrophoresis

The majority of dielectrophoresis enabled LOAC patents have been filed in the last ten years [6] due to its extensive use in a large number of LOAC applications, particularly for cell sorting and characterization. Dielectrophoresis describes the motion of neutral, dipolar particles in a medium when exposed to a non-uniform electric field. If a neutral dielectric particle is suspended in a medium and exposed to an electric field, the particle will polarize and an electric dipole will form in the

particle. The induced dipole formed depends on the magnitude and frequency of the applied field. It also depends upon the dielectric properties of both the particle and the medium in which it is suspended in [2, 3].

In figure 2-1 below, the field is shown to be spatially uniform. The particle is equidistant from the positive and negative electrodes. A dipole is formed in the small sphere but it remains balanced. The Coulombic and dipolar charges are equal and opposite. Therefore, there is no net force applied to the particle and no movement.

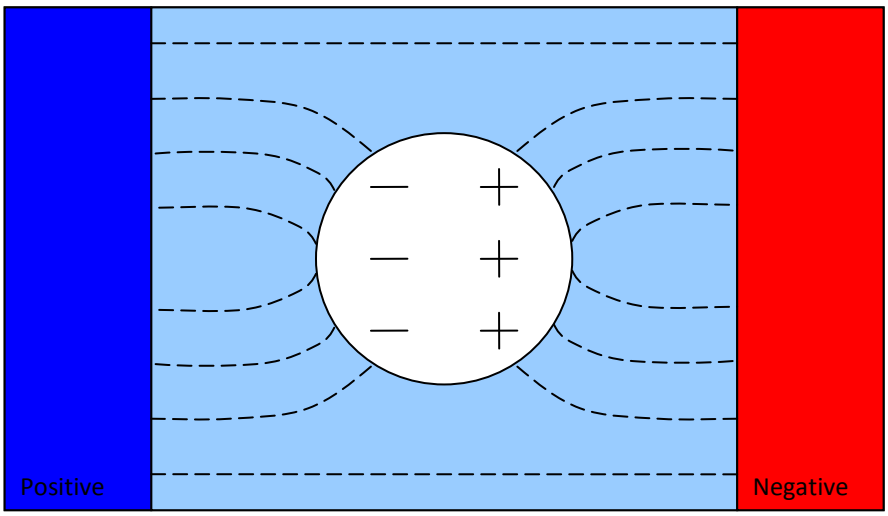


Figure 2-1: Neutral dielectric particle in uniform electric field

If, on the other hand, the applied field is spatially non-uniform, there will be net force acting on the particle which will cause motion as shown below in figure 2-2. The fields are not spatially uniform because the negative electrode is physically smaller than the positive electrode. Consequently, this configuration produces an

electric field gradient. In lab-on-a-chip devices, the sample volume is sufficiently small that the large field squared gradients (V^2/m^3) required for moving particles in fluids (see Equation 2-1) are achieved for only moderate applied voltages (a few Volts). The polarization of the particle relative to its surrounding medium determines the direction of its motion relative to the maximum electric field squared gradient. If a particle is attracted towards regions of increasing field gradient, the particle experiences positive dielectrophoresis, (pDEP). Conversely, if the particle is repelled from regions of maximum field gradient, the particle experiences negative dielectrophoresis (nDEP). In figure 2-2, the two particles respond differently in the non-uniform field. In one case, the green particle, the dipole has aligned opposite to the direction of the increasing field gradient. The green particle is experiencing attraction (pDEP) towards the increasing field gradient. Conversely, the purple particle experiences nDEP. Its dipole has aligned with the increasing field gradient.

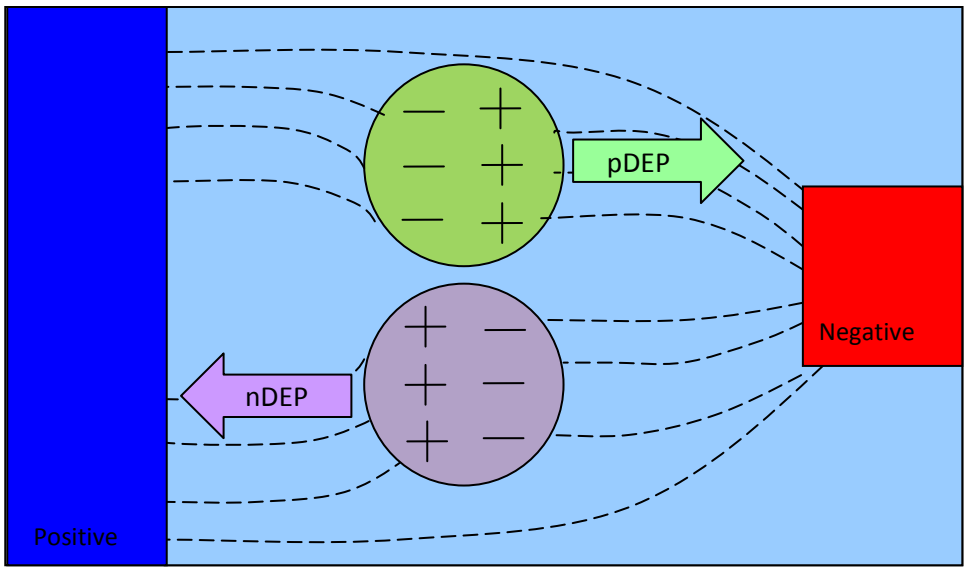


Figure 2-2: Neutral dielectric particles in non-uniform electric field. The green particle is experiences pDEP, it is attracted to the maximum field gradient. The purple particle is experiencing nDEP, it is repelled from the maximum field gradient.

The same principles can be administered on a silicon substrate with microfabricated electrodes as shown in figure 2-3. Depending on the polarities and conductivities of the particles and surrounding medium, some particles will experience pDEP (green) and be drawn towards the electrodes and others will experience nDEP (purple) and be repelled away from the electrodes.

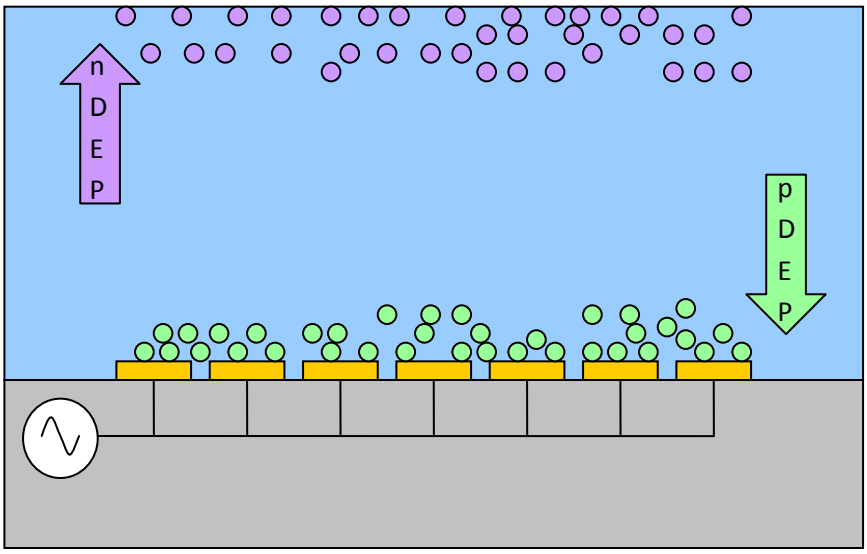


Figure 2-3: Microfabricated electrodes on a silicon substrate creating non-uniform electrode fields. Green particles are drawn towards the increasing field gradient and experience pDEP. Purple particles are repelled away from the increasing field gradient and experience nDEP.

The time-averaged dielectrophoretic force equation is given by Pohl [8]:

$$\left\langle F_{dep} \right\rangle = 2\pi\varepsilon_0\varepsilon_{med} R^3 \,\mathrm{Re}\left[K(\omega)\right]\nabla E_{rms}{}^2 \qquad (2\text{-}1)$$

where,

ε_0 is the permittivity of vacuum,

ε_{med} is the relative permittivity of the medium the particle is suspended in,

R is the radius of the particle,

K(ω) is the Clausius-Mossotti (CM) factor and

E_{rms} is root mean amplitude value of the applied, non-uniform, electric field.

The Clausius-Mossotti (CM) equation for a sphere is given below [10-11]:

$$K(\omega) = \frac{\varepsilon_p^* - \varepsilon_{med}^*}{\varepsilon_p^* + 2\varepsilon_{med}^*} \quad (2\text{-}2)$$

where,

ε_p^* is the complex permittivity of the particle and

ε_{med}^* is the complex permittivity of the medium.

The complex permittivity may be given as:

$$\varepsilon_i^* = \varepsilon_i + \frac{\sigma_i}{j\omega}. \quad (2\text{-}3)$$

In equation 2-3,

ε_i^* is the complex permittivity of the material,

ε_i is the permittivity of the material,

σ_i is the conductivity of the material,

j is square root of -1 and

ω is the angular frequency.

The detailed derivation of the expression for the dielectrophoretic force, equation 2-1, is given in Appendix A.

If the real part of the CM equation (Eqn. 2-2) is positive, the particles experience attraction towards the increasing field gradient. The particles will repel from the higher field gradient if the real part of the CM factor is negative. Since the complex dielectric permittivities of both the particle and medium are frequency dependent, the CM equation is also frequency dependent. A particle in a given medium may be attracted towards the maximum field gradient at one frequency and repelled at another frequency. Below is the Clausius-Mossotti spectrum for polystyrene beads and yeast. The frequencies at which the real value of Eqn. 2-2 are positive is when a particle is experiencing pDEP or attraction towards the increasing field gradient. The frequencies where the particle is repulsed from the increasing field gradient are when the real part of Eqn 2-2 are negative and the particle experiences nDEP. Notice how polystyrene beads remain nDEP. The real part of Eqn 2-2 remains below zero for polystyrene beads throughout the spectra, whereas the yeast can be both pDEP and nDEP. The Matlab plot in figure 2-5 was generated using code fragments written by Sean Romanuik [12]. The code appears in Appendix B.

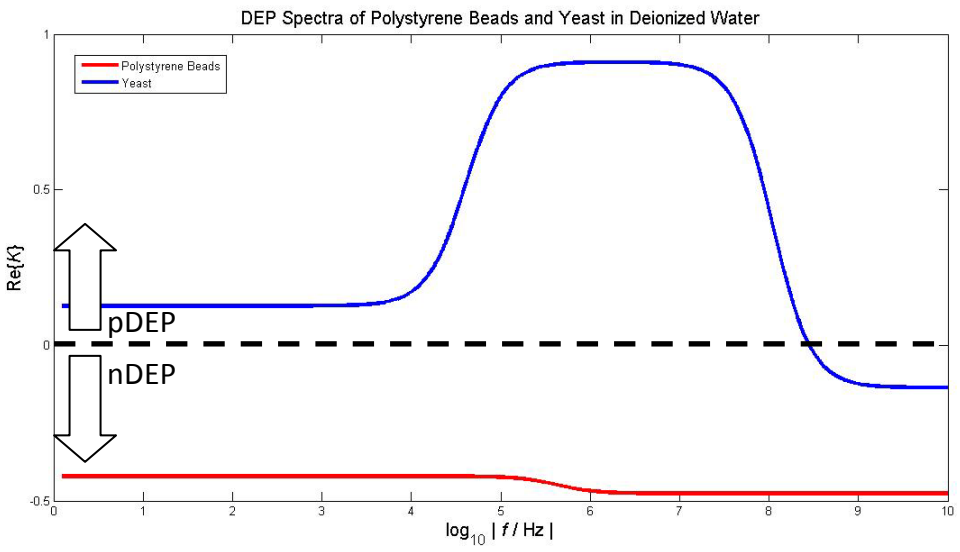


Figure 2-4: Clausius-Mossotti spectra for polystyrene beads and yeast in deionized water.

Polystyrene beads have a consistent nDEP spectra because of the large difference between the permittivity of the DI water (78) versus the beads (2.5). The difference in conductivity (DI water is 5.5×10^{-6} S/m versus beads are 2×10^{-4}S/m) is less of a factor. In general, there is one dispersion or change in the Clausius-Mossotti curves for each interface. Polystyrene spheres are homogenous. There is only one interface between the medium in which the bead is in and the bead itself. As such, in the CM curve above, the real CM values are fairly constant except for a change, or diffusion, in between 100 kHz and 1 MHz where the beads become more strongly nDEP. Yeast has various layers and is modelled using Jones [9] shell method below.

$$\varepsilon_{eq}^{*} = \varepsilon_{shell}^{*}\left[\frac{\left(\frac{r_{sphere}+d_{shell}}{r_{sphere}}\right)^{3}+2\left(\frac{\varepsilon_{sphere}^{*}-\varepsilon_{shell}^{*}}{\varepsilon_{sphere}^{*}+2\varepsilon_{shell}^{*}}\right)}{\left(\frac{r_{sphere}+d_{shell}}{r_{sphere}}\right)^{3}-\left(\frac{\varepsilon_{sphere}^{*}-\varepsilon_{shell}^{*}}{\varepsilon_{sphere}^{*}+2\varepsilon_{shell}^{*}}\right)}\right] \quad (2\text{-}4)$$

The equation above is implemented for multi-shelled particles inside out successively until all the layers are taken into account where:

ε_{eq}^{*} is the combined complex permittivity of the sphere and its surrounding shell,

ε_{shell}^{*} is the complex permittivity of the shell around the sphere,

ε_{sphere}^{*} is the complex permittivity of the material in the inner sphere,

r_{sphere} is the radius of the inner sphere and

d_{shell} is the thickness of the outer shell.

Hölzel's [13] permittivity and conductivity values for yeast were used to generate the plot in figure 2-4 and the values are shown in table 2-1 below. The cytoplasm is the innermost sphere and shells of cytoplasmic wall, periplasmic space, inner cell wall and outer cell wall follow.

Cell Regions (from center out)	Values			
	Conductivity (S/m)	Relative Permittivity	Diameter (m)	Thickness (m)
Cytoplasm	12000e-4	51	6e-6	n/a
Cytoplasmic Wall	0.0302e-4	3	n/a	3.5e-9
Periplasmic Space	41e-4	14.4	n/a	25e-9
Inner Cell Wall	3.04322e-4	60	n/a	110e-9
Outer Cell Wall	200e-4	5.9	n/a	50e-9

Table 2-1: Yeast shell model parameters

Another phenomenon related to this general dielectrophoresis is called traveling wave dielectrophoresis (TWD). If the applied field "travels" through space, a polarizable particle will interact with the electric field as if it was a physical wave. In the figure below, an implementation of a TWD is shown. Electrodes fabricated and interconnected on a silicon substrate such that signals of constant frequency but with a phase difference of 0^{o}, 90^{o}, 180^{o} and 270^{o} are applied to a group of neutral, dipolar particles. The phase of the applied field varies with position. If the particle has a higher polarizability (at that frequency) than that of the medium, the

dipole of the particle will align itself in a direction opposite to the electric field. An attractive force will act on the particles, which follow the direction of the travelling applied field. The pink particles in the figure 2-5 below are following the traveling wave. If the particle is less polarizable than the medium it will move in the opposite direction of the travelling applied field. The beige particles shown in figure 2-5 below are moving against the travelling wave. It is possible to generate electric fields that have spatially variant electric field magnitudes and phases such that a particle may experience both dielectrophoresis and traveling wave dielectrophoresis [4].

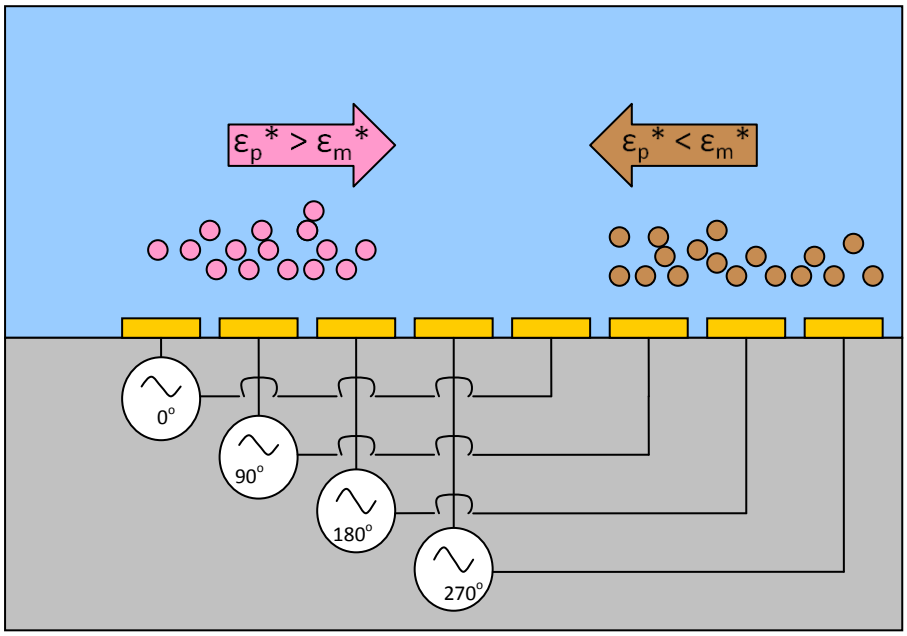


Figure 2-5: Particles experiencing traveling wave dielectrophoresis (TWD). Pink particles are following the wave and brown particles are moving against the wave.

2.3: Dielectrophoresis-Based Lab-on-a-Chip Designs

Washizu, Masuda and Namba [15,16] are accredited with the first dielectrophoretic fluid integrated circuits in the early 1990s. They created cell shift registers to move cells electrostatically. They also manipulated cells to fuse together. A variety of DEP LOAC designs have been made since then. Most designs incorporate microfluidics etched into a substrate to manipulate microliter samples. Applications have focused on cell detection and manipulation for medical applications, particularly cancer. Das et al. [17] exploited pDEP by programming an array of electrodes at different frequencies. Cell mixtures flowed above the electrodes and breast tumour, granulocytes, lymphocytes and erythrocytes bound to different electrodes selectively based on frequency. This design is nicknamed the electrosmear. Another notable design is from Cen et al [18]. They developed a very sophisticated microchip used to analyze malignant cells. Cells were manipulated using DEP and TWD and then characterized using a related technique called electrorotation.

Medoro *et al.* demonstrated a particularly interesting LOAC design where particles in fluid could be manipulated in between electrodes above and below [19]. The electrodes emitted fields at the appropriate frequency for nDEP. In this design, particles are drawn into the space between a specific electrode and the lid because the signals were counter-phase in relation to the other neighbouring electrodes. This pocket of space is referred to as an nDEP cage.

To visualize how the nDEP cage is formed and translated, the electric fields are determined by solving the Laplace equation throughout the fluid chamber using COMSOL Multiphysics®, a commercially available finite element analysis program. The results are shown in Figure 2-6, where the colour bar represents electric field ranging from 0 V/m (blue) to 2.5×10^4 V/m (red). The higher electric field gradients were intentionally left out in the following simulations to focus the reader on the lower field gradients where particles would be drawn to. The dark blue horizontal bar that extends completely across the chamber geometry represents the conductive lid. Three blue horizontal lines below are the coplanar electrodes of width 100µm on the chamber bottom. The microfabricated electrodes are spaced 75µm from each other. The electric field extends from these electrodes mostly upward into the fluid filled chamber of height 60µm, while partially extending into the silicon substrate below. In Figure 2-6, the nDEP cage is formed above the rightmost electrode. For this condition to occur, the rightmost electrode and the conductive lid are driven counter-phase relative to that of the neighbouring electrodes. Since the rightmost electrode and conductive lid are in-phase with each other and have the same voltage, the electric field in the region directly between them is minimal. Therefore, polystyrene spheres, which experience nDEP, will migrate from neighbouring regions having larger fields toward this field null region.

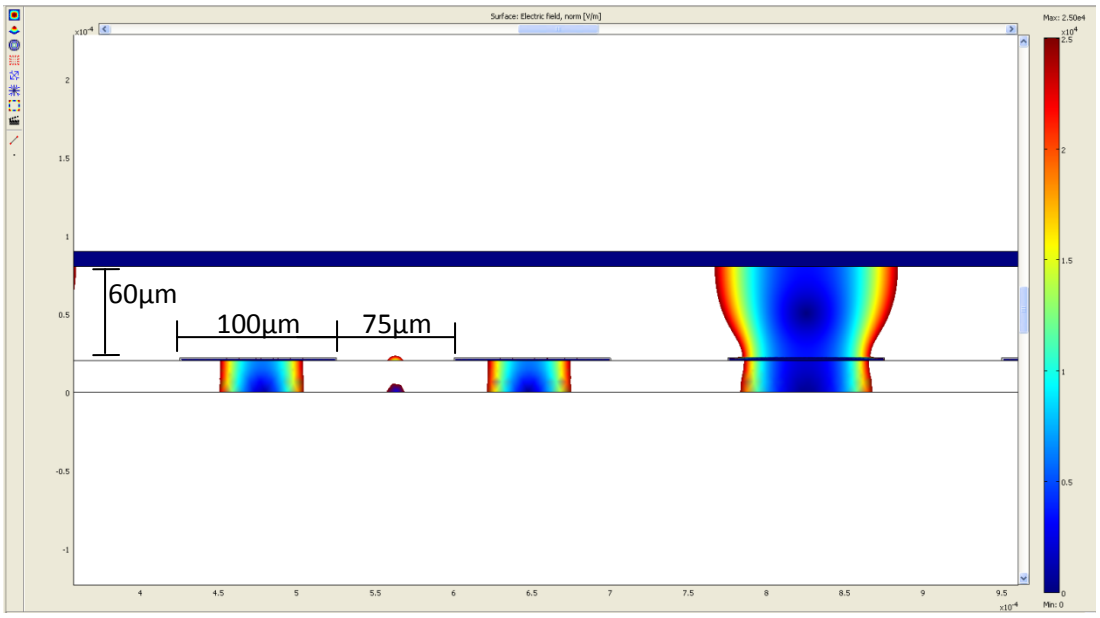


Figure 2-6: Step 1 in nDEP cage movement. Electrode on the far right has the nDEP cage directly above it.

Once the spheres gather in the nDEP cage, they can be collectively translated across the chamber by first switching the phase of the middle electrode. This creates an extended field null region as shown in Figure 2-7. This will allow the spheres to freely translate toward the middle electrode, which is now in-phase with the rightmost electrode and conductive lid. This intermediate step allows the spheres

to translate from electrode to electrode in a fairly continuous fashion.

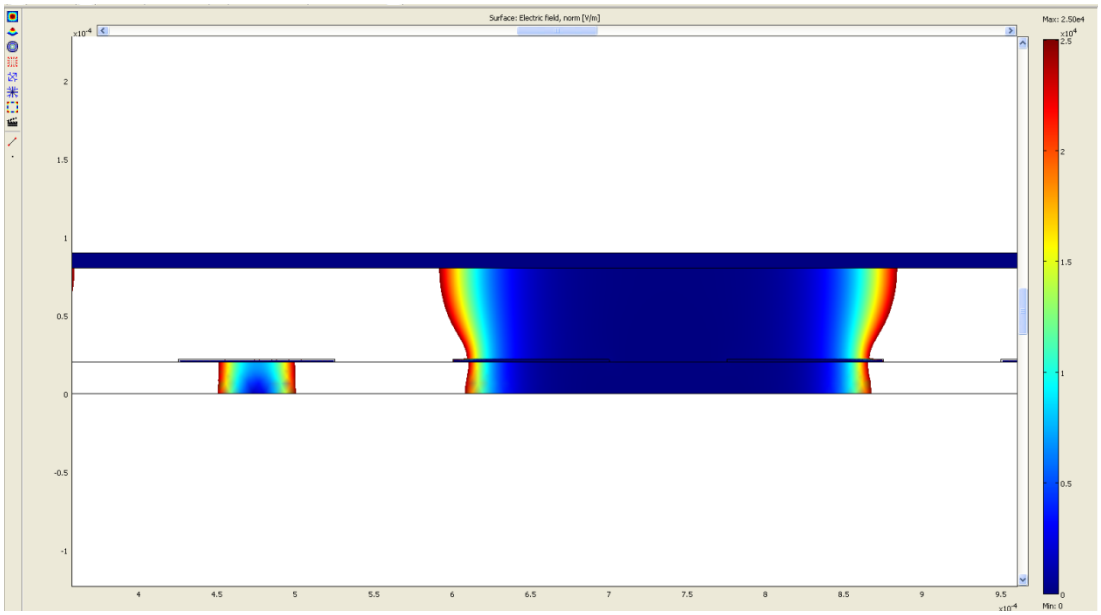


Figure 2-7: Step 2 in nDEP cage movement. The cage has grown one electrode left.

After the spheres disperse within the extended field null, they are collectively translated toward the middle electrode by driving the rightmost electrode counter-phase relative to the middle electrode and conductive lid. As shown in Figure 2-8, the nDEP cage has reduced to its original size and is established between the middle electrode and conductive lid. As a result, the particles follow the nDEP cage one electrode left.

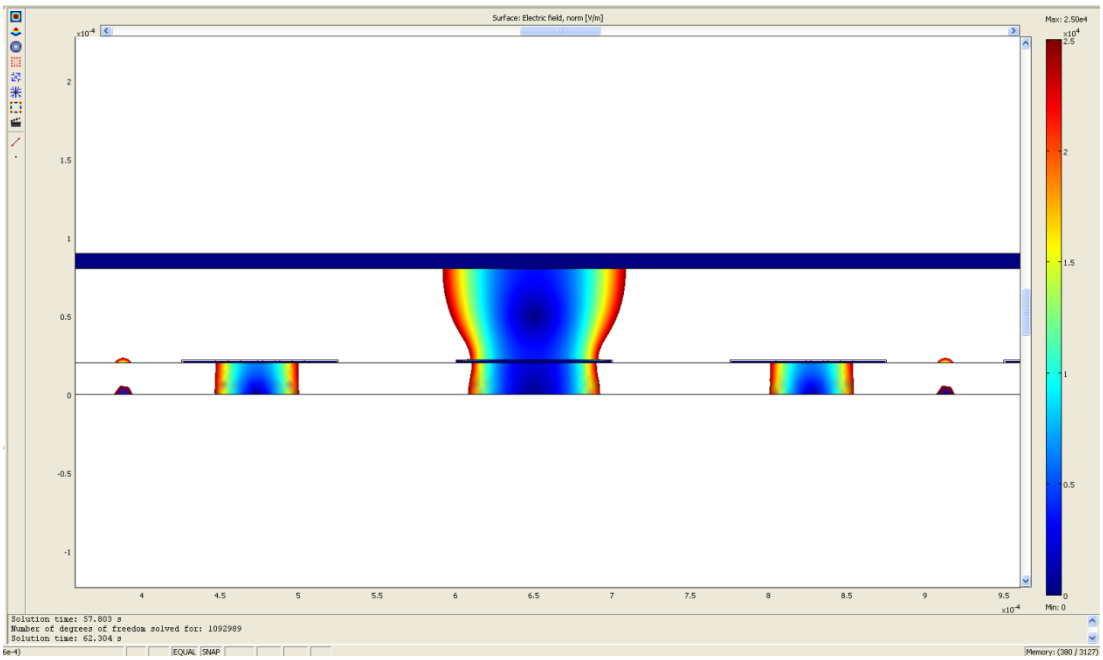


Figure 2-8: Step 3 in nDEP cage movement. The cage has shrunk back down to one electrode but has moved one position left.

The three steps described above clearly show how the nDEP cage can be moved from one electrode to the next.

Preliminary fabrication of the Medoro *et al.* dielectrophoretic cage device included a PCB board with electrodes patterned on it with a parallel conductive clear lid. The two parallel plates had gaskets to prevent leaking and optical fibers on the ends to delimit the spacing between each other. Lastly, clamps held the entire structure together. Figure 2-9 below shows the preliminary design [19].

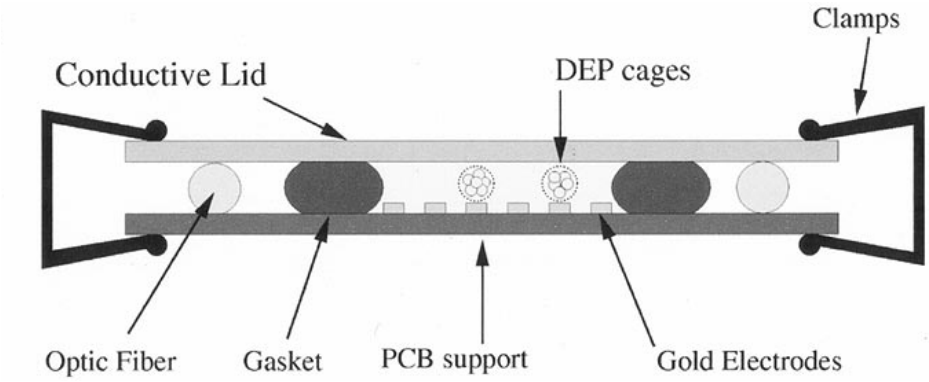


Figure 2-9: The preliminary Medoro *et al.* DEP LOAC adaptation as found in [19]. Reproduced with permission. © 2003 IEEE

In subsequent work, the Medoro group fabricated more sophisticated designs [20]. The Medoro research group still employed PCB board for the interconnections but now use SmartSlide by WaferGen, a commercially available silicon substrate that had electrodes patterned on it. A microfluidic chamber fabricated with dry photo resist is used in the middle and a conductive clear glass slide with powder blasted holes creates various inlet/outlet ports and phase-guided laminar control. That design is shown in figure 2-10 below.

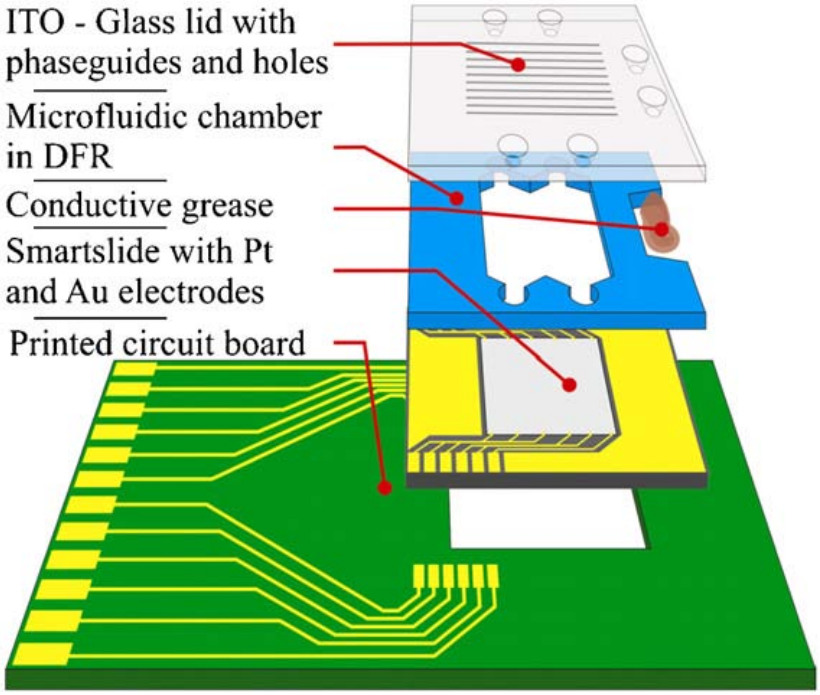


Figure 2-10: The latest Medoro *et al.* DEP LOAC adaptation as found in [20]. Reproduced with permission.

This thesis has been greatly influenced by the work done by Medoro and his colleagues. In the following chapters a different implementation of a dielectrophoresis-based lab on a chip design will be presented.

[1] C.S. Effenhauser and A. Manz, "Miniaturizing a Whole Analytical Laboratory Down to a Chip Size", American Laboratory, Vol 26, pp 15, 1994

[2] H. Anderson, A. Berg, "Microfluidic Devices for Cellomics: a Review", Sensors and Actuators, Vol 92, pg 315 – 325, 2003

[3] Y. Ghallab, W. Badawy, “Sensing Methods for Dielectrophoresis Phenomenon: From Bulky Instruments to Lab-on-a-Chip”, IEEE Circuits and Systems Magazine, Third Quarter, pg 5 – 15, 2004

[4] M. P. Hughes, *Nanoelectromechanics in Engineering and Biology*, CRC Press LLC, New York, 2003

[5] P.R.C. Gascoyne, J.V. Vykoukal, “Dielectrophoresis-Based Sample Handling in General-Purpose Programmable Diagnostic Instruments”, Proceedings of the IEEE, Vol 92, No 1, pg 22 – 42, January 2004

[6] M. P. Hughes, “Strategies for Dielectrophoretic Separation in Laboratory-on-a-chip Systems”, Electrophoresis, Vol 23, pg 2569-2582, 2002

[7] L. Kricka, “Microchips, Microarrays, Biochips and Nanochips: Personal Laboratories for the 21st Century”, Clinica Chimica Acta, Vol 307, pg 219 – 223, 2001

[8] H.A. Pohl, *Dielectrophoresis*, Cambridge University Press, Cambridge 1978.

[9] T.B. Jones, *Electromechanics of Particles,* Cambridge University Press, Cambridge, 1995

[10] O. F. Mossotti, "Discussione analitica sull'influenza che l'azione di un mezzo dielettrico ha sulla distribuzione dell'elettricità alla superficie di più corpi elettrici disseminati in esso," Memorie di Matematica e di Fisica della Societa Italiana delle Scienze, vol. XXIV, Parte seconda, pp. 49-74, (Modena), 1850.

[11] R. J. E. Clausius, *Die mechanische Behandlung der Electricität*. Braunschweig: F. Vieweg, 1879.

[12] Romanuik, S.F., *A Microflow Cytometer with Simultaneous Dielectrophoretic Actuation for the Optical Assay and Capacitive Cytometry of Individual Fluid Suspended Bioparticles*. Thesis (Masters). University of Manitoba, 2009

[13] R. Hölzel, "Electrorotation of single yeast cells at frequencies between 100 Hz and 1.6 GHz", J. Biophys., vol. 73, pp. 1103-1109, 1997.

[14] X.B. Wang, M.P. Hughes, Y. Huang, F.F. Becker, P.R.C. Gascoyne, "Non-Uniform Spatial Distributions of Both the Magnitude and Phase of AC Electric Fields Determine Dielectrophoretic Forces", Biochimica et Biophysica, Vol 1243, pg 185-194, 1995

[15] M. Washizu, T. Nanba, S. Masuda, "Novel Method of Cell Fusion in Field Constriction Area in Fluid Integrated Circuit", IEEE Transactions on Industry Applications, Vol 25, pg 732-737, 1989

[16] M. Washizu, T. Nanba, S. Masuda, "Handling Biological Cells using a Fluid Integrated Circuit", IEEE Transactions on Industry Applications, Vol 26, pg 352 -358, 1990

[17] C. M. Das, F. Becker, S. Vernon, J. Noshari,C. Joyce, and P. R. C. Gascoyne, "Dielectrophoretic Segregation of Different Human Cell Types on Microscope Slides", Analytical Chemistry, Vol. 77, No. 9, pg 387 – 401, 2005

[18] E.G. Cen, C. Dalton, Y. Li, S. Adamia, L.M. Pilarskib and K.V.I.S. Kaler, "A Combined Dielectrophoresis, Traveling Wave Dielectrophoresis and Electrorotation Microchip for the Manipulation and Characterization of Human Malignant Cells", Journal of Microbiological Methods, Vol. 58, pg. 387–401, 2004

[19] G. Medoro, N. Manaresi, A. Leonardi, L. Altomare, M. Tartagni, R. Guerrieri, “A Lab-on-a-Chip for Cell Detection and Manipulation”, IEEE Sensors Journal, Vol 3, No 3, pg 317-325 June 2003

[20] A. Vulto, G. Medoro, L. Altomare, G A Urban, M. Tartagni, R. Guerrieri, N. Manaresi, “Selective Sample Recovery of DEP-Separated Cells and Particles by Phaseguide-Controlled Laminar Flow”, Journal of Micromechanics and Microengineering, Vol 16, pp 1847-1853, 2006

Chapter 3: Design and Fabrication

The traveling wave dielectrophoresis (TWD) design illustrated in figure 2-5 has the distinct advantage of requiring only four contacts to control n-electrodes. However, the design requires sinusoidal wave signals at 0^{o}, 90^{o}, 180^{o}, and 270^{o} phases that are fixed to continually actuate particles in one direction (left or right). The Medoro [1,2] cage device uses one contact for each electrode and hence n-electrodes require n-contacts which is costly. However, the cage device offers more functionality compared with the TWD design since it can be programmed to switch the electrode phases in sequences corresponding to left or right directions on the fly, as well as selectively trapping particles. The device presented in this thesis aims toward exploiting the advantages of each design. It is fabricated completely on silicon, and designed such that particles can be trapped and moved with as few as four contacts.

This chapter begins with a description of the four-contact design for particle movement using nDEP cages. Details on the device fabrication are discussed beginning with the importance of alignment marks for ensuring that the metal layers used in the four contact design align and do not short. Next, two iterations of the device design are explained with emphasis on the improvements of the working second design. Finally, the development of a recipe for sputtering oxide is outlined.

3.1: Four Contact Design for nDEP Cage Movement

Dielectrophoretic cage devices offer more control through switches than TWD designs because in TWD designs the four signals are either on or off. However in DEP cage devices, the duration of time in which the cages hold the particles is controlled. The dielectrophoretic cage devices have historically required one contact per electrode. The design presented in this thesis is a DEP cage device but only requires four electrical contacts for particle movement.

In figure 3-1 below, three electrodes (A, B and C) are driven such that an nDEP cage is created and manipulated to translate particles rightward. At time = t_0, the shaded nDEP cage is formed and particles are drawn above electrode A. As described in chapter 2.3, the nDEP cage is formed by driving a counter phase signal with respect to the neighbouring electrodes. The conducting lid, not pictured, continuously drives the counter phase signal throughout all the time steps. At time = t_1, the shaded nDEP cage is enlarged to include both electrodes A and B. When the nDEP cage is shrunk back to its original size over electrode B at time = t_2, the particles have followed the cage one step to the right. In further steps shown below, time = t_3 and time = t_4, the nDEP cage is manipulated to move the particles a further step right. See figure 3-1 below.

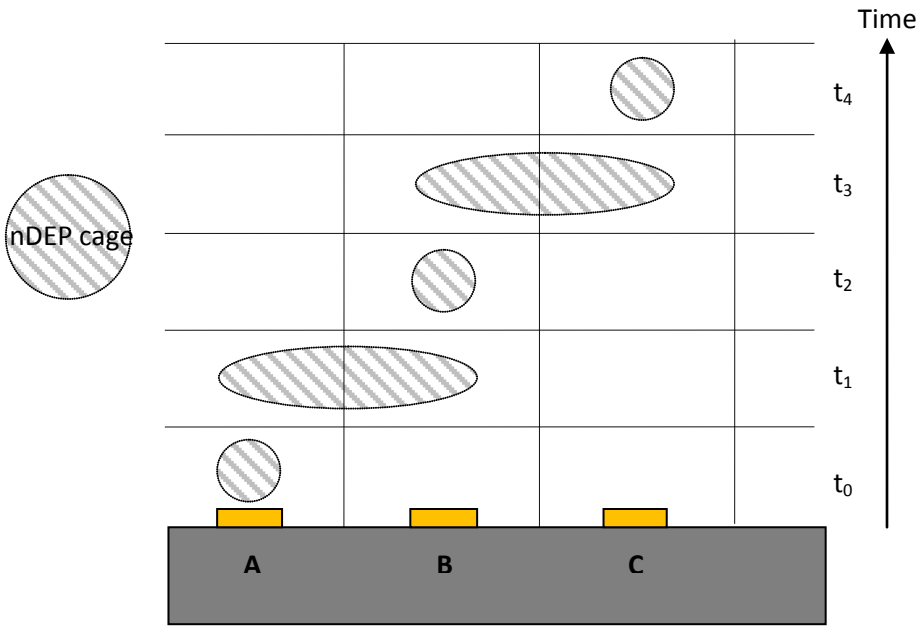


Figure 3-1: nDEP cage manipulation using three electrodes

Now picture many more electrodes and wave after wave of nDEP cages pulling particles right. This is shown in figure 3-2 below.

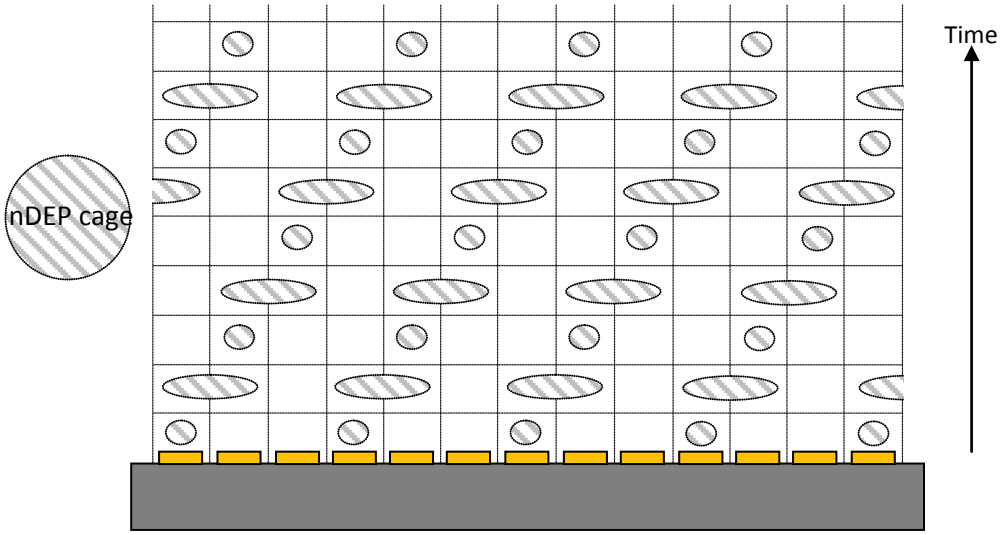


Figure 3-2: nDEP cage manipulation using many electrodes.

The sequence of signals that could be used to form and move the nDEP cages are shown in figure 3-3. For example, all the electrodes labelled A to M including the lid could drive a five volt sinusoidal signal at 10 kHz. The electrodes with the red signal are applied with a 180^0 phase shift (i.e. $\theta = 180^o$) and the white signals do not have the phase shift (i.e. $\theta = 0^o$). The conductive lid, not pictured, is continuously driven by the 180^0 offset signal. The situation described is shown in figure 3-3.

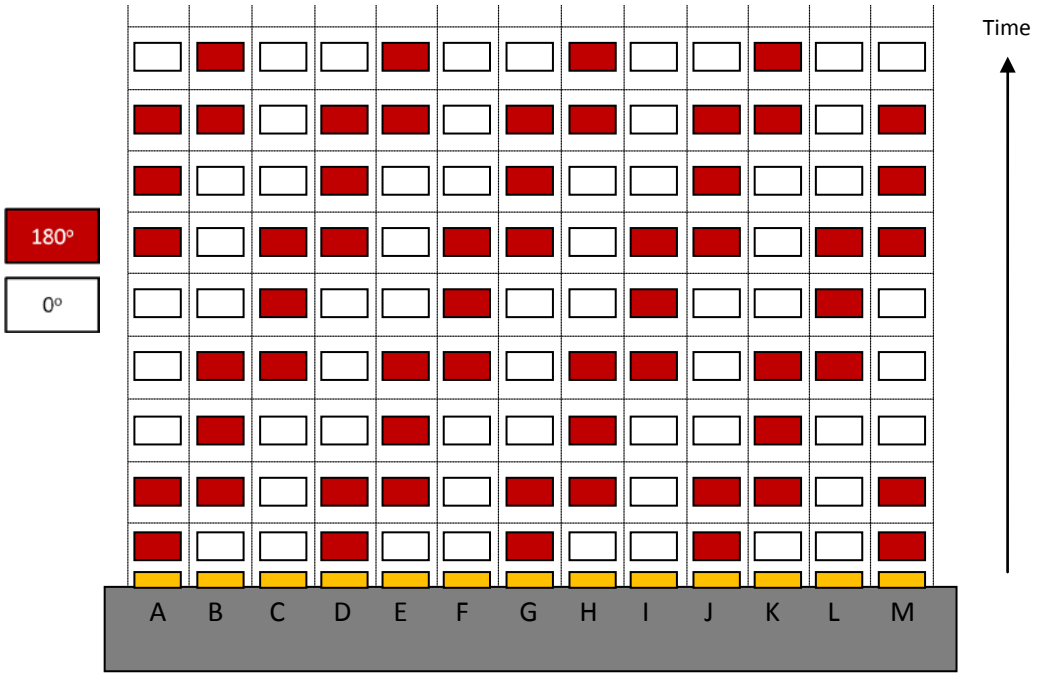


Figure 3-3: Signals for nDEP cage manipulation using many electrodes. All the signals are sinusoidal and have the same amplitude and frequency but the red signals and the lid (not pictured) have a 180^0 phase shift to them.

Whenever designing circuits, it is best to reduce the number of contacts to reduce cost. It is useful to notice how one set of the electrodes have the same signals

throughout. For example, in figure 3-4 below, electrodes labelled A, D, G, J and M could be shorted together because the same sequence of signals is switched between them.

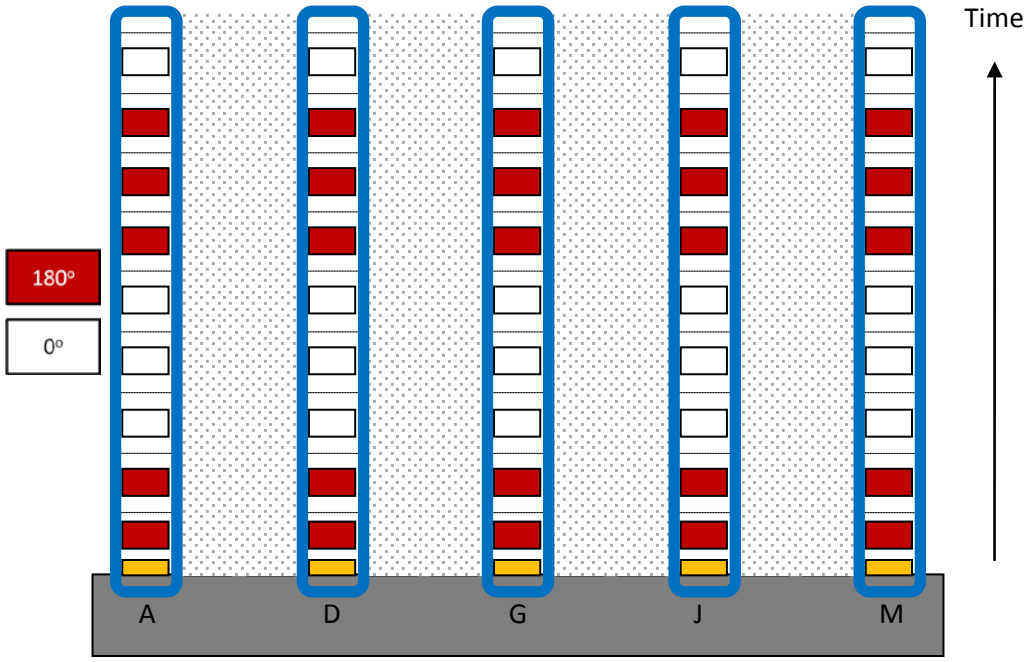


Figure 3-4: Electrodes A, D, G, J and M have the same sequence of signals.

Furthermore, figure 3-5 shows that electrodes B, E, H and K have the same sequence of signals.

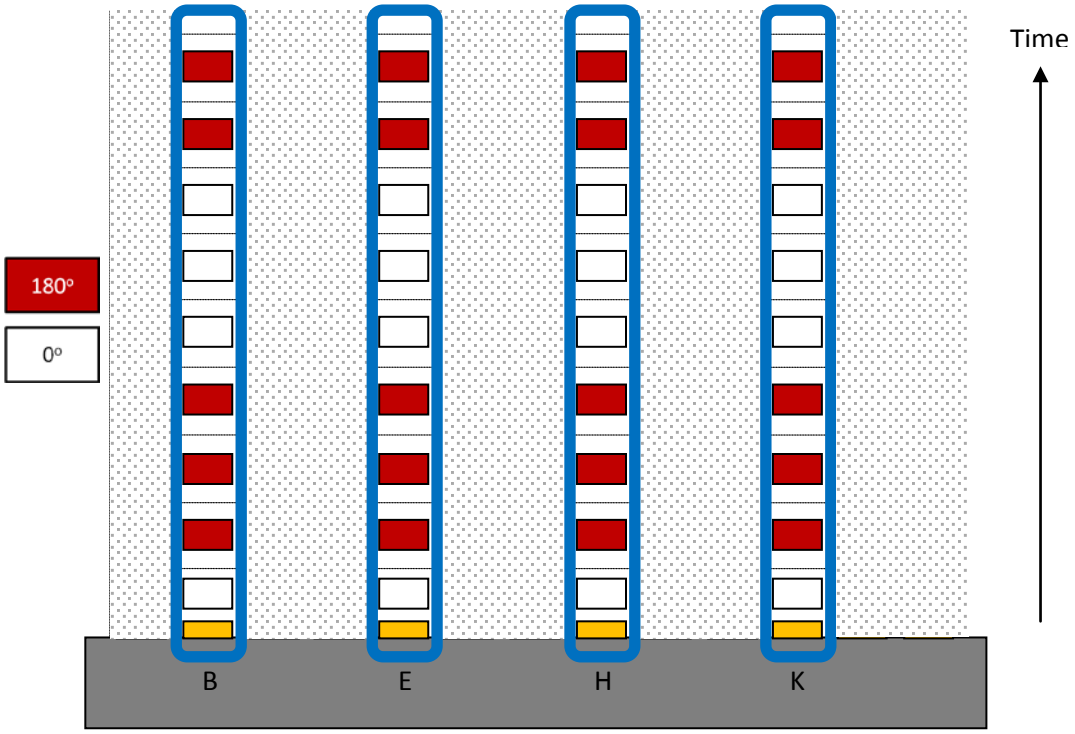


Figure 3-5: Electrodes B, E, H and K have the same sequence of signals.

And lastly, figure 3-6 below shows that electrodes C, F, I, and L could be tied together.

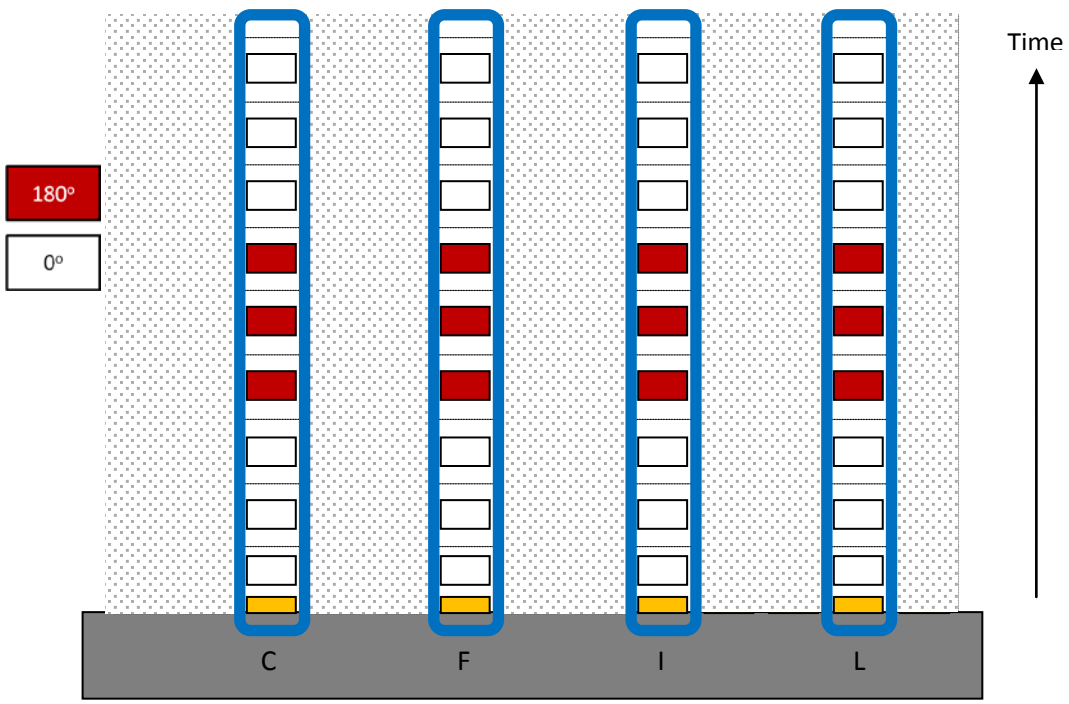


Figure 3-6: Electrodes C, F, I and L have the same sequence of signals.

Regardless of the number of electrodes, with the help of a switch, only four contacts are needed to coordinate nDEP cage movement. Three contacts are needed for the patterned electrodes on the substrate as illustrated in figures 3-3 to 3-6. An additional contact is needed to drive the conductive optically clear lid with the continuous counterphase (i.e., $\theta = 180^{o}$) signal. In addition to reducing cost, having similar lines tied together reduces addressing time. A supplementary electrode could be used to drive all the nDEP cages toward it. This target electrode would need a constant nDEP cage over it. Keeping the lid and the target electrode on separate contacts would enable additional functionality of capacitive measurement between them. These capacitive measurements could be used to determine how many particles were "caught" over the target electrode.

3.2: Alignment Marks

Complementary metal oxide semiconductor (CMOS) fabrication is generally planar. In order to interconnect the lines as needed multiple metal layers and multiple insulation layers are needed to prevent unattended shorts. The designs in this thesis incorporate three metal layers and two insulation layers in between them. The best way to align multiple layers is to have all the layers align to one single layer. Otherwise errors can cascade aligning one to layer to the next, then another to the next and so forth. Figure 3-7 shows the pattern that was etched into the silicon. The pattern is shown in dark green. Black is the background. Cross boxes for size 1, 2.5, 5, 7.5, 10, 25 and 50 µm were used as well as vertical and horizontal verniers. There are many cross boxes of different sizes because one usually aligns bigger features first and then slowly refines to smaller and smaller features.

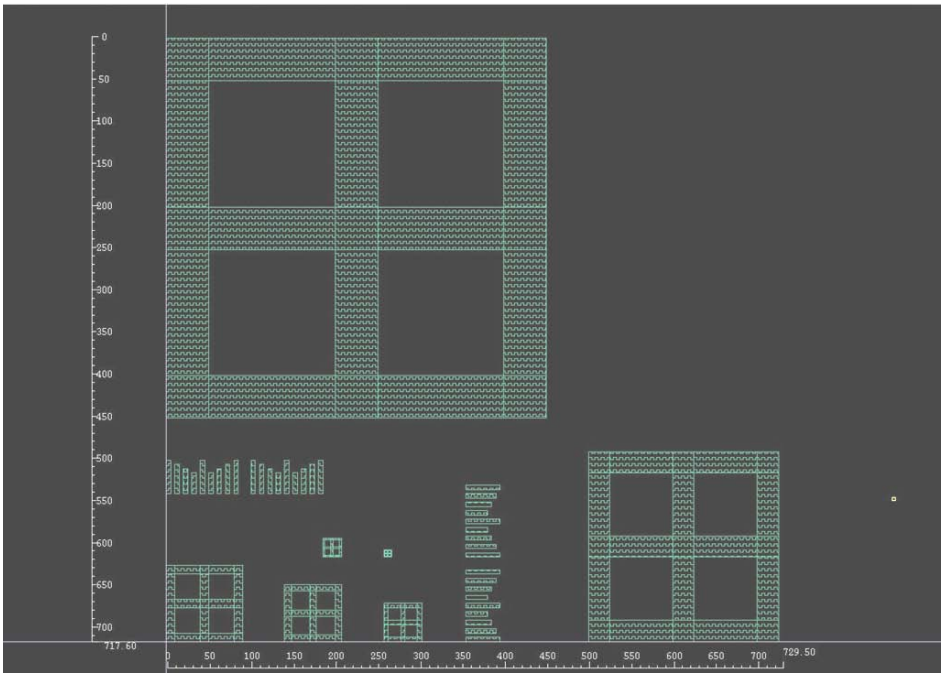


Figure 3-7: Alignment marks etched into silicon shown in dark green. The etch marks included crossboxes of size 1, 2.5, 5, 7.5, 10, 25 and 50 µm and horizontal and vertical verniers.

In the following figure, figure 3-8, it's shown how metal layer 1 aligns to the marks etched in silicon. The marks etched in silicon are in dark green and the marks that are patterned for metal 1 are shown in dark blue.

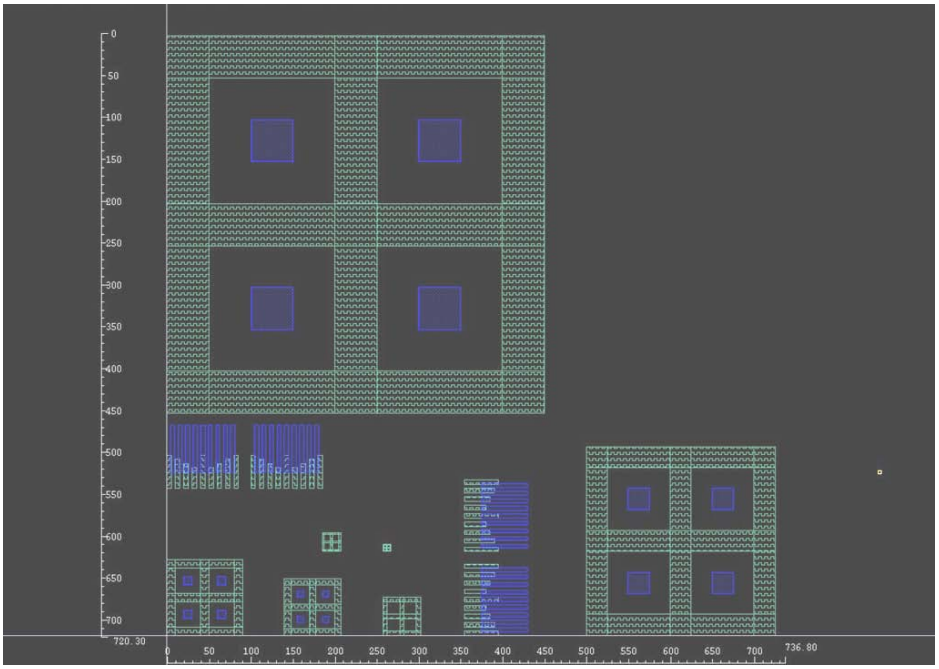


Figure 3-8: Metal1 layer shown in dark blue aligns to the etched marks in silicon shown in dark green.

Verniers are especially useful because they quickly visually indicate if a misalignment occurs and if so by how much. A better view of the vernier marks is given in figure 3-9. Again, dark green marks are etched in silicon and dark blue marks are aligned afterwards when patterning metal 1.

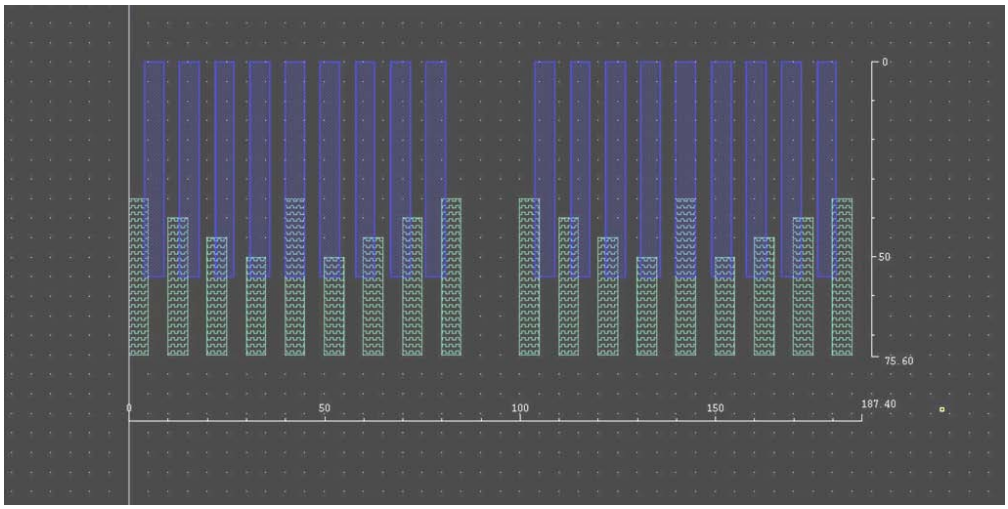


Figure 3-9: Vernier marks, dark green silicon etch, dark blue metal 1.

Figure 3-10 below are all five masks' alignment marks to the etched design. From left to right it is metal1 (dark blue), via12 (light blue), metal2 (yellow), via23 (lime green) and lastly metal3 (orange). The order of the alignment marks is also the order that materials were deposited and patterned.

Figure 3-10: Full alignment mark set for metal1 (dark blue), via12 (light blue), metal2 (yellow), via23 (lime green) and metal3 (orange).

The Engineering Lion and University of Manitoba logos were patterned using metal layer 3 and are shown in more detail in figure 3-11.

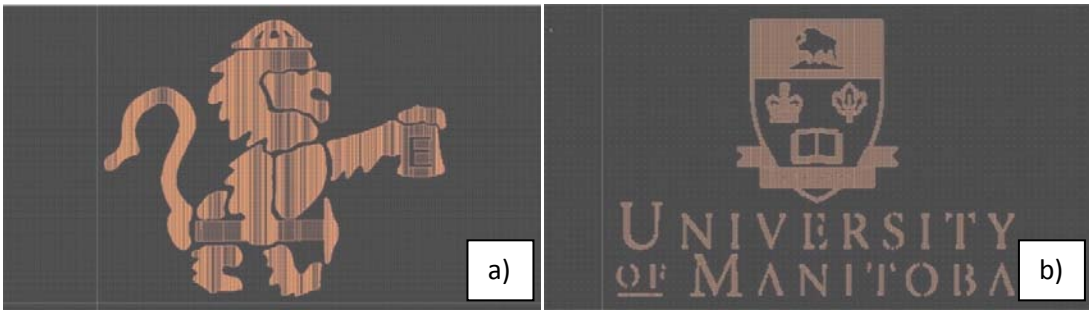
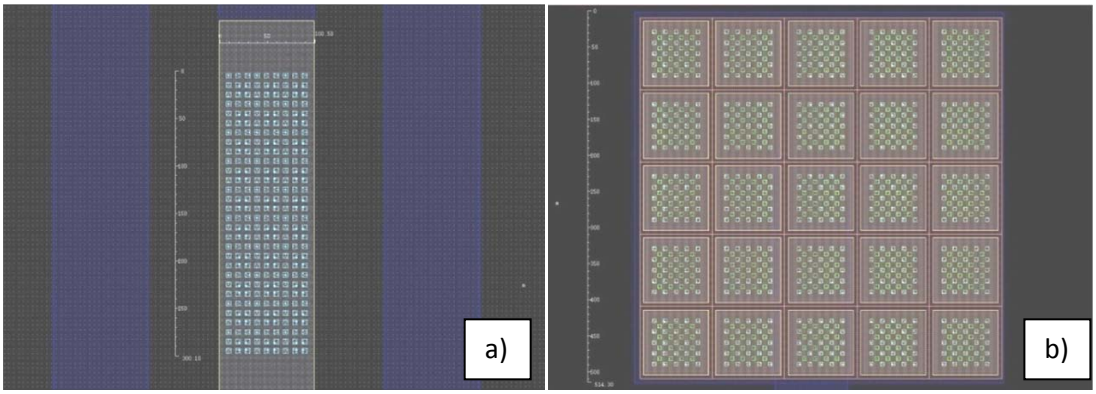


Figure 3-11 a) Engineering Lion patterned in metal3 b) U of M logo also patterned in metal3.

Aligning the via masks is especially difficult because they are patterned with a negative mask. The majority of the mask is opaque except for a few micron sized squares that act as perforations in the fully covered layer of oxide. The oxide is perforated to maintain its structural integrity; otherwise bows and craters could develop. Through the vias, one metal layer can be connected to another metal layer as needed in the design. Figure 3-12 shows two uses for vias. In the first mask image below, metal1 (dark blue) and metal2 (yellow) are connected with via12 (light blue). In the second mask image, all five mask layers are connected in order to create a contact pad. It's difficult to separate but metal1 is shown in dark blue, via12 is shown in light blue, metal2 is shown in yellow, via23 is shown in lime green and lastly metal2 is shown in orange.



Figures 3-12 a) via12 connects metal1 and metal2 b) All five layers connected with the help of via12 and via23.

The odd shape shown in figure 3-13 was used to align the vias to the original etch into silicon. The dark green is the pattern to be etched into silicon. The light blue is via12 and the lime green is via23.

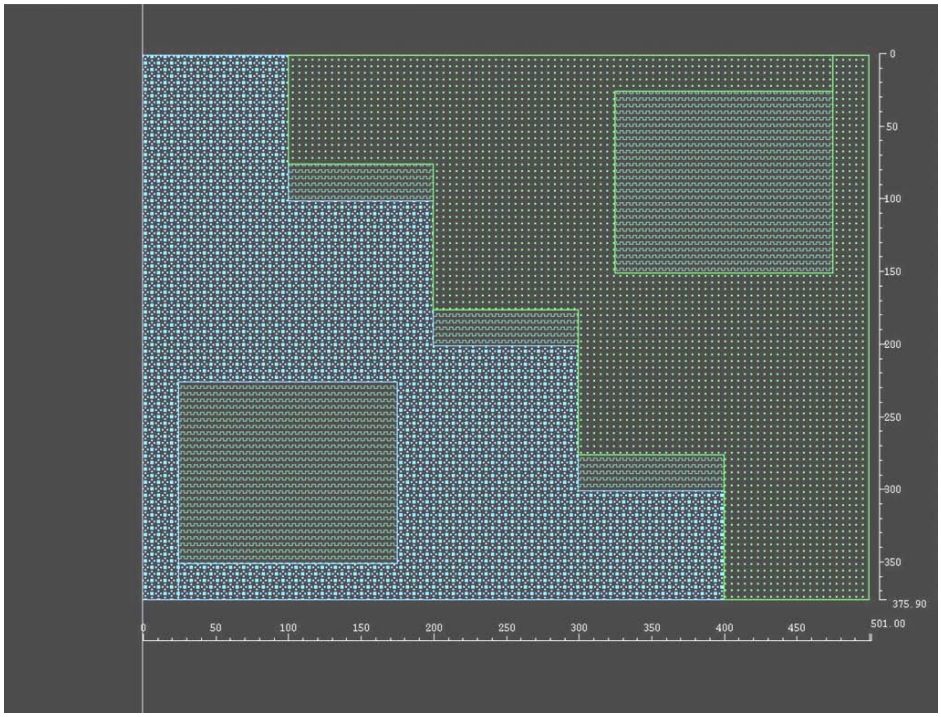


Figure 3-13: Gross alignment marks for via12 and via 23.

3.3: First Design

Now that the alignment marks have been explained, the first sensor design will be discussed. The fabrication process begins with a deep etch into silicon to create the cavity where the fluids and particles will be.

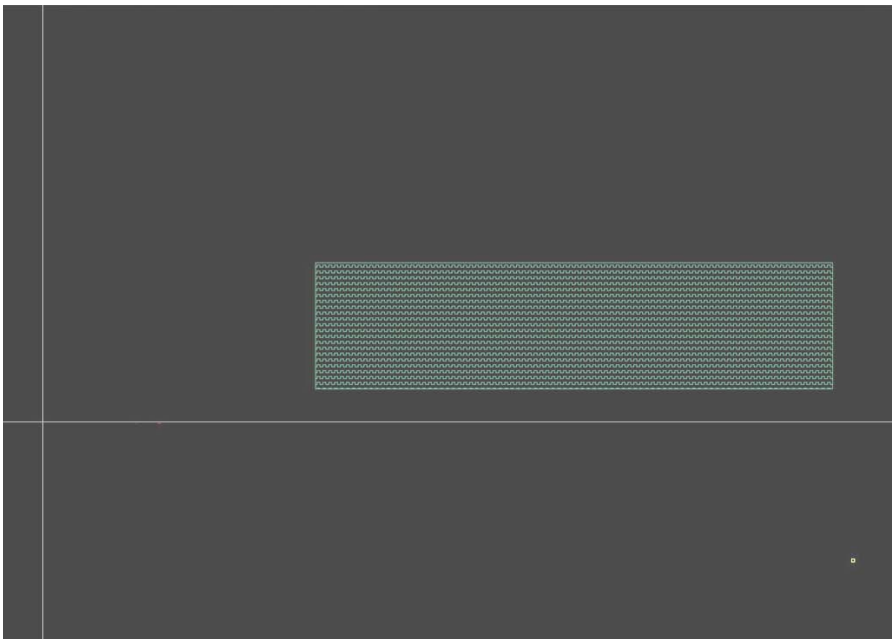


Figure 3-14: Mask for the deep silicon etch in the first design. The green rectangular area represents that cavity created to hold the fluid and particles.

Initially, the alignment marks were etched into the silicon at the same time as the deep trench but that became problematic. In order to create a deep trench of nearly 60 µm depth, the silicon wafer remained in a 80 ^{0}C 50 % KOH bath for nearly 80 minutes. As one can see from the images in figure 3-15 extended exposure in the KOH bath makes the alignment marks indiscernible. The first set was etched for 5 minutes, the second for 20 minutes and the last set for 60 minutes.

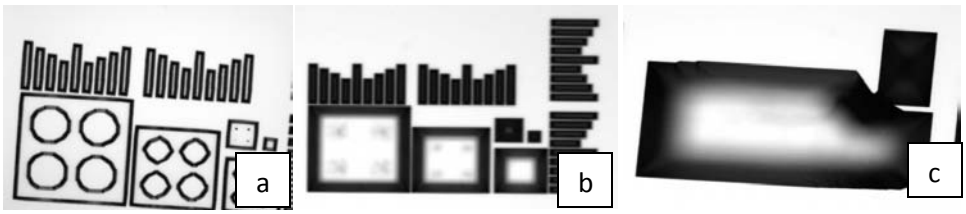


Figure 3-15: Alignment marks etched in 50% KOH bath for a) 5 min. b) 20 min. c) 60 min. Even after 5 min., the features of the alignment marks are distorted.

The etches were broken down into a long etch for the deep trenches and a short 1 min. etch for the alignment marks. That processing method ensured a deep etch for the trenches and functional alignment marks. The deep etch into Silicon (100) also made the wafer very fragile. The remedy was processing the wafers a few degrees off axis to avoid the cleaving planes.

In the next step, metal1 was deposited and patterned using the mask shown in figure 3-16. Dark blue is for metal 1. Dark green is for the initial deep silicon trench.

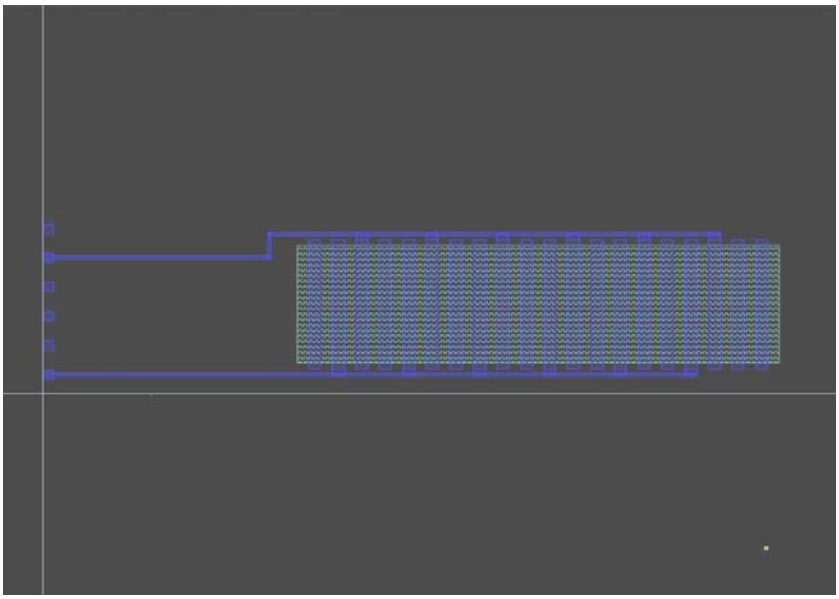


Figure 3-16: Metal 1 mask for first design. Metal 1 in dark blue, deep silicon etch in dark green.

Then the insulating oxide separating metal 1 and metal2 was deposited. The selected areas where metal1 and metal2 needed to be connected was patterned and etched and via12 was formed. Following the oxide patterning, metal2 (shown in yellow) was deposited and patterned as shown in figure 3-17.

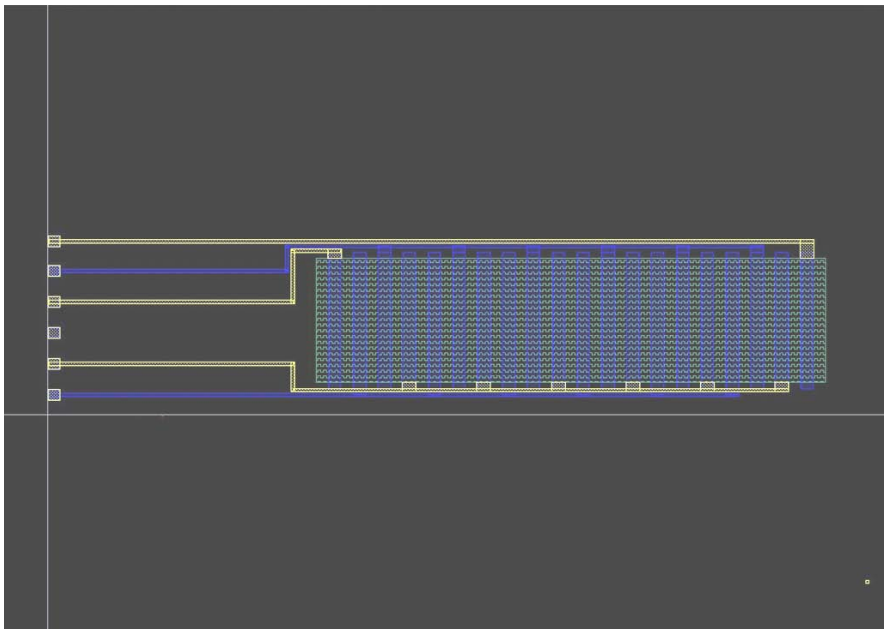


Figure 3-17: Metal 2 mask for first design. Metal 2 shown in yellow.

Then the oxide layer separating metal2 and metal 3 was deposited. The areas in which metal2 and metal3 needed to connect were etched to form via23. After that, metal3 was deposited and patterned. Metal3 provided the metal frame around the trench for the conductive lid to be placed. It was also used for all the contacts. Figure 3-18 shows the metal3 mask.

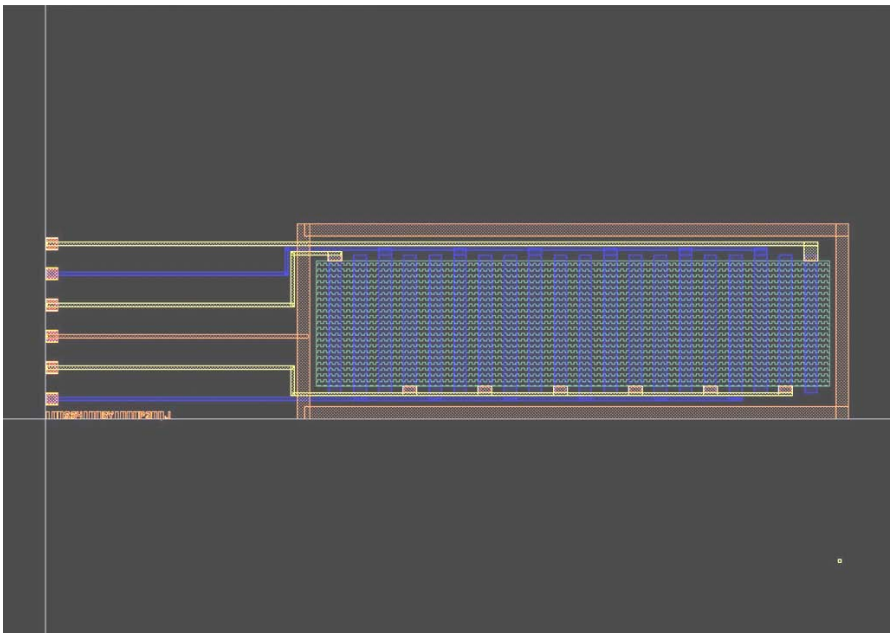


Figure 3-18: Metal 3 mask for first design. M3 shown in orange.

There were many problems with the first design. Choosing to etch so far into silicon was very challenging. Metal lines failed to adhere to the inclined slope and as such produced “opens” where there was limited or no conductivity.

Figure 3-19: Disconnected metal lines. Lines meant to follow the incline of the etched trench released and fell inside the trench.

It was also found that the liquid photoresist (that was used in the photolithography process to pattern the metals) pooled in the edges of the trench when spun. This nonuniformity of photoresist thickness made it difficult to pattern the electrodes inside the trench. The etched trench in figure 3-20 outlined in dark black was exposed for close to a minute. Everything inside the trench was meant to be removed but because the photoresist gathered in the corner while being spun, it was impossible to uniformly expose and develop the desired features. Even after 55

seconds of exposure and development persistent undesired features emerged inside the trenches like below.

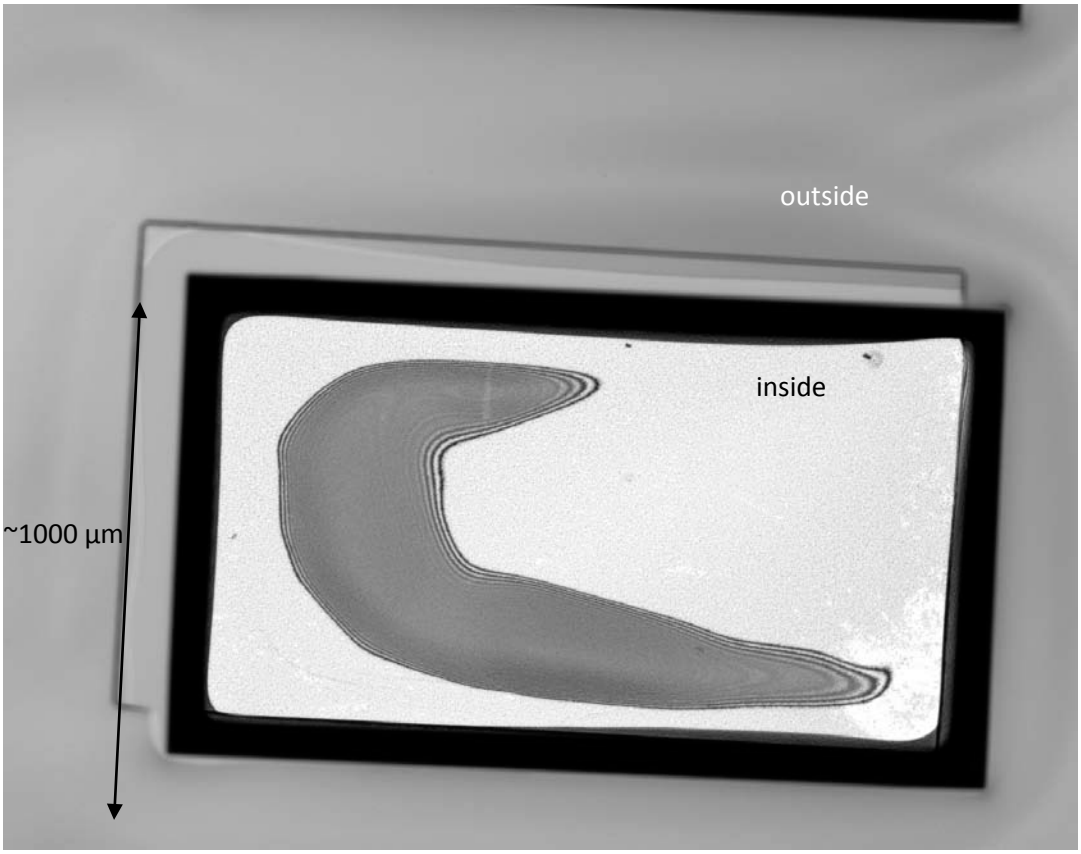


Figure 3-20: Photoresist pooling inside the trench

Until now, the discussion has focused on the problems with one device, but there were also significant problems with the entire mask set. Devices were grouped together such that multiple sensors could be tested at once. Liquid and particles were placed in an array of trenches and then the conductive lid was glued on top using conductive glue called CrystalBond™. In order to reuse the sensors, acetone was used to release the glue that held the lid. Unfortunately acetone

proved to be very destructive it released the patterned metals and destroyed a group of sensors instead of just one. There was no reproducibility in this mask set either. Many variations of the sensor were printed with varying electrode lengths, widths and pitches. Even the alignment marks were not periodically spaced apart. A matrix of alignment marks in the x and y direction is ideal because after one alignment is made, scrolling in the x and y direction aligns the rest of the wafer. The entire 4 inch mask including all devices and metal/insulation masks is shown in figure 3-21.

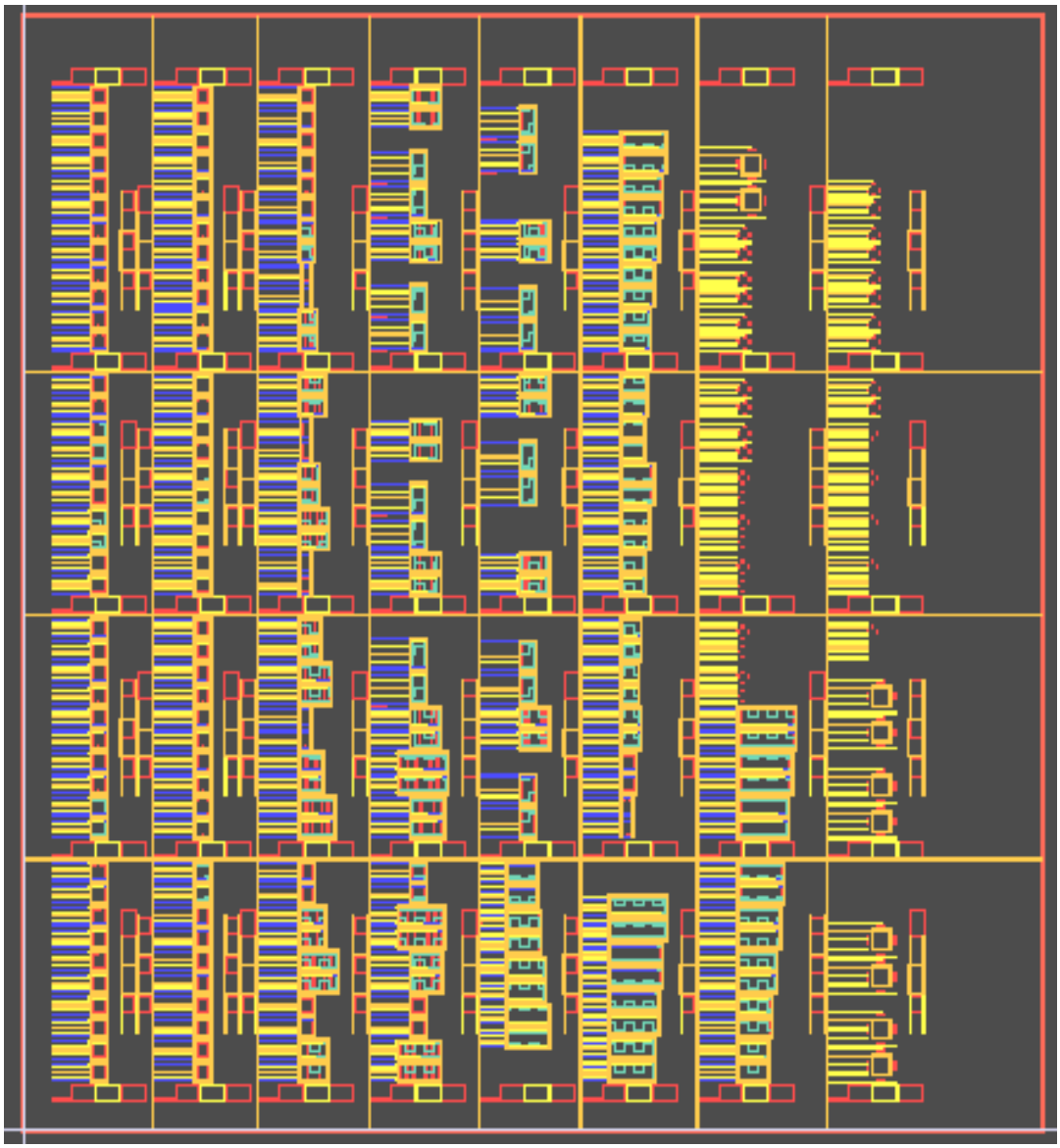


Figure 3-21: Four inch mask set of first design. Notice the lack of reproducibility. There is one chance to make each device.

It was assumed that the wafer holder in the mask aligner system would have full mobility and allow a wafer to reach all corners of the mask but that was not the case. A lot of the mask was not usable because the stage that held the 3 inch silicon

wafer could only move a few centimetres from the center and did not allow corners to be printed. This is shown in figure 3-22. The dashed lines represent how far from the center the wafer could be positioned.

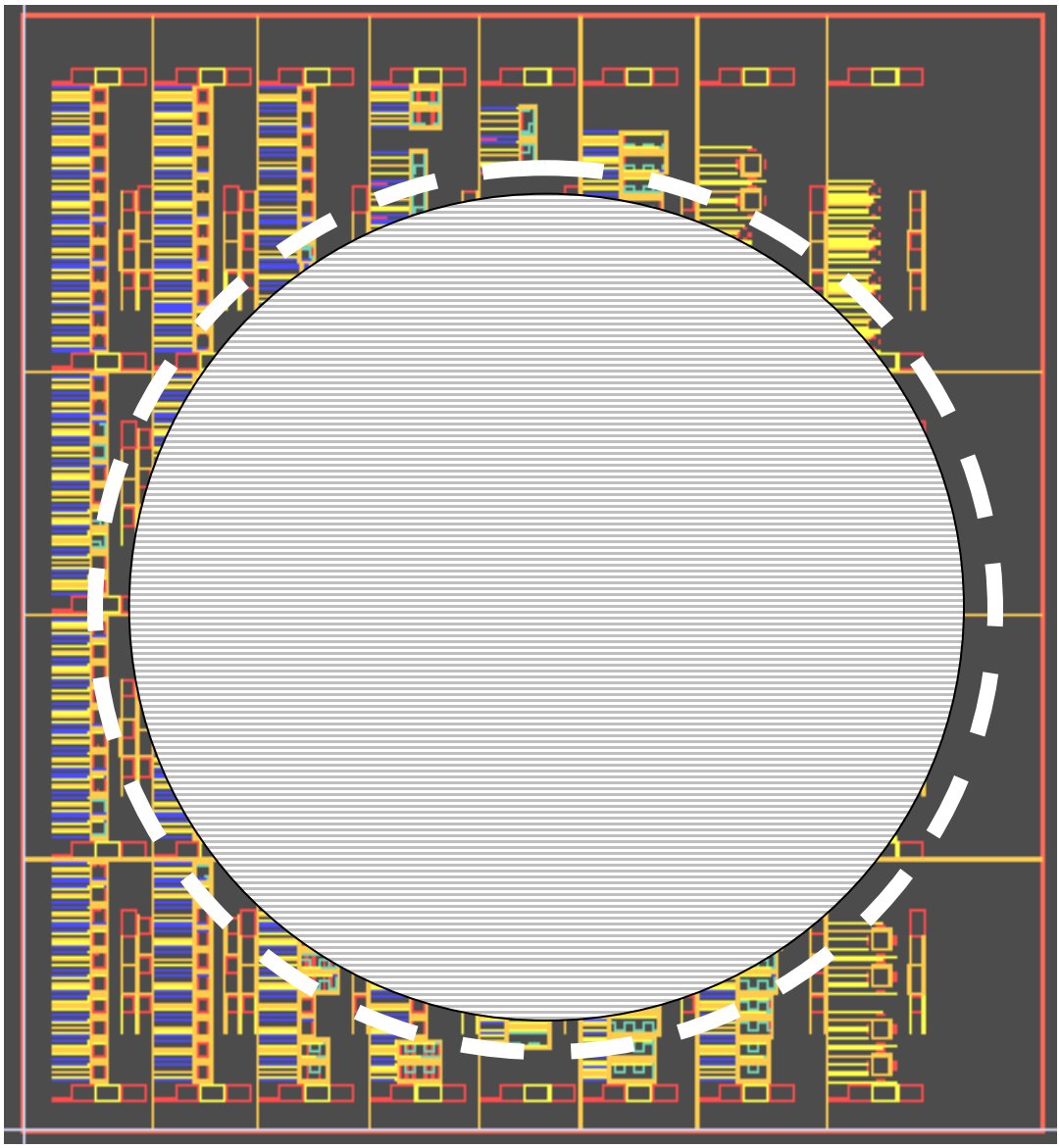


Figure 3-22: Four inch mask set of first design with three inch silicon wafer in the middle. Dashed line shows how far from the center the wafer holder allows movement of the three inch wafer. The wafer stage cannot move the wafer to the mask corners or very far from the center so many devices could not be made.

3.4: The Second Design

For the various reasons discussed above, the first design was not entirely successful. The second design fared significantly better and is discussed in more detail below. In order to avoid many of the problems caused by the extremely deep etch into the silicon, the conductive lid was instead milled to produce the cavity necessary for the particles and fluid. Therefore, the deep etch into silicon was avoided all together. For repeatability of use, inlet and outlet ports were used to avoid releasing the lid for reusability. The milled lid with inlet and outlet ports is shown in figure 3-23.

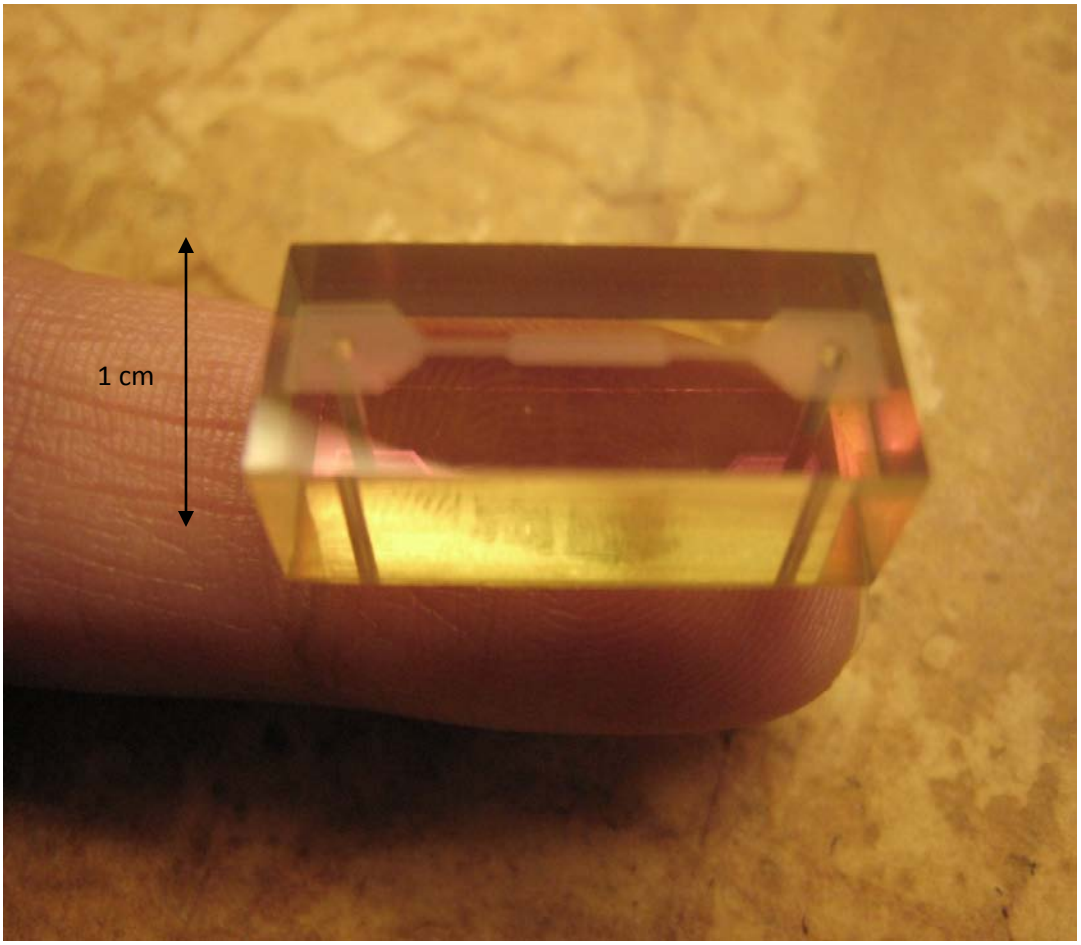


Figure 3-23: Milled conductive clear lid with inlet and outlet ports.

In the second design, an oxidized silicon wafer was used to provide electrical insulation between neighbouring electrodes. A short buffered oxide etch (BOE) was used to pattern the surface oxide with alignment marks and the general shape of the device as shown in figure 3-24. The openings at each end are for inlet and outlet ports.

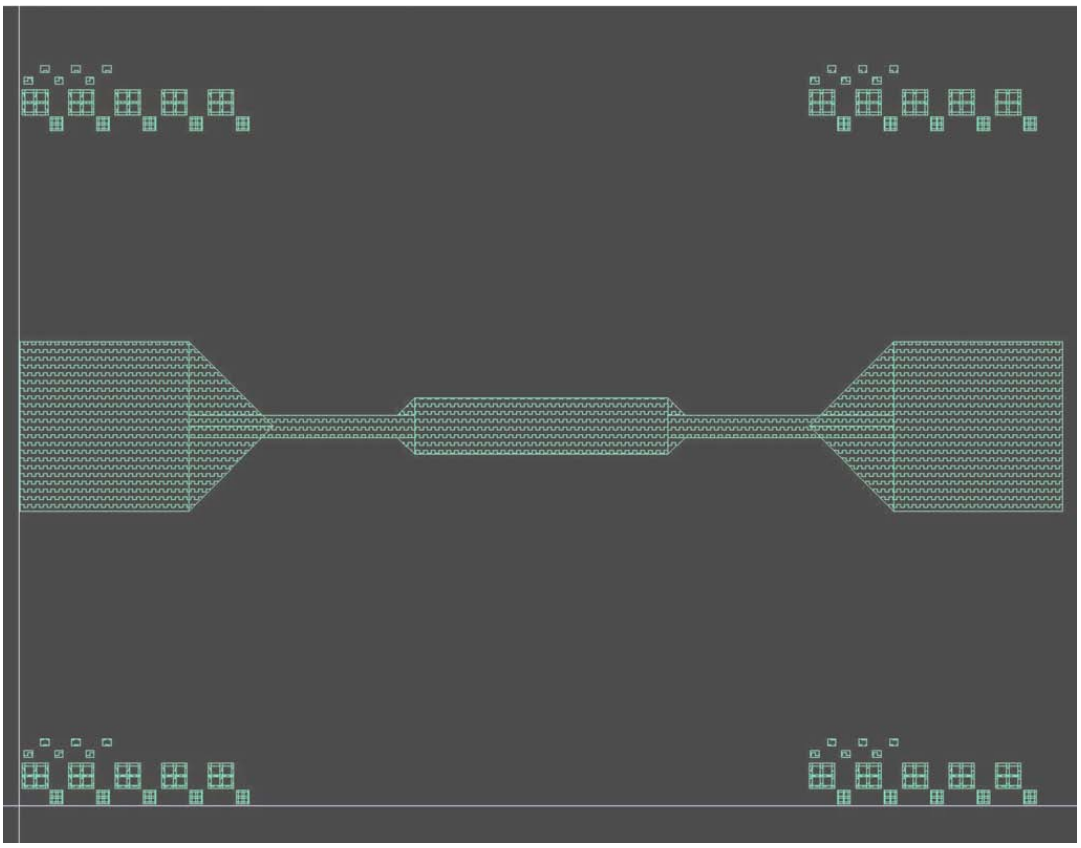


Figure 3-24: Mask for short BOE etch for the second design.

After the short BOE etch, to pattern the wafer surface oxide, metal 1 was deposited and patterned. The mask used is shown in figure 3-25.

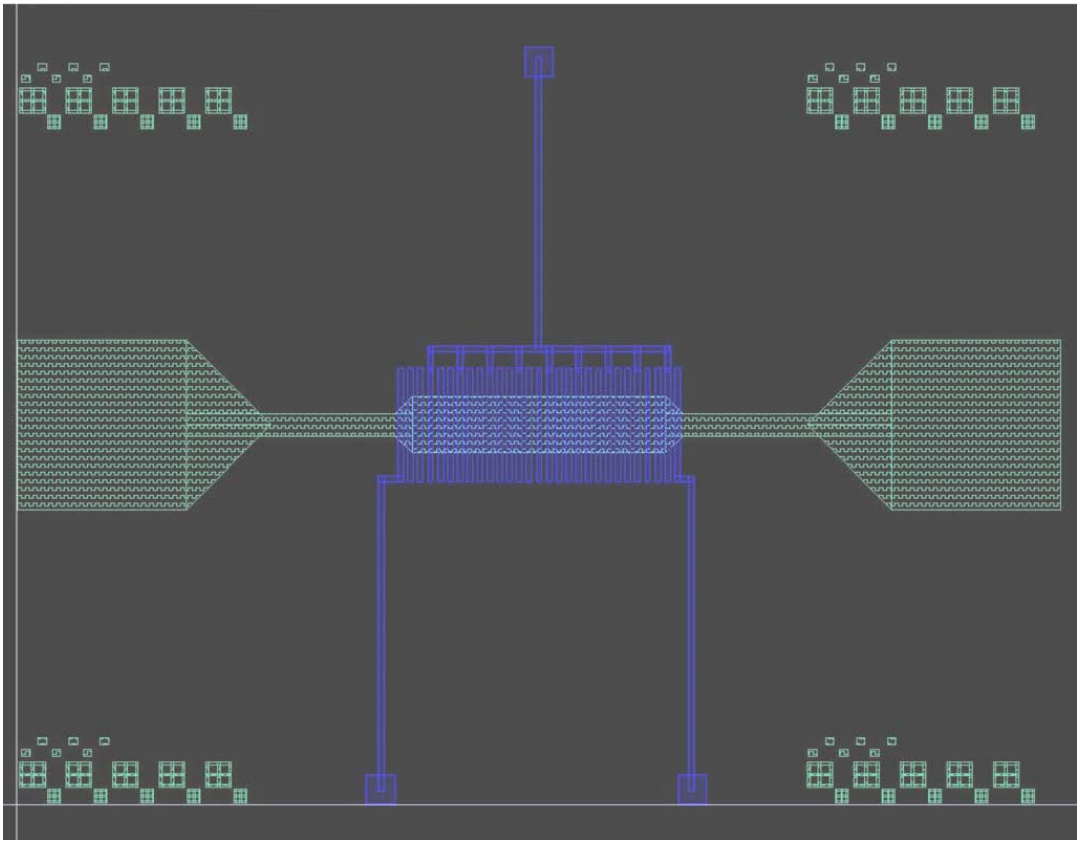


Figure 3-25: Metal1 shown in dark blue in second design

The insulating oxide was then sputtered and via12 was patterned. Following which metal2 (shown in yellow in figure 3-26) was sputtered and patterned.

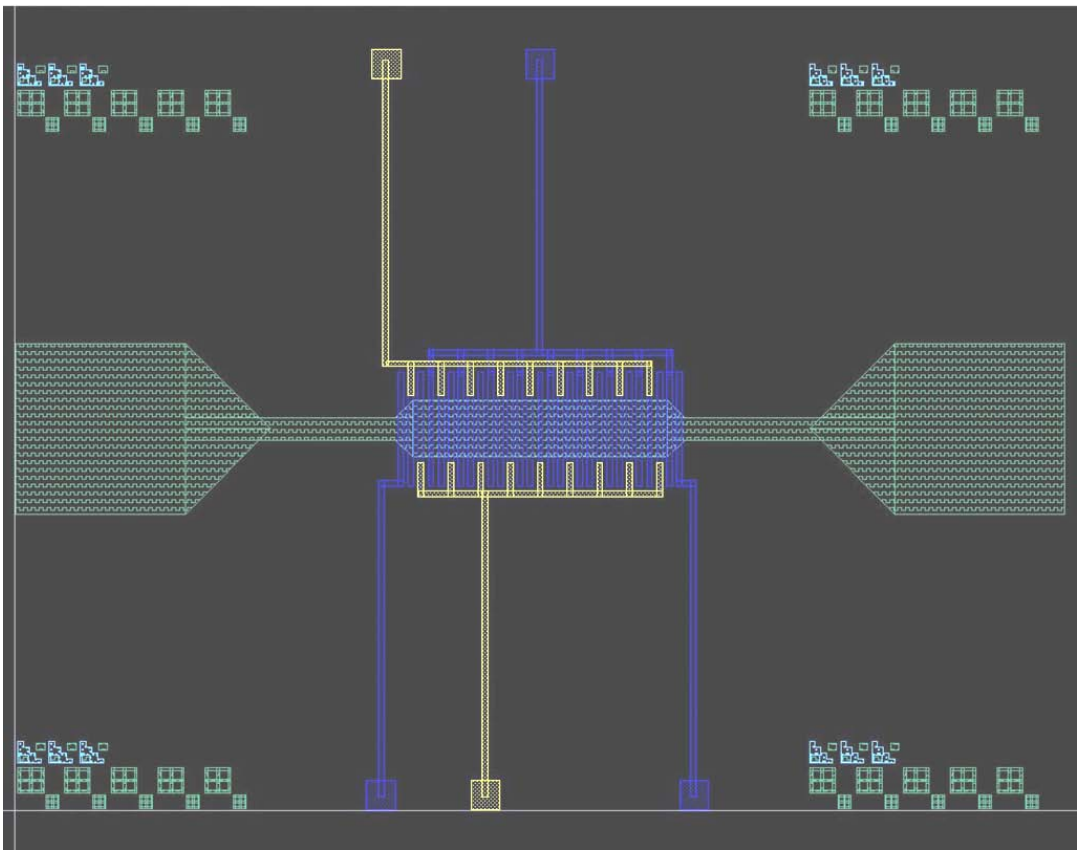


Figure 3-26: Metal2 shown in yellow in the second design.

A second layer of oxide was then sputtered over the metal 2 layer, via23 was patterned and metal3 (shown in orange) was sputtered and patterned. In the first design, a box was patterned around the trench to drive the conductive lid that was place over it. In the second design, there was not a box surrounding the channel. There were two long lines adjacent to the channel on both sides that the lid was glued on top of and driven. In the first design, all the probe pads were located on one side of the device. In the second design probe pads were divided up on both

sides of the device, that way the probes wouldn't be as crowded as the previous design.

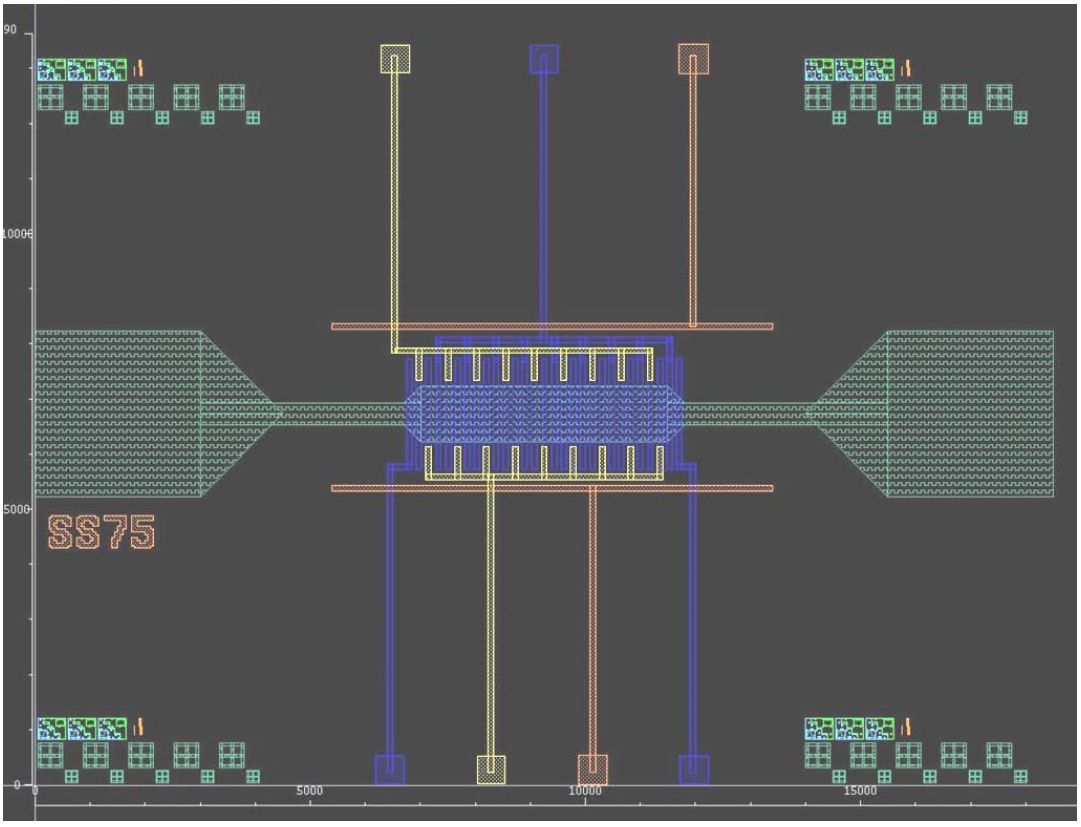


Figure 3-27: Metal3 shown in orange in the second design.

In the last step all off the oxide covering metal1 in the channel was removed. The successive layers of oxide were deposited to insulate between metal1 /metal2 and metal2/metal3 interfaces. However inside the microfluidic channel only metal1 was patterned. There was no need to have metal1 in the channel covered by the oxide. In fact, metal1 needed to be exposed to permeate the fields necessary to

trap and manipulate the particles. Metal1 needed to be opened up and was done so by the etch mask used in the channel as shown in figure 3-28.

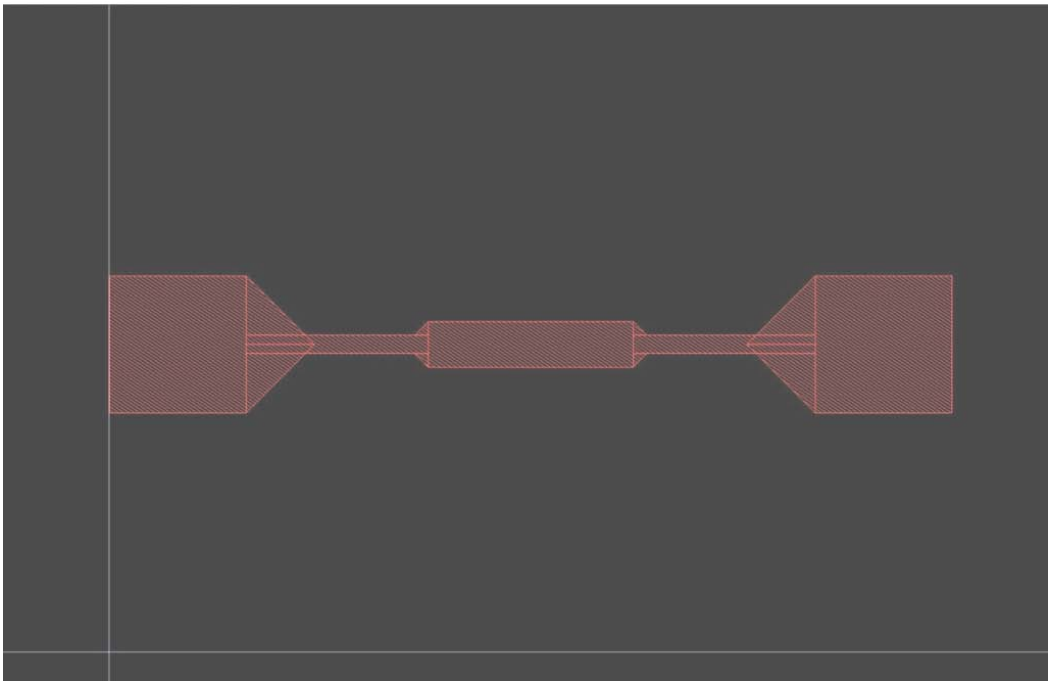


Figure 3-28: BOE mask used in the channel to expose metal1 patterned lines.

Improvements were not only made on individual devices. Many changes were made on the entire 4 inch mask set that greatly improved fabrication processes. For example, gross alignment boxes were put in the corners of the 4 inch mask set at every mask design level. These alignment boxes were not meant to be printed on a 3 inch wafer, they were outside the reach of the wafer stage. The inside corners of each of the four boxes were meant to just touch the outside perimeter of a 3 inch wafer. These strategically placed boxes enabled quick centering of 3 inch wafers. After that minor alignment and rotation was checked,

the wafer could be exposed after that. This greatly reduced the time needed to pattern the wafer. Furthermore, devices and alignment marks were organized to allow for repeatability on the same wafer. There was more than one opportunity to print a device on the same wafer. Alignment marks were found strategically throughout the wafer such that if one alignment mark was set scrolling up or down would align another set. The entire mask 4 inch mask with its many improvements is shown in figure 3-29.

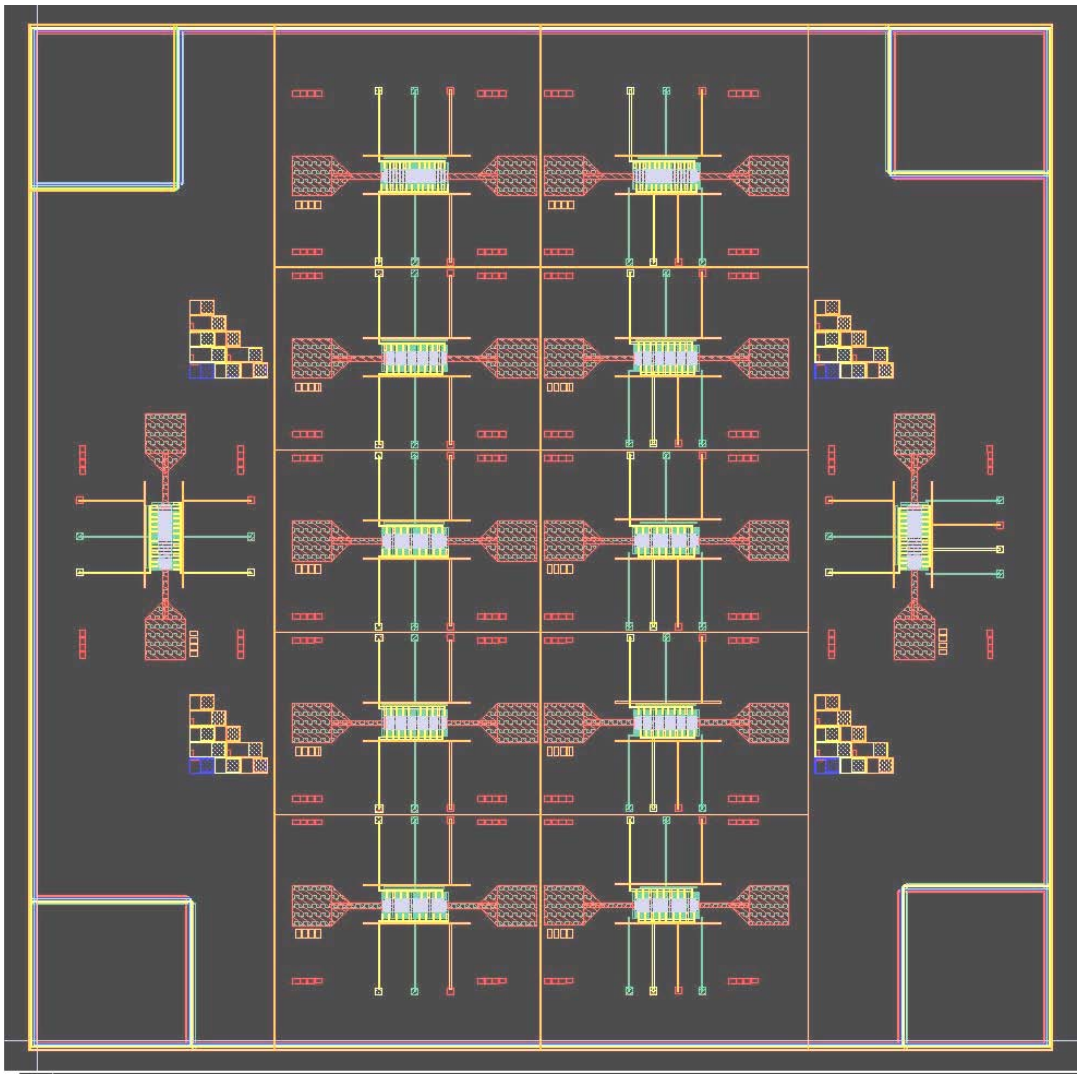


Figure 3-29: Entire mask set for second design.

After the entire wafer was been fabricated, individual sensors were cut using a wafer saw. A conductive, optically transparent layer of Indium-tin-oxide (ITO) was deposited onto the milled lid. The milled lid was glued to the silicon as shown below after which micropipes were attached to allow for fluid input.

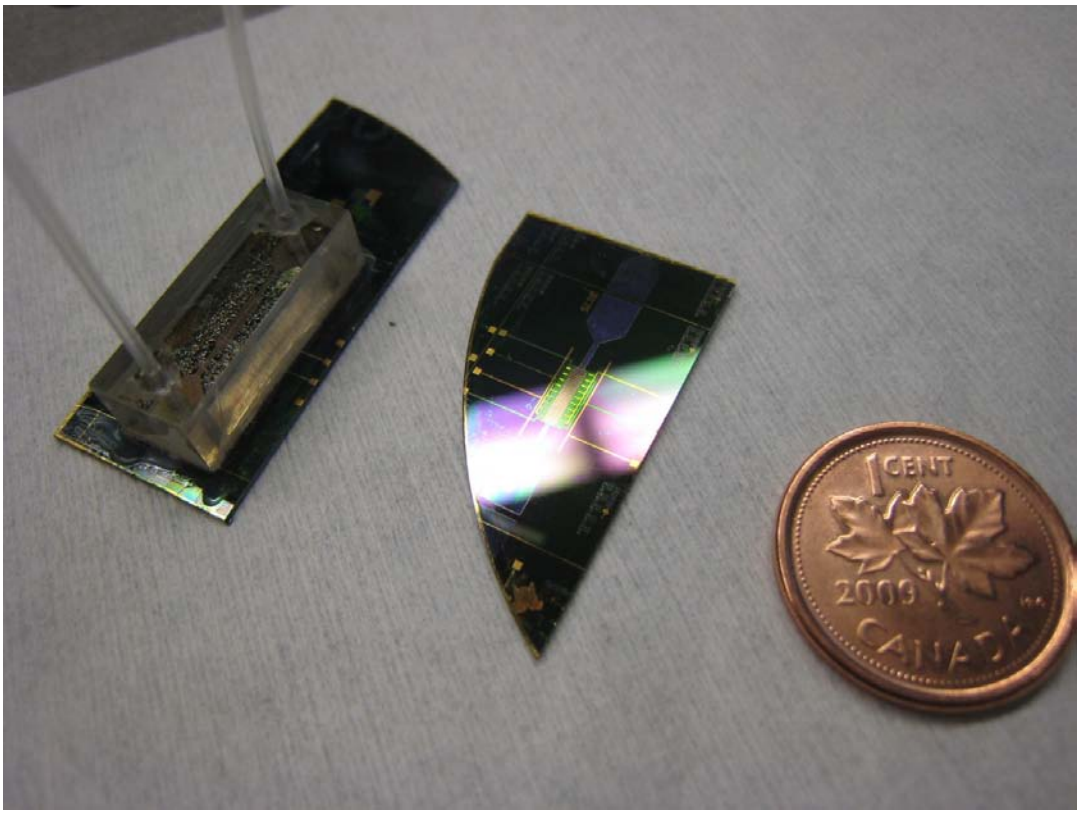


Figure 3-30: Fully assembled device on the left, patterned wafer in the middle and a penny on the right to show scale.

3.5: Additional Fabrication Details

Since Plasma Enhanced Chemical Vapour Deposition (PECVD) was not available, other methods to deposit low temperature oxide needed to be developed. Low temperature oxide was necessary because metal layers would be compromised if exposed to a high heat source. Through trial and error, a sputtered oxide recipe was found:

Power	Gases	Sputter Duration	Pressure	Thickness
300 W RF	56 sccm O_2, 37 sccm Argon	50 minutes	$1.5e^{-3}$mT	≈ 300 nm

Table 3-1: Oxide sputter recipe. "sccm" = standard cubic centimetres per minute

The recipe produced a very good insulator. Six current-voltage measurements were done on half of a 3 inch wafer sample in the positions shown in figure 3-31 in order to determine oxide quality. The capacitors measured were approximately $1x10^{-5}$ cm^2 in area.

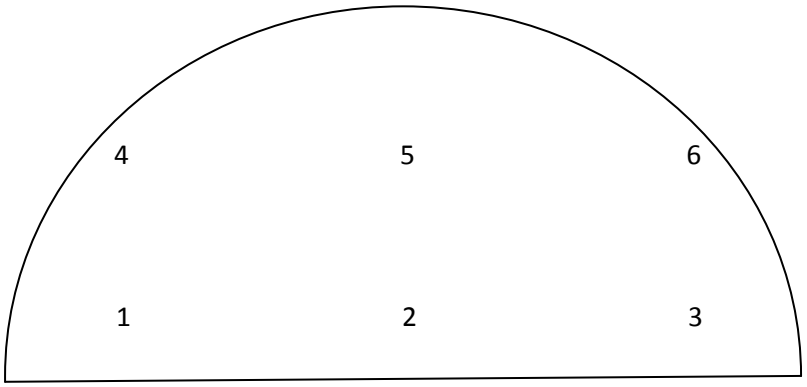


Figure 3-31: Six positions on a 3 inch wafer where current-voltage measurements were taken after having low temperature oxide sputtered

Figure 3-32 shows that all six positions had very good insulating properties. Voltage was applied at gradually increasing values and the current was measured and recorded. For applied potentials as high as -100V, the capacitor leakage current did not exceed 0.1 pA. The thickness of the oxide tested was roughly 2790 Å.

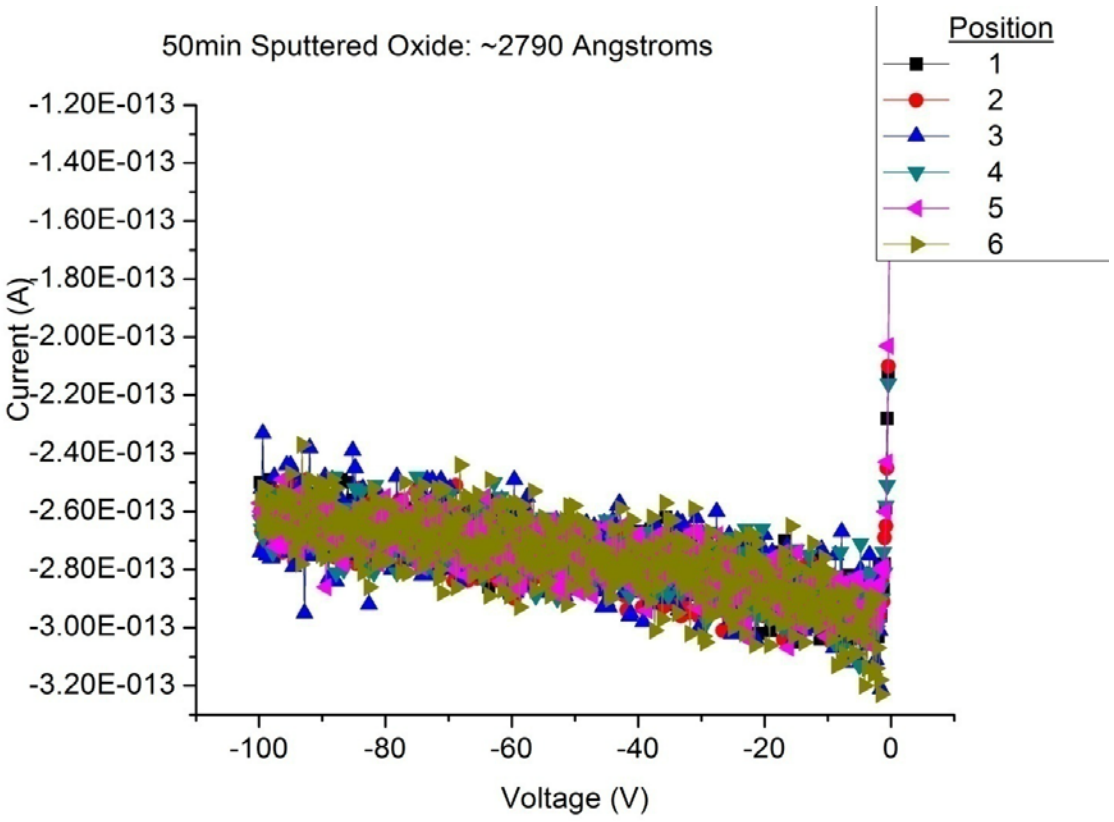


Figure 3-32: Current-Voltage measurements for a 50 min sputtered oxide sample. The oxide remains a good insulator up to -100V.

Sputtering for a short time like a minute or two guarantees fairly uniform deposition rates. Unfortunately, long sputtering runs, like the 50 minute oxide depositions needed in this thesis, do not have uniform rates. The highest deposition rates are found directly under

the target. The rates are lower further from the center of the target. There was a 10% oxide thickness variation from the center of the wafer to the edge. An attempt to have uniform oxide sputtering coverage was made by continuously rotating the sputter stage. This method did not produce quality oxide; in fact the samples that were rotated had pinholes.

Since BOE was needed to pattern the oxide, the metals chosen in the fabrication process would need to be resistant to it. This is especially true for this design because the deposition rate of the sputtered oxide was not uniform. The oxide layer had to be etched for the thickest part in the center. Metals at the edge of the wafer, where the oxide would be over-etched, were vulnerable. Gold was used because it is inert and was less likely to stick to the particles being actuated in the microfluidic channel. In order to deposit gold an adhesion layer is needed. Two choices were titanium, which is sensitive to BOE and chrome which is not. Therefore chrome was used. Below are the recipes used for depositing chrome and gold:

Power	Gases	Sputter Duration	Pressure	Thickness
200 W DC	41 sccm Argon	2 minutes	$5e^{-3}$mT	≈ 140 nm

Table 3-2: Sputter recipe for Chrome

Power	Gases	Sputter Duration	Pressure	Thickness
300 W DC	41 sccm Argon	1 minute	$5e^{-3}$mT	≈ 240 nm

Table 3-3: Sputter recipe for Gold

The robustness of the recipes was tested using the tape test. After the deposition, a piece of scotch tape was stuck on to the surface and then pulled off quickly. The metal remained on the wafer which meant the adhesion layer worked.

[1] G. Medoro, N. Manaresi, A. Leonardi, L. Altomare, M. Tartagni, R. Guerrieri, "A Lab-on-a-Chip for Cell Detection and Manipulation", IEEE Sensors Journal, Vol 3, No 3, pg 317-325 June 2003

[2] A. Vulto, G. Medoro, L. Altomare, G A Urban, M. Tartagni, R. Guerrieri, N. Manaresi, "Selective Sample Recovery of DEP-Separated Cells and Particles by Phaseguide-Controlled Laminar Flow", Journal of Micromechanics and Microengineering, Vol 16, pp 1847-1853, 2006

Chapter 4: Experimental Apparatus

This chapter outlines the configuration of the experimental apparatus used for driving the dielectrophoretic cage device. Two versions of the successfully fabricated design described in chapter 3 are described here. Particles are drawn above a target electrode centrally located in the electrode array in the first version, whereas in second version there are two targets located at the rightmost and leftmost electrodes. In the second version, particles are translated left or right toward the target electrode depending on the chosen sequence of phase shifts. A graphical user interface was developed for automated testing, in which the user decides the type and duration of the phase shift sequence, as well as the frequency, amplitude, and phase of the applied voltage. The operation of the software program is described using a state chart.

4.1: Experimental Apparatus

The devices were tested on the Cascade Microtech Alessa REL-6100 probe station. Cascade Microtech MH2-B micropositioners were used to probe the contacts of the devices, which were driven by signals fed from the Keithley 7001 Switch System. The signals that fed into the switch matrix were generated from the Thurlby Thandar TTi 40 MHz Arbitrary Waveform Generator TGA1244. An

oscilloscope, Tektronix TDS 2024B, verified that correct signals were being provided from the signal generator. A digital camera, Imaging Source DFK 31bf03.h, was used to record the experiments. The arrangement of these instruments is shown in Figure 4-1.

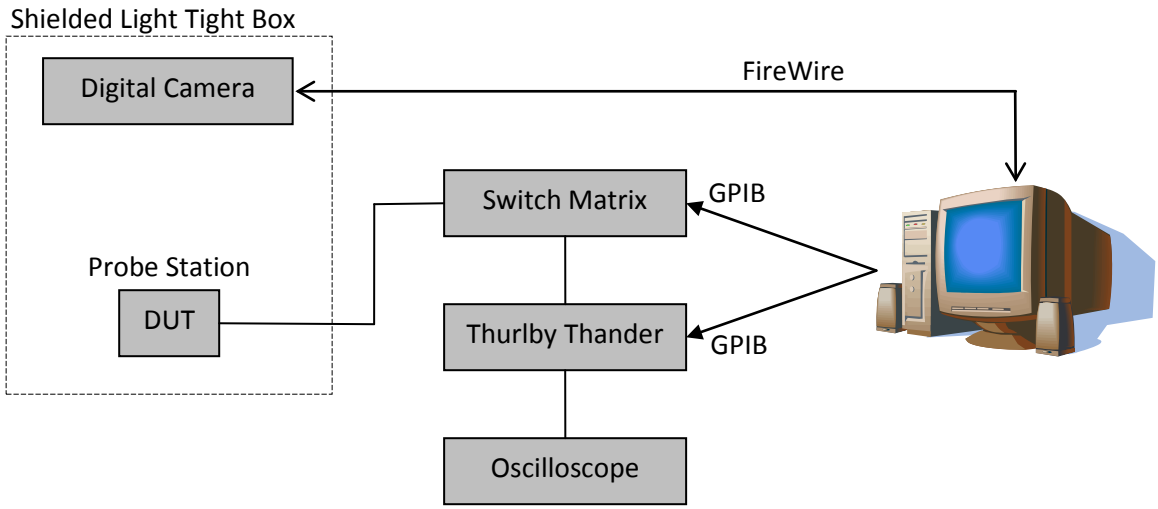


Figure 4-1: Experimental apparatus used for driving the dielectrophoretic cage device.

4.2: Device Contacts

Two variations of the dielectrophoretic cage device were fabricated. The first is shown in figure 4-2, which has six contacts used to drive particles toward a central target electrode, E. Contact C and/or F drive the lid. Contacts B, D and A are connected to the first, second and third nearest electrodes on both sides of central electrode E, respectively. Relative to these positions, each contact (B,D,A) is

additionally connected to every third electrode onward extending in the left- and rightward directions.

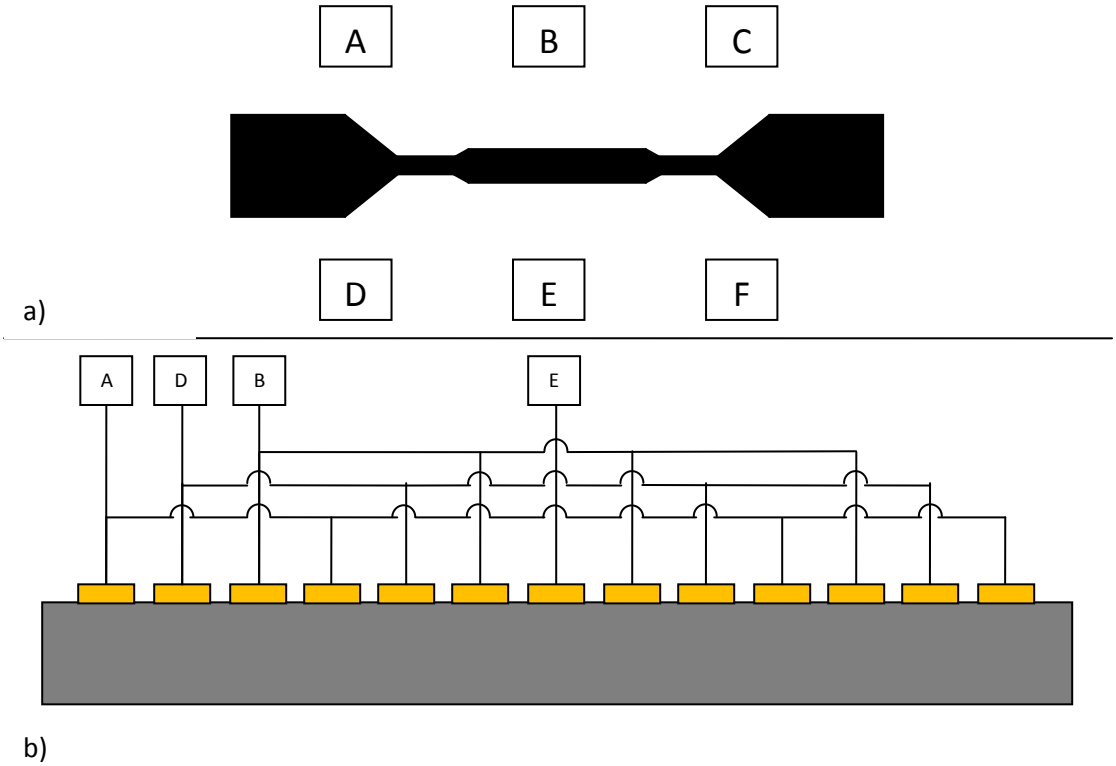


Figure 4-2: In the six contact device, three electrical contacts - A, B, and D - are used to translate particles toward a centralized target electrode, E. The conductive lid is driven using either electrode C or F. a) top view of the device b) arrangement of the electrode interconnections

The second variation moves the particles left or right. As in the first variation, there are two contacts to drive the lid, C and/or F. Contacts D and G are the left- and rightmost electrodes. Contacts A, E, and B are driven to actuate

particles left or right depending on what the user selects in the GUI interface of the computer program used to control the experiment. Figure 4-3 shows the seven contact device.

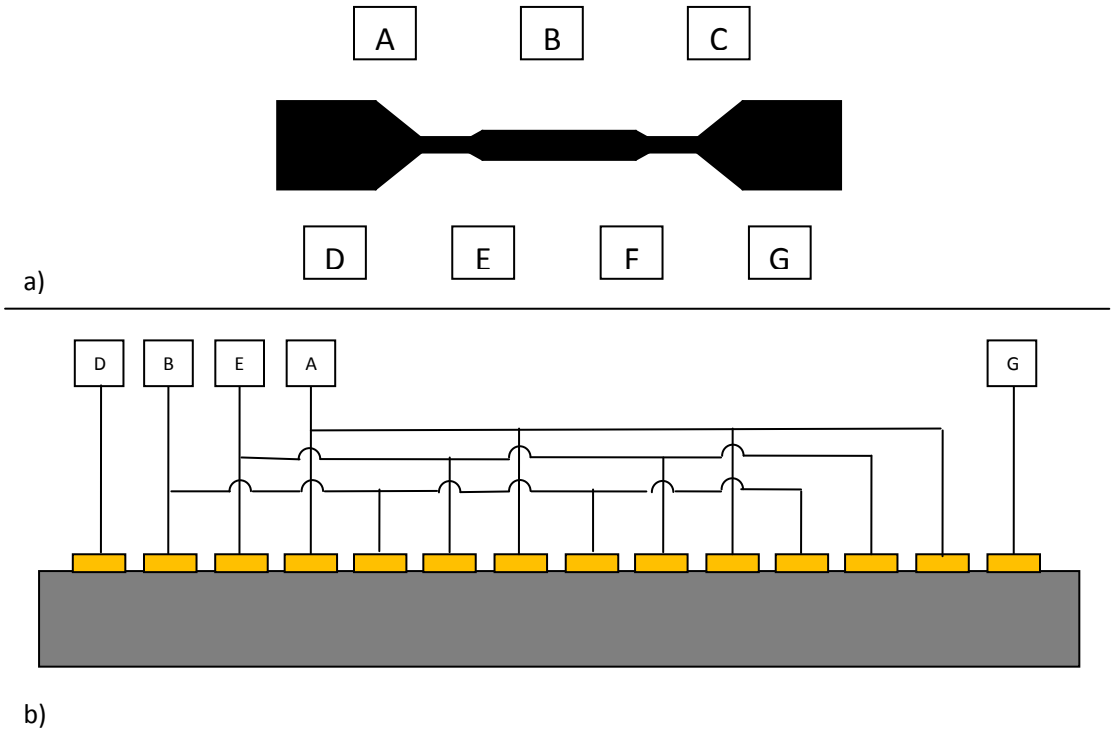


Figure 4-3: In the seven contact device, three electrical contacts - A, B, and E - are used to either translate particles leftward toward target electrode, D, or rightward toward target electrode, G. Again, the conductive lid is driven using either electrode C or F. a) top view of the device b) arrangement of the electrode interconnections

4.3: Software

A software program with a GUI (graphical user interface) was developed using Visual Basic. It allows the user to specify the direction of the particles: left, right or toward the center. It also allows the user to enter the amplitude and frequency of the sinusoidal signals to be applied to the device. The default phase offset is set to 180^0 but can be changed. The user also chooses how often to switch the signals and how long to run the experiment. Depending on the type of experiment chosen, reminders on probe-contact pairings are generated and shown. There is also a start and stop button which commences or ends the program. The GUI is shown below, in figure 4-4.

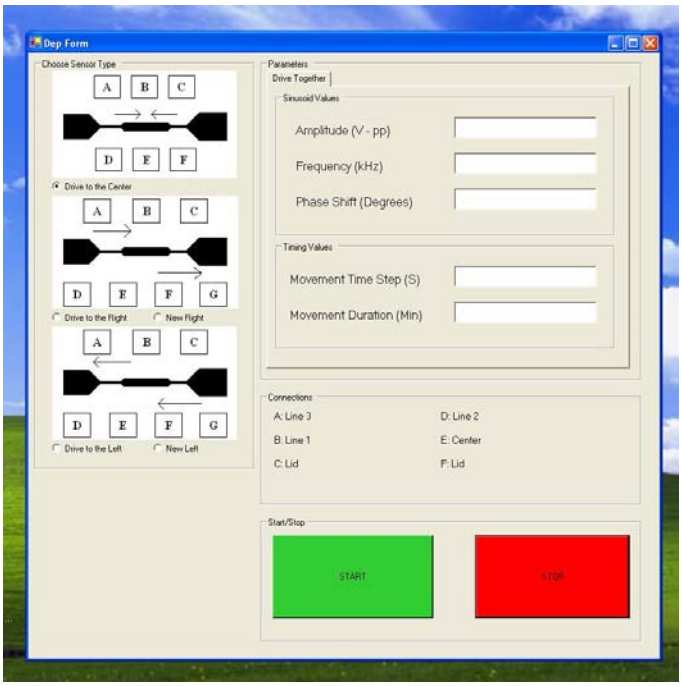


Figure 4-4: GUI of software program that automates testing of devices.

The entire software program can be found in Appendix C. The state chart below helps explain it. Once the start button is pressed, the fields entered are checked for validity. For example, the waveform generator can handle sine waves up to 20 V in amplitude. If the user enters a value over 20 V, the field will discolour and the user must choose another value and hit the start button again. Once the fields are entered correctly, the GPIB connections to the waveform generator and switch matrix are initialized. As soon as the GPIB connections are established, the switch matrix begins switching the signals from the waveform generator to the device. This continues until the time for the entire experiment runs out. After the time runs out, all the connections to and from the matrix are opened and the program is done. The user can also end the program at any time by hitting the stop button. In that case, the matrix will also open all connections and the program will end. Figure 4-5 provides a state chart of the software program.

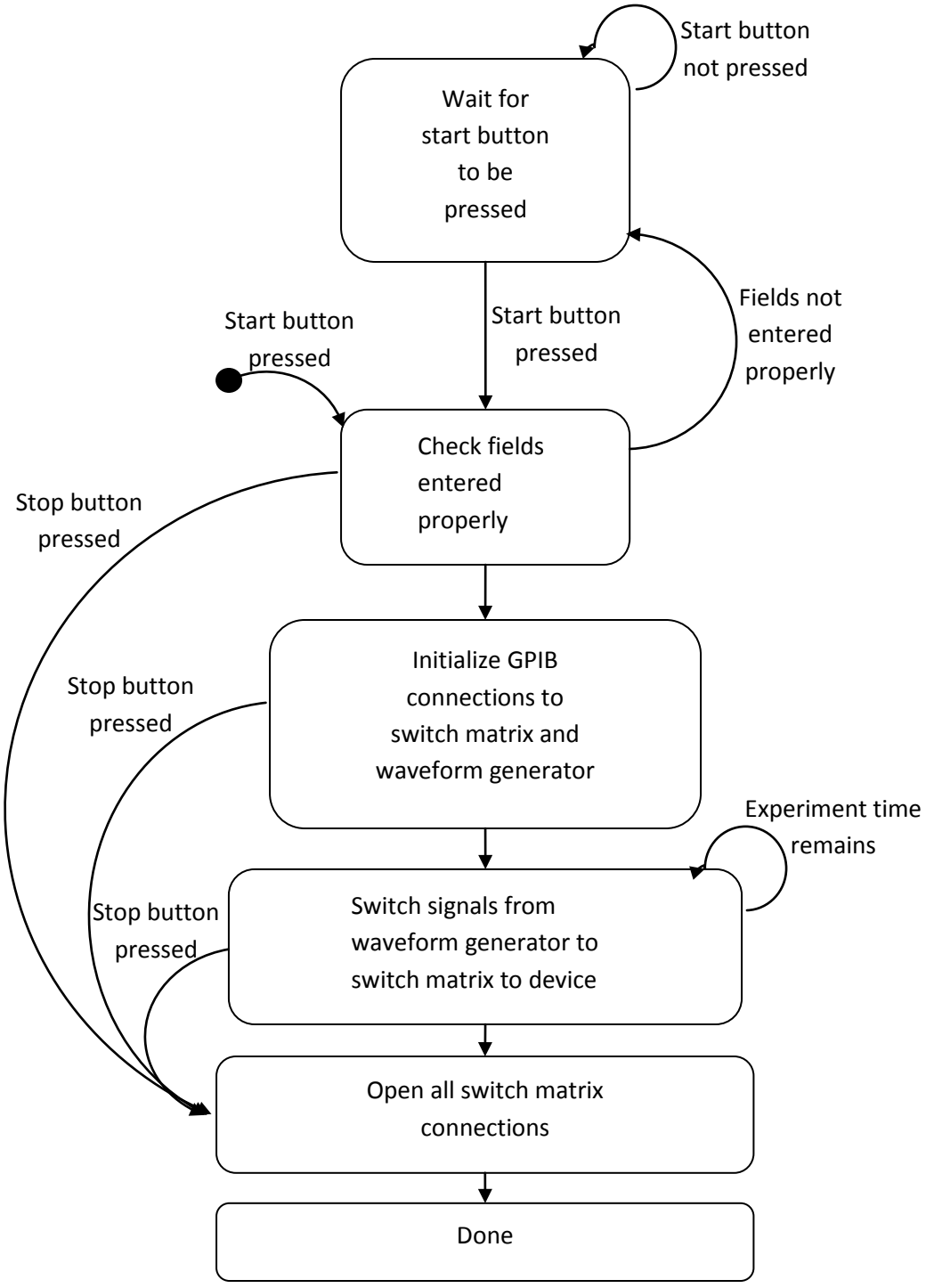


Figure 4-5: State chart of program used to automate testing of devices.

Chapter 5: Device Performance

This chapter describes the performance of the dielectrophoretic cage device. Particle trajectories were evaluated by tracking the particle positions at every frame in a collected video file. From this data, the particle velocities and accelerations can also be evaluated. Since this is a dielectrophoretic device, the particle acceleration is proportional to the dielectrophoretic force in the direction of the particles trajectory. The overall force was calculated using the "Particle Tracing" post-processing tab in COMSOL Multiphysics. Lastly, at the end of this chapter an alternative particle actuation scheme will be introduced.

5.1: Particle Tracking

As mentioned in chapter 4, experiments were recorded using a digital camera at 15 fps (frames per second). These videos were analyzed using Tracker [1], an open-source program used for video analysis and modeling in physics education. The software was made by Douglas Brown, a retired instructor at Cabrillo College located in Aptos, California.

As an example, in the video presented here, there were twenty-two 10 µm diameter polystyrene beads which moved from left to right in an approximately 4 minute long experiment. In the screen shot shown in figure 5-1, one of the polystyrene beads is tracked. There are some distortions in the images due to

roughness of the lid surface caused during the milling process. The particle that was traced frame by frame is shown in red. The position and time information were extracted for further analysis. The gold bars are electrodes 100 μm in width. The darker regions between the electrodes are 75 μm wide gaps. With the help of the Tracker software, purple lines were drawn in to designate the origin and axis. A blue 100 μm scale bar was also made based on the known electrode width.

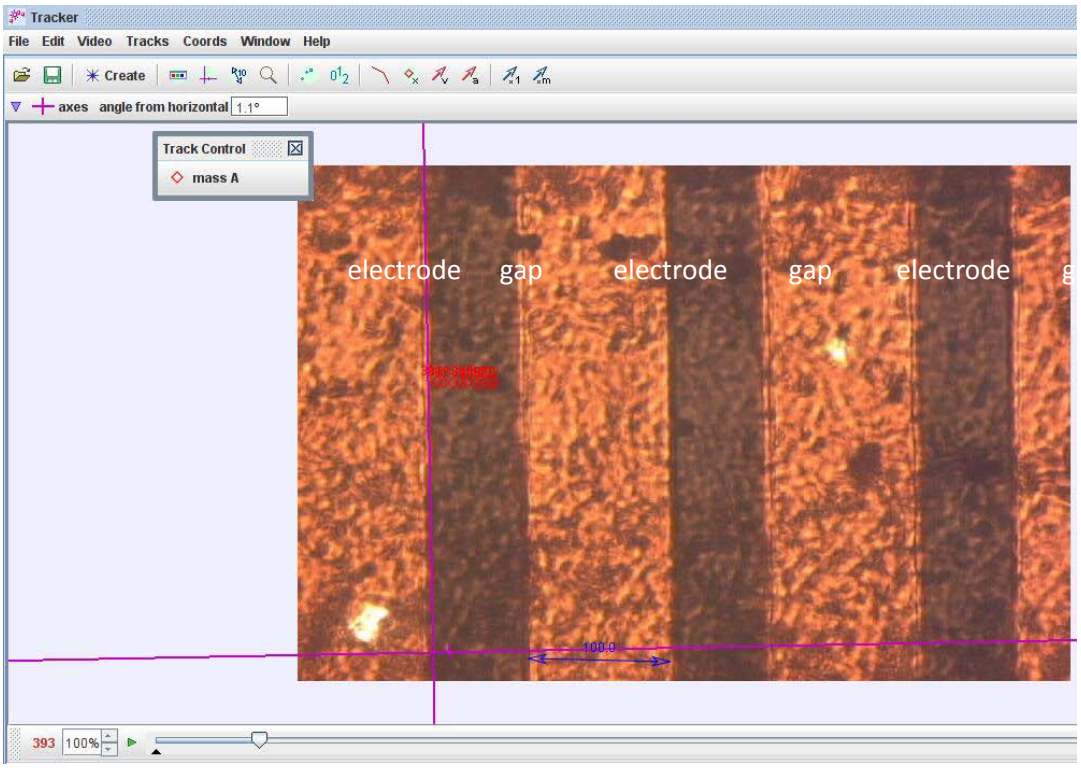


Figure 5-1: Screenshot of particle being tracked inside the microfluidic cavity of the DEP based LOAC device. The series of red marks track the particle as it moves from left to right. The golden strips are 100 µm wide electrodes and the darker strips are 75 µm gaps between them. The origin and axis are shown in purple. A blue line is drawn to denote length of 100 µm.

5.2: Particle Velocity vs. Position Analysis

Particle movement was a very positive sign. However, it needed to be verified that the particles were being actuated electrically by the driven

microfabricated electrodes in accordance with dielectrophoresis principles. Brownian motion was not investigated. According to Jones [2] for particles from 1 micron to 1 millimetre in diameter Brownian motion is minimal. The dielectrophoretic force (Eqn 2.2) is directly proportional to the gradient of the electric field squared, which can be rewritten in 2-D as follows:

$$\nabla|E_{RMS}|^2 = 2\{(E_x E_{x,x} + E_y E_{y,x})\hat{x} + (E_x E_{x,y} + E_y E_{y,y})\hat{y}\}$$

Particles moved from left to right along the x-axis during the experiment. For that reason, only the field gradient information in the x-axis was necessary for analysis.

$$\nabla\left|E_{RMS_x}\right|^2 = 2(E_x E_{x,x} + E_y E_{y,x})\hat{x}$$

Bearing in mind that the real part of the Clausius-Mossotti factor was -0.42 for the polystyrene bead of 10 µm diameter moving in DI water and actuated by sinusoidal signals at 10 kHz, the dielectrophoretic force would be proportional to the negative electric field squared gradient.

The nDEP cage manipulation steps are shown along with the corresponding values of negative electric field squared gradient in the x-direction. COMSOL Multiphysics® was used to calculate the gradient of the electric field squared everywhere in the chamber. The applied voltage amplitude was 5 V. Figure 5-2 illustrates the DEP cage progression at three consecutive switch times. At any given time, if the applied frequency is chosen correctly, suspended polystyrene spheres will move along the electric field squared gradient toward the nearest electric field

minimum. In Figure 5-2, electric field minima appear as dark blue regions that are localized (A), expanded (B), and re-localized (C).

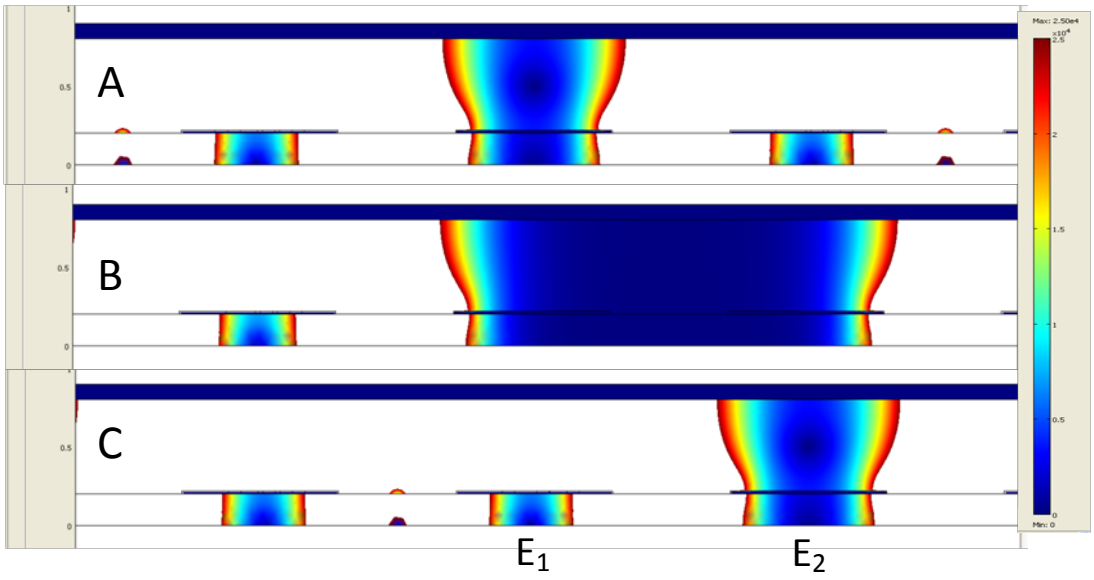


Figure 5-2: COMSOL Multiphysics simulations. A) a localized nDEP cage over electrode E_1, B) an expanded nDEP cage over electrode E_1 and E_2 and C) a re-localized nDEP cage over electrode E_2. The colour bar represents the electric field, which ranges from 0 V/m (blue) to 2.5×10^4 V/m (red).

These transitions are modelled in figure 5-3 in terms of the negative electric field squared gradient, which is proportional to the nDEP force experienced by the spheres. Spheres experiencing nDEP will be drawn to negative field squared gradients. At time A, particles are driven toward the field (and DEP force) null above Electrode E_1. After collecting there, the electrode phases are shifted at time B, when the field null region expands, allowing particles to move freely from left to

right. At time C, most of the particles at the right edge of electrode E_1 will move right because of the increasing negative field squared gradient to the right of electrode E_1. However, particles that are located slightly left of the right edge of electrode E_1 can rock back leftward because there is an increasing negative field squared gradient path there as well.

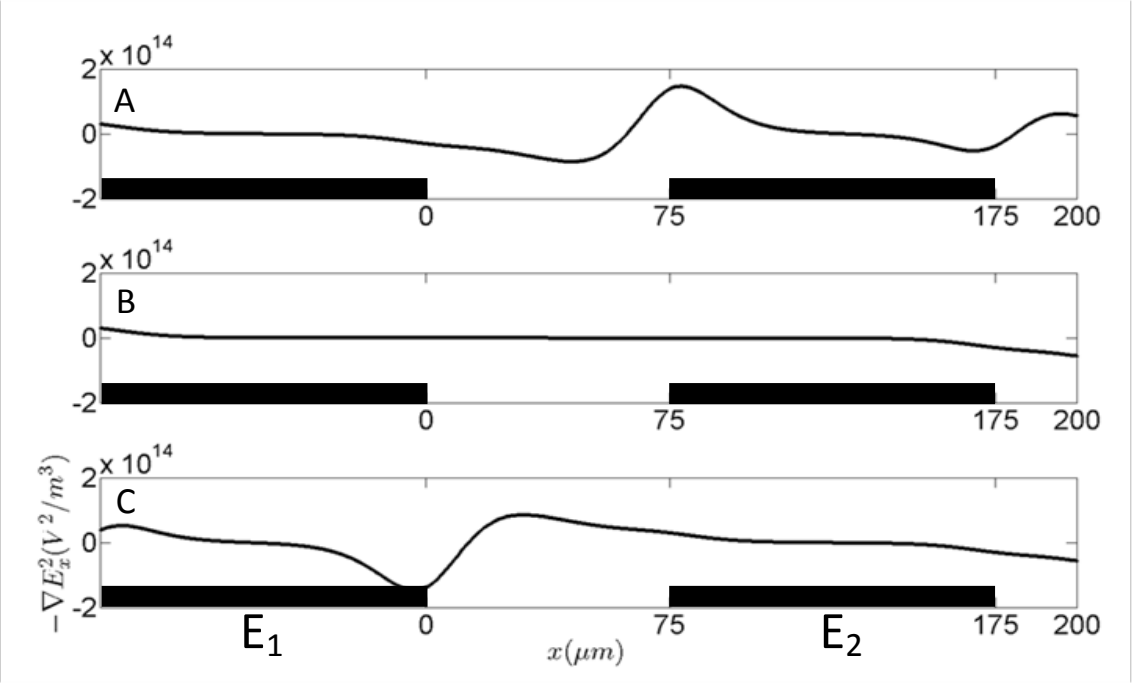


Figure 5-3: COMSOL Multiphysics cross-sections of the negative field squared gradient for A) a localized nDEP cage over electrode E_1, B) an expanded nDEP cage over electrode E_1 and E_2 and C) a re-localized nDEP cage over electrode E_2

At time C, when the expanded nDEP cage re-localizes one electrode right to electrode E_2 is when particle movement should occur. The measurement data shows a strong correlation between the velocity profile of 10 µm diameter

polystyrene sphere in DI water and the negative field squared gradient at time C. This data analysis is shown in figure 5-4.

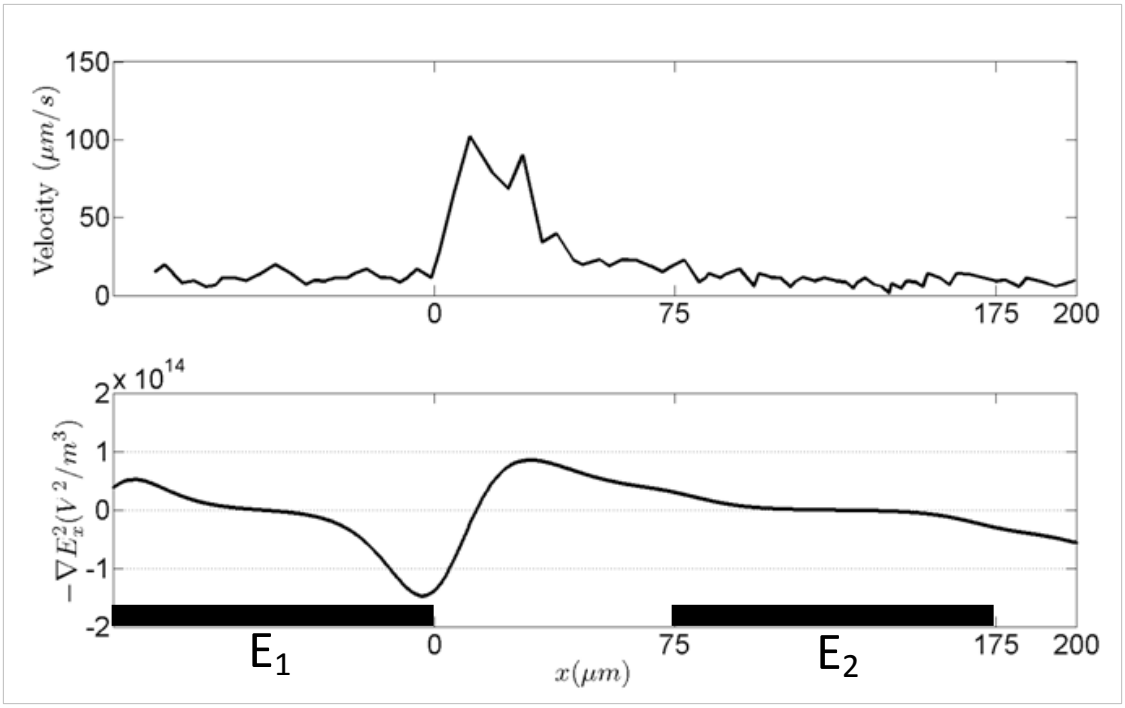


Figure 5-4: Correlation between the velocity profile of a 10 µm diameter polystyrene bead in DI water (top) and negative field squared gradient at time C (bottom).

There is a point in between electrode E_1 and electrode E_2 where the field squared gradient is 0 V^2/m^2. At this point it is reasonable to suspect no dielectrophoretic force and thus no particle velocity. However, that point is very brief on a steep positive slope in the negative field squared gradient profile and a 10 micron diameter would likely overcome that brief point and continue moving right. Figure 5-5 plots a particle crossing the gap between electrode E_1 and electrode E_2 in approximately three seconds.

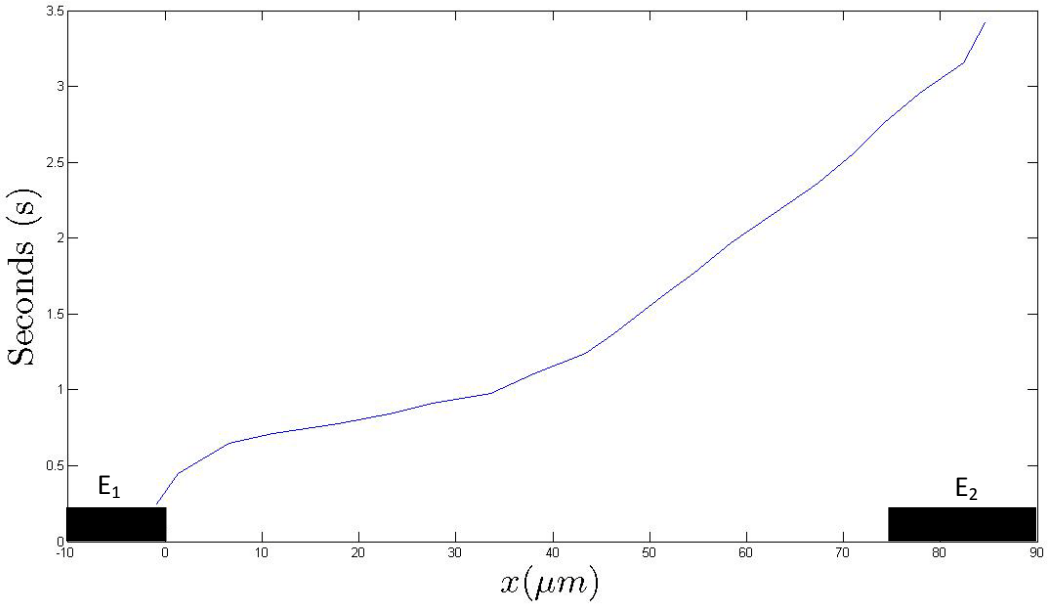


Figure 5-5: Position versus time plot of a 10 micron diameter polystyrene bead crossing the gap between electrode E_1 and electrode E_2.

5.3: Alternative Particle Actuation Scheme

During many experiments it was noticed that the particles had a tendency to settle to the bottom of the microfluidic chamber. According to Stoke's Law, sedimentation velocity occurs when drag force equals net gravity force:

$$6\pi\eta Rv = \frac{4}{3}\pi R^3(\rho - \rho_0)g$$

Solving for v:

$$v = \frac{2}{9}R^2(\rho - \rho_0)\frac{g}{\eta}$$

For R=5 microns, rho = 1050 kg/m (water), rho_0 = 1000 kg/m (polystyrene), eta=0.001, g=9.81m/s^2, sedimentation velocity is:

$$v = 2.73\ \mu m/s$$

Below in figure 5-6a the nDEP manipulation steps are shown for actuating particles leftward. In time step α, particles are drawn to the nDEP cage over electrode E_3. In time step β, the nDEP cage is grown to include electrode E_2 and E_3. Finally in time step γ, the nDEP cage is just over electrode E_2. It was thought that the nDEP cage spread too low in time β. Throughout the experiments nDEP forces had to overcome gravity to keep the particles levitated above the electrodes so that they could be translated sideways. However, in step β, particles were vulnerable to settle in between electrode E_2 and E_3.

In figure 5-6b the nDEP cage was lifted away from the device floor. In figure 5-6a all the electrodes were driven by the same sinusoidal amplitude of 5V. In figure 5-6b, time step β, the lid was driven with 1V with a phase shift of 180^0 and the electrodes on the device floor were driven with 10 V.

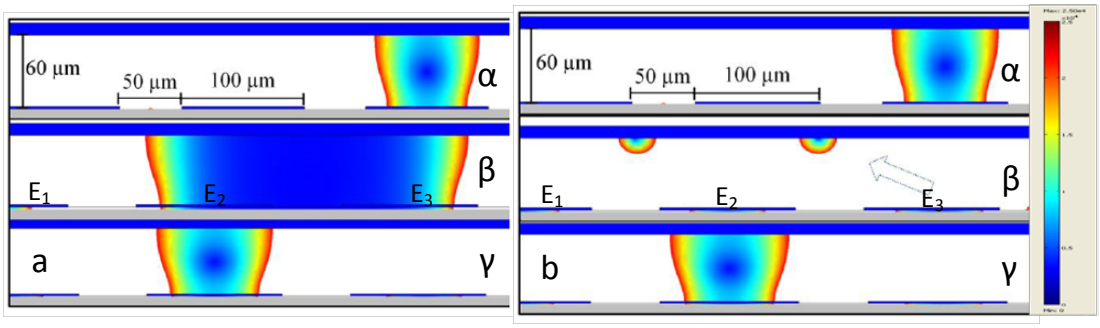


Figure 5-6: a)Steps shown for moving nDEP cages leftward as presented previously in this thesis. All the electrodes and the conductive lid are driven with 5 V. b) An alternative scheme for moving particles by raising the nDEP cage higher in time β. The nDEP cage is lifted higher by driving the conductive lid at 1V and driving the electrodes on the device floor at 10V.

Figures 5-7 to 5-9 show screen shots of a column of particles being actuated left using the alternative scheme.

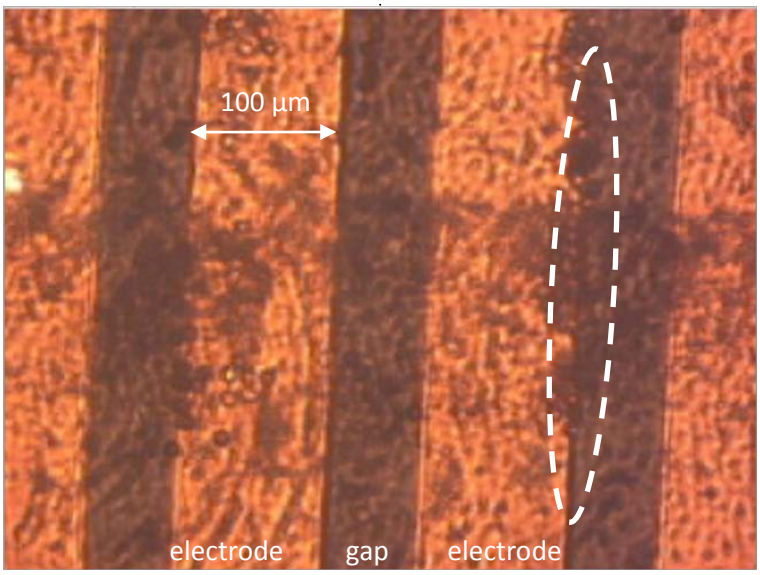


Figure 5-7: Column of particles being actuated from right to left at t=0s. Electrodes are 100 µm wide and gaps are 75 µm wide

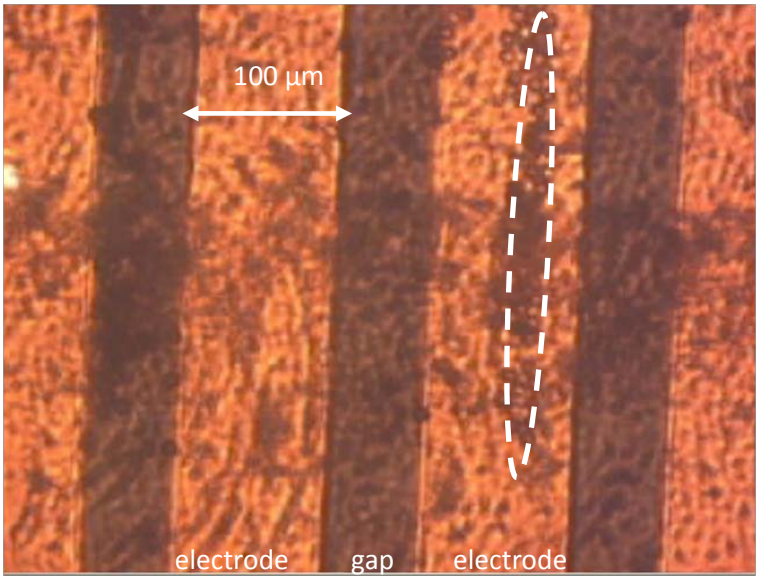


Figure 5-8: Column of particles being actuated from right to left at t=8s. Electrodes are 100 µm wide and gaps are 75 µm wide

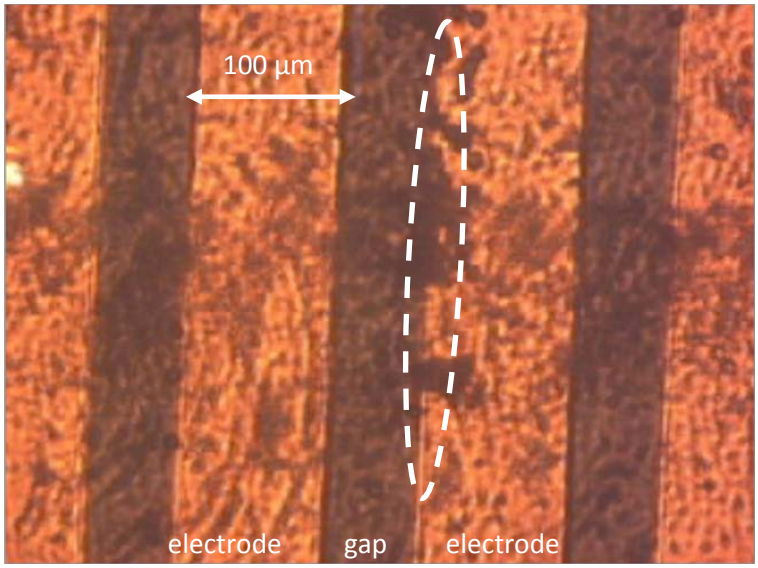


Figure 5-9: Column of particles being actuated from right to left at t=16s. Electrodes are 100 µm wide and gaps are 75 µm wide

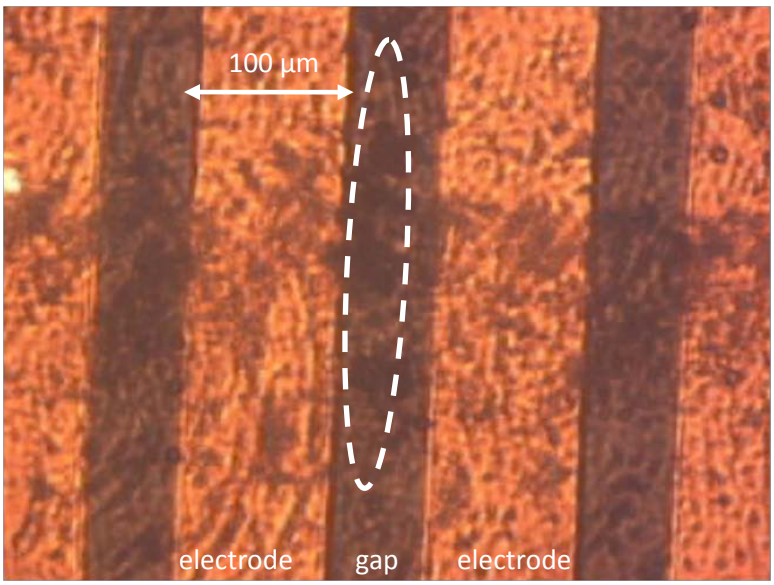


Figure 5-10: Column of particles being actuated from right to left at t=18s. Electrodes are 100 µm wide and gaps are 75 µm wide

[1] "Tracker", www.cabrillo.edu/~dbrown/tracker, last accessed June 2010

[2] T.B. Jones, *Electromechanics of Particles,* Cambridge University Press, Cambridge, 1995

Chapter 6: Summary and Future Work

Feynman's talk "Plenty of Room at the Bottom" was an invitation for scientists of different backgrounds to work together for common goals. Following the successful collaboration of scientists all around the world on the human genome project, lab-on-a-chip is providing another exciting opportunity for multidisciplinary work. Lab-on-a-chip initiatives have rapidly developed, particularly for exploiting the dielectrophoresis phenomenon in lab-on-a-chip devices. The number of dielectrophoresis-related publications has increased from 50 written in the year 2000 to a projection of approximately 350 written in the year 2010. In a recent review written by Ronald Pethig[1] entitled, "Dielectrophoresis: Status of the theory, technology, and applications", he mentions that the publishing trend for 2010 suggests that the theory (5%) and technology (18%) of dielectrophoresis have matured sufficiently for efforts to be directed mainly toward the publication of applications (77%). In this light, the dielectrophoresis-based lab-on-a-chip device developed in this work will act as another incremental step toward the development of DEP applications on silicon.

There were two main objectives of this work. The first was to develop a fabrication process for a dielectrophoresis-based lab-on-a-chip device on silicon. The second was to successfully test the microfabricated device. After many years of work and two entirely new lithography mask sets both objectives were met. The

reader was privy to many of the challenges that were overcome. Hopefully future students can use this document to help them in their fabrication endeavours. Students, staff and industry specialists working in the Nano Systems Fabrication Laboratory (NSFL) at the University of Manitoba can benefit from many established processes as a result of this thesis. The cleanroom as of the now does not have a Plasma Enhanced Chemical Vapour Deposition (PECVD) system to deposit low temperature oxide. Therefore, the recipe found and tested in this thesis is the only known way to deposit low temperature oxide which is imperative in multi-layer metal designs. Although the KOH silicon etch bath process was not used in the end for the final working device, it remains for others to use. Similarly the gold sputter deposition recipes which were employed to successfully adhere gold to the silicon substrate can be reused.

6.1: Future Work

The brunt of the challenges presented themselves in the design and fabrication stage of this thesis. Now that working devices have been made and a fabrication process established, there are many future research opportunities. This work focused on testing the devices with 10 µm diameter polystyrene beads but future research could investigate biological cells of various phenotypes and sizes. In the design and experimental apparatus chapters it was explained that particles could be driven right or left. One could theoretically separate a mixed population of

particles by driving one phenotype left and another phenotype right. In the device performance chapter it was mentioned that particles settling in the gaps between electrodes was a challenge. The experiments in this thesis focused on 5 second switching times but it would be interesting to investigate the role of switching speeds on particle settling. An alternative particle actuation scheme was provided in the device performance chapter that was shown to work but further analysis should be done. Finally, performance of the dielectrophoresis-enabled lab-on-a-chip device was judged upon video microscopy, but in the design chapter it was explained that capacitive measurements between a target electrode and the conductive lid could be used to evaluate particle movement.

[1] R. Pethig, "Dielectrophoresis: Status of the Theory, Technology, and Applications", Biomicrofluidics, Vol 4, Issue 2, pg 1 – 35, 2010

Appendix A: Derivation of Dielectrophoresis Formula

The Dielectrophoresis equation is referred to often in this thesis. An attempt to derive the formula follows. This proof includes a lot of work published by Jones[1] and Cheng[2].

The proof begins with an infinitesimally small dipole. The dipole is situated a distance away from the origin. The vector **r** describes the distance between the origin and the negative pole of the dipole. Vector **d** describes the separation between the negative pole and the positive pole. Each of the poles experiences a different field as shown below.

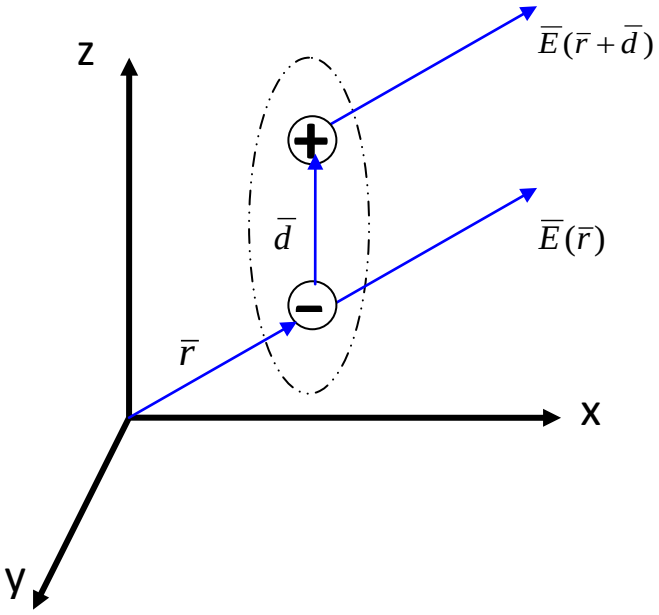


Figure 1: Forces acting upon dipole from [1]

The total force acting upon the dipole is:

$$\overline{F} = q\overline{E}(\overline{r} + \overline{d}) - q\overline{E}(\overline{r})$$

Where the field on the positive pole can be expanded to:

$$\overline{E}(\overline{r} + \overline{d}) = \overline{E}(\overline{r}) + \overline{d} \bullet \nabla\overline{E}(r) + ...$$

Therefore,

$$\overline{F} = q(\overline{E}(\overline{r}) + \overline{d} \bullet \nabla\overline{E}(\overline{r}) + ...) - q\overline{E}(\overline{r})$$

$$\overline{F} = q\overline{d} \bullet \nabla\overline{E}(\overline{r})$$

$$\overline{F}_{dipole} = \overline{p} \bullet \nabla\overline{E}$$

However, what is the momentum **p**? The first step towards determining the momentum is defining the electrostatic force acting upon the dipole. In order to do this, the dipole is repositioned on the axis. Now vector **r** describes the distance between the middle of the dipole and an arbitrary point. Vector $\mathbf{r}_+$ and $\mathbf{r}_-$ describe the distances between the positive pole and negative pole and the same arbitrary point. This new arrangement is shown below.

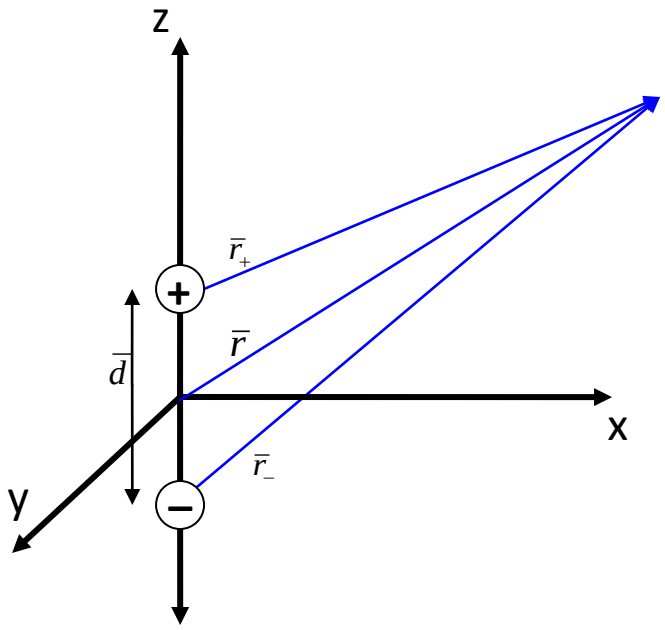


Figure 2: Vectors originating from the dipole to an arbitrary point found in [1]

The distance between two points in polar coordinates is:

$$\left(r_1^{\,2} + r_2^{\,2} - 2r_1 r_2 \cos(\theta_1 - \theta_2)\right)^{1/2}$$

So then,

$$r_+ = [r^2 + \left(\frac{d}{2}\right)^2 - 2\left(\frac{d}{2}\right) r\cos(\theta)]^{1/2}$$

$$r_+ = [r^2 + \left(\frac{d}{2}\right)^2 - dr\cos(\theta)]^{1/2}$$

$$r_{+} = r[1+\left(\frac{d}{2r}\right)^{2} - \left(\frac{d}{r}\right)\cos(\theta)]^{1/2}$$

$$\left(\frac{r}{r_{+}}\right) = [1+\left(\frac{d}{2r}\right)^{2} - \left(\frac{d}{r}\right)\cos(\theta)]^{-1/2}$$

Also,

$$r_{-} = [r^{2} + \left(\frac{-d}{2}\right)^{2} - 2\left(\frac{-d}{2}\right)r\cos(\theta)]^{1/2}$$

$$r_{-} = [r^{2} + \left(\frac{d}{2}\right)^{2} + dr\cos(\theta)]^{1/2}$$

$$r_{-} = r[1+\left(\frac{d}{2r}\right)^{2} + \left(\frac{d}{r}\right)\cos(\theta)]^{1/2}$$

$$\left(\frac{r}{r_{-}}\right) = [1+\left(\frac{d}{2r}\right)^{2} + \left(\frac{d}{r}\right)\cos(\theta)]^{-1/2}$$

So that,

$$\left(\frac{r}{r_{\pm}}\right) = [1+\left(\frac{d}{2r}\right)^{2} \mp \left(\frac{d}{r}\right)\cos(\theta)]^{-1/2}$$

In the Maclaurin Series:

$$(1+x)^{-1/2} = 1 - \frac{x}{2} + \frac{3x^2}{8} - \frac{5x^3}{16} + \ldots$$

If x is substituted with:

$$x = \left(\frac{d}{2r}\right)^2 \mp \left(\frac{d}{r}\right)\cos(\theta)$$
$$x = \left(\frac{d}{r}\right)\left(\frac{d}{4r} \pm \cos(\theta)\right)$$

Then:

$$\left(\frac{r}{r_\pm}\right) = 1 - \frac{d}{2r}\left(\frac{d}{4r} \mp \cos(\theta)\right) + \frac{3}{8}\left(\frac{d}{r}\right)^2\left(\frac{d}{4r} \mp \cos(\theta)\right)^2 - \frac{5}{16}\left(\frac{d}{r}\right)^3\left(\frac{d}{4r} \mp \cos(\theta)\right)^3 + \ldots$$

$$\left(\frac{r}{r_\pm}\right) = 1 - \frac{d^2}{8r^2} \pm \frac{d}{2r}\cos(\theta) + \frac{3}{8}\frac{d^2}{r^2}\left(\left(\frac{d}{4r}\right)^2 \mp \frac{d}{2r}\cos(\theta) + \cos^2(\theta)\right)$$
$$-\frac{5}{16}\left(\frac{d}{r}\right)^3\left(\left(\frac{d}{4r}\right)^3 \mp 3\left(\frac{d}{4r}\right)^2\cos(\theta) \mp 3\left(\frac{d}{4r}\right)\cos^2(\theta) + \cos^3(\theta)\right)$$

$$\left(\frac{r}{r_\pm}\right) = 1 - \frac{d^2}{r^2} \pm \frac{d}{2r}\cos(\theta) + \frac{3}{128}\frac{d^4}{r^4} \mp \frac{3}{16}\frac{d^3}{r^3}\cos(\theta) + \frac{3}{8}\frac{d^2}{r^2}\cos^2(\theta) - \frac{5}{1024}\frac{d^6}{r^6}$$
$$\pm\frac{15}{256}\frac{d^5}{r^5}\cos(\theta) \pm \frac{15}{64}\frac{d^4}{r^4}\cos^2(\theta) - \frac{5}{16}\frac{d^3}{r^3}\cos^3(\theta)$$

If all terms greater than third order are ignored the ratio is reduced to:

$$\left(\frac{r}{r_{\pm}}\right)=1\pm\left(\frac{d}{2r}\right)\cos(\theta)+\left(\frac{d}{2r}\right)^{2}\left(\frac{3}{2}\cos^{2}(\theta)-\frac{1}{2}\right)\pm\left(\frac{d}{2r}\right)^{3}\left(\frac{5}{2}\cos^{3}(\theta)-\frac{3}{2}\cos(\theta)\right)$$

If the terms are grouped into terms P_0, P_1, P_2 and P_3:

P_n	Term
n=0	1
n=1	$\cos(\theta)$
n=2	$\frac{3}{2}\cos^{2}(\theta)-\frac{1}{2}$
n=3	$\frac{5}{2}\cos^{3}(\theta)-\frac{3}{2}\cos(\theta)$

Table 1: Legendre's Polynomials

It becomes evident that the distance ratios are Legendre's equations and can be written as:

$$\left(\frac{r}{r_{\pm}}\right)=P_0\pm\left(\frac{d}{2r}\right)P_1+\left(\frac{d}{2r}\right)^{2}P_2\pm\left(\frac{d}{2r}\right)^{3}P_3$$

The electrostatic potential of the dipole can be described as:

$$\Phi(r,\theta) = \frac{q}{4\pi\varepsilon r_+} - \frac{q}{4\pi\varepsilon r_-} = \frac{q}{4\pi\varepsilon}\left(\frac{1}{r_+} - \frac{1}{r_-}\right)$$

If both sides of the equation are multiplied by r such that:

$$r\Phi(r,\theta) = \frac{q}{4\pi\varepsilon}\left(\frac{r}{r_+} - \frac{r}{r_-}\right)$$

Then the distance ratios can be substituted to get:

$$r\Phi(r,\theta) = \frac{q}{4\pi\varepsilon}\left(P_0 + \left(\frac{d}{2r}\right)P_1 + \left(\frac{d}{2r}\right)^2 P_2 + \left(\frac{d}{2r}\right)^3 P_3 - \left(P_0 - \left(\frac{d}{2r}\right)P_1 + \left(\frac{d}{2r}\right)^2 P_2 - \left(\frac{d}{2r}\right)^3 P_3\right)\right)$$

$$r\Phi(r,\theta) = \frac{q}{4\pi\varepsilon}\left(2\left(\frac{d}{2r}\right)P_1 + 2\left(\frac{d}{2r}\right)^3 P_3\right)$$

$$r\Phi(r,\theta) = \frac{qd\cos(\theta)}{4\pi\varepsilon r} + \frac{qd^3\left(\frac{5}{2}\cos\cos^3(\theta) - \frac{3}{2}\cos(\theta)\right)}{16\pi\varepsilon r^3}$$

Then if the r term is divided out on both sides:

$$\Phi(r,\theta) = \frac{qd\cos(\theta)}{4\pi\varepsilon r^2} + \frac{qd^3\left(\frac{5}{2}\cos^3(\theta) - \frac{3}{2}\cos(\theta)\right)}{16\pi\varepsilon r^4}$$

The first term describes the first dipole moment (p=qd) and will be needed later.

Next, the Laplace equation in polar coordinates is considered:

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial V}{\partial r}\right)+\frac{1}{r^2\sin(\theta)}\frac{\partial}{\partial\theta}\left(\sin(\theta)\frac{\partial V}{\partial\theta}\right)=0$$

Where:

$$V(r,\theta)=\Gamma(r)\Theta(\theta)$$

So that the Laplace equation can be simplified to:

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\left(\Gamma(r)\Theta(\theta)\right)\right)+\frac{1}{r^2\sin(\theta)}\frac{\partial}{\partial\theta}\left(\sin(\theta)\frac{\partial}{\partial\theta}\left(\Gamma(r)\Theta(\theta)\right)\right)=0$$

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\Gamma(r)}{\partial r}\Theta(\theta)\right)+\frac{1}{r^2\sin(\theta)}\frac{\partial}{\partial\theta}\left(\sin(\theta)\Gamma(r)\frac{\partial\Theta(\theta)}{\partial\theta}\right)=0$$

$$\frac{\Theta(\theta)}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\Gamma(r)}{\partial r}\right)+\frac{\Gamma(r)}{r^2\sin(\theta)}\frac{\partial}{\partial\theta}\left(\sin(\theta)\frac{\partial\Theta(\theta)}{\partial\theta}\right)=0$$

$$\Theta(\theta)\left(2r\frac{\partial\Gamma(r)}{\partial r}+r^2\frac{\partial^2\Gamma(r)}{\partial r^2}\right)+\frac{\Gamma(r)}{\sin(\theta)}\left(\cos(\theta)\frac{\partial\Theta(\theta)}{\partial\theta}+\sin(\theta)\frac{\partial^2\Theta(\theta)}{\partial\theta^2}\right)=0$$

$$\frac{1}{\Gamma(r)}\left(2r\frac{\partial\Gamma(r)}{\partial r}+r^2\frac{\partial^2\Gamma(r)}{\partial r^2}\right)+\frac{1}{\Theta(\theta)\sin(\theta)}\left(\cos(\theta)\frac{\partial\Theta(\theta)}{\partial\theta}+\sin(\theta)\frac{\partial^2\Theta(\theta)}{\partial\theta^2}\right)=0$$

If the first term is equated to k^2 then the second term becomes $-k^2$ and the equation can be written as:

$$\frac{1}{\Gamma(r)}\left(2r\frac{\partial\Gamma(r)}{\partial r}+r^2\frac{\partial^2\Gamma(r)}{\partial r^2}\right)=k^2$$

$$\frac{1}{\Theta(\theta)\sin(\theta)}\left(\cos(\theta)\frac{\partial\Theta(\theta)}{\partial\theta}+\sin(\theta)\frac{\partial^2\Theta(\theta)}{\partial\theta^2}\right)=-k^2$$

$$\frac{1}{\Gamma(r)}\left(2r\frac{\partial\Gamma(r)}{\partial r}+r^2\frac{\partial^2\Gamma(r)}{\partial r^2}\right)=k^2$$

$$r^2\frac{\partial^2\Gamma(r)}{\partial r^2}+2r\frac{\partial\Gamma(r)}{\partial r}-k^2\Gamma(r)=0$$

The last step proves that the Laplace equation in polar coordinates is a Cauchy-Euler equation in the form of:

$$x^2\frac{\partial^2 y}{\partial x^2}+2x\frac{\partial y}{\partial x}-k^2 y=0$$

In order to solve the Cauchy-Euler equation the following substitution is made:

$$x=e^t$$

From this point it follows that:

$$\frac{dx}{dt} = e^t$$

$$x\frac{dy}{dx} = \frac{e^t dy}{e^t dt} = \frac{dy}{dt}$$

$$x^2\frac{d^2y}{dx^2} = \left(e^t\right)^2\frac{d}{dx}\left(\frac{dy}{dx}\right) = \left(e^t\right)^2\frac{d}{e^t dt}\left(\frac{dy}{e^t dt}\right) = e^t\frac{\partial}{\partial t}\left(e^{-t}\frac{dy}{dt}\right)$$

$$x^2\frac{d^2y}{dx^2} = e^t\left(e^{-t}\frac{d^2y}{dt^2} - e^{-t}\frac{dy}{dt}\right) = \frac{d^2y}{dt^2} - \frac{dy}{dt}$$

With the substitutions the Cauchy-Euler equation can be written as:

$$\frac{d^2y}{dt^2} - \frac{dy}{dt} + 2\frac{dy}{dt} - k^2y = 0$$

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} - k^2y = 0$$

The following auxiliary equation can be used to solve for y(x):

$$s^2 + s - k^2 = 0$$

$$s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$s = \frac{-(1) \pm \sqrt{(1)^2 - 4(1)(-k^2)}}{2(1)}$$

$$s_1 = \frac{-1 + \sqrt{1 + 4k^2}}{2} = n$$

$$s_2 = \frac{-1 - \sqrt{1 + 4k^2}}{2} = -(n+1)$$

The proof of the relationship between n and r follows:

$$s_2 = -(n+1) = -n-1 = -s_1 - 1$$

$$-\left(\frac{-1+\sqrt{1+4k^2}}{2}\right)-1=\left(\frac{1-\sqrt{1+4k^2}}{2}\right)-1=\frac{-1-\sqrt{1+4k^2}}{2}=s_2$$

Then y(x) can be written:

$$y(x) = A_n x^n + B_n x^{-(n+1)}$$

Thus:

$$\Gamma(r) = A_n r^n + B_n r^{-(n+1)}$$

In order to find $\Theta(\theta)$ the relationship between k and n is proven:

$$k^2 = n(n+1)$$

$$k^2 = \left(\frac{-1+\sqrt{1+4k^2}}{2}\right)\left(\frac{-1+\sqrt{1+4k^2}}{2}+1\right)$$

$$k^2 = \left(\frac{-1}{2}+\frac{\sqrt{1+4k^2}}{2}\right)\left(\frac{1}{2}+\frac{\sqrt{1+4k^2}}{2}\right)$$

$$k^2 = \frac{-1}{4}-\frac{\sqrt{1+4k^2}}{4}+\frac{\sqrt{1+4k^2}}{4}+\frac{1+4k^2}{4}$$

$$k^2 = \frac{-1+1+4k^2}{4} = k^2$$

Now $\Theta(\theta)$ is solved for:

$$\frac{1}{\Theta(\theta)\sin(\theta)}\left(\sin(\theta)\frac{\partial^2\Theta(\theta)}{\partial\theta^2}+\cos(\theta)\frac{\partial\Theta(\theta)}{\partial\theta^2}\right) = -k^2$$

$$\sin(\theta)\frac{\partial^2\Theta(\theta)}{\partial\theta^2}+\cos(\theta)\frac{\partial\Theta(\theta)}{\partial\theta^2}+k^2\sin(\theta)\Theta(\theta) = 0$$

$$\sin(\theta)\frac{\partial^2\Theta(\theta)}{\partial\theta^2}+\cos(\theta)\frac{\partial\Theta(\theta)}{\partial\theta^2}+n(n+1)\sin(\theta)\Theta(\theta) = 0$$

The last line looks suspiciously like a Legendre's Equation but it needs to be confirmed:

$$\frac{\partial}{\partial x}\left[\left(1-x^2\right)\frac{dy}{dx}\right]+n(n+1)y = 0$$

After a substitution:

$$x = \cos(\theta)$$

$$\frac{dx}{d\theta} = -\sin(\theta)$$

$$\therefore 1 - x^2 = 1 - \cos^2(\theta) = \sin^2(\theta)$$

$$\frac{-1}{\sin(\theta)} \frac{\partial}{\partial \theta} \left[1 - \cos^2(\theta) \frac{\partial y}{-\sin(\theta) \partial \theta} \right] + n(n+1) y = 0$$

$$\frac{-1}{\sin(\theta)} \frac{\partial}{\partial \theta} \left(\frac{1 - \cos^2(\theta)}{-\sin(\theta)} \frac{\partial y}{\partial \theta} \right) + n(n+1) y = 0$$

$$\frac{1}{\sin(\theta)} \frac{\partial}{\partial \theta} \left(\sin(\theta) \frac{\partial y}{\partial \theta} \right) + n(n+1) = 0$$

$$\frac{1}{\sin(\theta)} \left(\sin(\theta) \frac{\partial^2 y}{\partial \theta^2} + \cos(\theta) \frac{\partial y}{\partial \theta} \right) + n(n+1) y = 0$$

$$\frac{\partial^2 y}{\partial \theta^2} + \frac{\cos(\theta)}{\sin(\theta)} \frac{\partial y}{\partial \theta} + n(n+1) y = 0$$

The equation has been proved to be a Legendre Equation. The Legendre solution for n=1 will be taken because one dipole is considered.

$$\Theta(\theta) = \cos(\theta)$$

The equation describing the potential on the dipole is:

$$V(r, \theta) = \Gamma(r) \Theta(\theta)$$

$$V(r, \theta) = (A_n r^n + B_n r^{-(n+1)}) \cos(\theta)$$

Two cases need to be considered when attempting to understand the potential on the dipole. One equation will describe what's happening outside the spherical dipole with fixed radius R in the medium. The other equation will describe the potential within the spherical dipole or particle. The figure below illustrates the distinction.

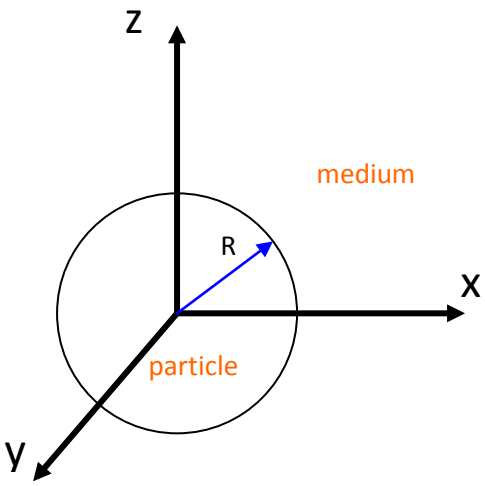


Figure 3: Dipole within a sphere of fixed radius R from [1]

So the equation for the electrostatic potential in the medium, V_{med} when r>R is:

$$V_{med}(r,\theta) = -E_o r\cos(\theta) + \frac{A\cos(\theta)}{r^2}$$

Where E_o is the applied field.

The equation for the electrostatic potential in the particle, V_p when r<R is:

$$V_p(r,\theta) = -Br\cos(\theta) + \frac{C\cos(\theta)}{r_2}$$

If it is assumed that the particle with a dipole is a perfect sphere and lossless, then coefficient C=0. Secondly, it is recognized that $V_{med} = V_p$ where r=R.

$$-E_o R\cos(\theta) + \frac{A\cos(\theta)}{R^2} = -BR\cos(\theta)$$
$$-E_o R + \frac{A}{R^2} = -BR$$
$$-E_o + \frac{A}{R^3} = -B$$
$$B = E_o - AR^{-3}$$

As well, the normal component of the displacement flux must be continuous across the boundary.

$$\varepsilon_p E_p(r = R,\theta) = \varepsilon_{med} E_{med}(r = R,\theta)$$

Where:

$$E = -\frac{\partial V}{\partial r}$$

$$E_{med} = -\frac{\partial}{\partial r}\left(-E_o R\cos(\theta) + AR^{-3}\cos(\theta)\right)$$

$$E_{med} = E_o\cos(\theta) - 2A\cos(\theta)R^{-3}$$

$$E_p = -\frac{\partial}{\partial r}\left(-BR\cos(\theta)\right)$$

$$E_p = B\cos(\theta)$$

So then:

$$\varepsilon_{med}(E_o\cos(\theta) + 2A\cos(\theta)R^{-3}) = \varepsilon_p(B\cos(\theta))$$

$$\varepsilon_{med}(E_o + 2AR^{-3}) = \varepsilon_p(B)$$

$$\varepsilon_{med}(E_o + 2AR^{-3}) = \varepsilon_p(E_o - AR^{-3})$$

$$\varepsilon_{med}E_o + \varepsilon_{med}2AR^{-3} = \varepsilon_p E_o - \varepsilon_p AR^{-3}$$

$$AR^{-3}(\varepsilon_p + 2\varepsilon_{med}) = E_o(\varepsilon_p - \varepsilon_{med})$$

$$A = R^3\frac{(\varepsilon_p - \varepsilon_{med})}{(\varepsilon_p + 2\varepsilon_{med})}E_o$$

$$B = E_o - AR^{-3}$$

$$B = E_o - \left(R^3\frac{(\varepsilon_p - \varepsilon_{med})}{(\varepsilon_p + 2\varepsilon_{med})}E_o\right)R^{-3}$$

$$B = E_o\left(1 - \frac{\varepsilon_p - \varepsilon_{med}}{\varepsilon_p + 2\varepsilon_{med}}\right)$$

$$B = E_o\left(\frac{2\varepsilon_{med} + \varepsilon_p - (\varepsilon_p - \varepsilon_{med})}{2\varepsilon_{med} + \varepsilon_p}\right)$$

$$B = \frac{E_o 3\varepsilon_{med}}{\varepsilon_p + 2\varepsilon_{med}}$$

The point dipole can be found now. The first term from the electrostatic potential from before is equated to the second term of the potential in the medium.

$$\frac{p\cos(\theta)}{4\pi\varepsilon_0\varepsilon_{med}R^2} = \frac{A\cos(\theta)}{R^2}$$

$$p = 4\pi\varepsilon_0\varepsilon_{med}R^3\left(\frac{\varepsilon_p - \varepsilon_{med}}{\varepsilon_p + 2\varepsilon_{med}}\right)E_o$$

Finally the dielectrophoretic force on a spherical lossless neutral particle can be described as:

$$\overline{F}_{dipole} = \overline{p} \bullet \nabla\overline{E}$$

$$\overline{F}_{dipole} = 4\pi\varepsilon_0\varepsilon_{med}R^3\left(\frac{\varepsilon_p - \varepsilon_{med}}{\varepsilon_p + 2\varepsilon_{med}}\right)E_o \bullet \nabla\overline{E}$$

Since:

$$E_o \bullet \nabla\overline{E} = \frac{1}{2}\nabla^2 E_{rms}$$

The time-averaged dielectrophoretic force for a neutral lossless spherical particle is:

$$\left\langle F_{dep} \right\rangle = 2\pi\varepsilon_0\varepsilon_{med}R^3\left(\frac{\varepsilon_p - \varepsilon_{med}}{\varepsilon_p + 2\varepsilon_{med}}\right)\nabla^2 E_{rms}$$

By extension, the dielectrophoretic force for a neutral and lossy spherical particle is:

$$\left\langle F_{dep} \right\rangle = 2\pi\varepsilon_0\varepsilon_{med}R^3\,\mathrm{Re}\left[\frac{\varepsilon_p^* - \varepsilon_{med}^*}{\varepsilon_p^* + 2\varepsilon_{med}^*}\right]\nabla^2 E_{rms}$$

Where:

$$\varepsilon_i^* = \varepsilon_i + \frac{\sigma_i}{j\omega}$$

The time-averaged dielectrophoretic force for a neutral and lossy spherical particle can be rewritten as:

$$\left\langle F_{dep} \right\rangle = 2\pi\varepsilon_0\varepsilon_{med}R^3\,\mathrm{Re}\left[K(\omega)\right]\nabla^2 E_{rms}$$

Where K(ω) is referred to as the Clausius-Mossotti relationship.

[1] T.B. Jones, *Electromechanics of Particles,* Cambridge University Press, Cambridge, 1995

[2] D.K. Cheng, *Field and Wave Electromagnetics 2nd Edition,* New York: Addison-Wesley Publishing Company, 1992

Appendix B: Matlab Code for Plotting DEP Spectra

```
% nm_beads_yeast.m
% Adapted from code by Sean Forrest Romanuik
% Aug. 24, 2009
% Dept of Electrical & Computer Engineering - University
of Manitoba

clear all
close all
clc
%
============================================================
============
% UNIVERSAL CONSTANT INITIALIZATION
%
============================================================
============
eps_0 = 8.854e-12; % permittivity of free space [F/m]
j = sqrt(-1); % unit imaginary number
%
============================================================
============
% FREQUENCY VARIABLE INITIALIZATION
%
============================================================
============
f = 10.^(0.1:0.001:10); % radial frequency [Hz]
w = (2*pi).*f; % angular frequency [rad/s]
%
============================================================
============
% FLUID MEDIUM INITIALIZATION
%
============================================================
============
eps_prime_fm = 78*eps_0; % real abs. fluid permittivity
[F/m]
cond_DI_H2O = 18e-4; % DI H2O conductivity [S/m]
% complex abs. DI H2O permittivity [F/m]
eps_bar_DI_H2O = eps_prime_fm - j.*cond_DI_H2O./w;
```

```
%
============================================================
============
% PSS INITIALIZATION
%
============================================================
============
eps_prime_pss = 2.5*eps_0; % real abs. PSS permittivity
[F/m]
cond_pss = 2e-4; % PSS conductivity [S/m]
% complex abs. PSS permittivity [F/m]
eps_bar_pss = eps_prime_pss - j.*cond_pss./w;
%
============================================================
============
% Re{K} CALCULATION
%
============================================================
============
K_bar_DI_H2O = ( eps_bar_pss - eps_bar_DI_H2O ) ...
./ ( eps_bar_pss + 2.*eps_bar_DI_H2O );
Re_K_bar_DI_H2O = real(K_bar_DI_H2O);
%
============================================================
============
% Re{K} PLOTTING
%
============================================================
============
plot(log10(f),Re_K_bar_DI_H2O,'color','r','LineWidth',
3);
hold on
xlabel('log_1_0 | \itf\rm / Hz |', 'FontSize', 15)
ylabel('Re\{\itK\rm\}', 'FontSize', 15)
title('DEP Spectra of Polystyrene Beads and Yeast in
Deionized Water', 'FontSize',15)



% Yeast_DEP_Spectra.m
% Written by Sean Forrest Romanuik
```

```
% Aug. 24, 2009
% Dept of Electrical & Computer Engineering - University
of Manitoba

%
============================================================
============
% FLUID MEDIUM INITIALIZATION
%
============================================================
============
eps_prime_fm = 78*eps_0; % real abs. fluid permittivity
[F/m]
cond_fm = 18e-4; % fluid conductivity [S/m]
% complex abs. fluid permittivity [F/m]
eps_bar_fm = eps_prime_fm - j.*cond_fm./w;
%
============================================================
============
% VIABLE YEAST COMPUTATIONS
%
============================================================
============
% yeast cell dimensions [m]
r_cyt = 3e-6; % cytoplasmic sphere radius
d_mem = 3.5e-9; % cytoplasmic membrane thickness
d_pss = 25e-9; % periplasmic space thickness
d_icw = 110e-9; % inner cell wall thickness
d_ocw = 50e-9; % inner cell wall thickness
% yeast cell real absolute permittivities [F/m]
eps_prime_cyt = 51*eps_0; % cytoplasmic sphere
eps_prime_mem = 3*eps_0; % cytoplasmic membrane
eps_prime_pss = 14.4*eps_0; % periplasmic space
eps_prime_icw = 60*eps_0; % inner cell wall
eps_prime_ocw = 5.9*eps_0; % outer cell wall
% yeast cell conductivities [S/m]
cond_cyt = 12000e-4; % cytoplasmic sphere
cond_mem = 0.0302e-4; % cytoplasmic membrane
cond_pss = 41e-4; % periplasmic space
cond_icw = 30.4322e-5; % inner cell wall
cond_ocw = 200e-4; % outer cell wall
```

```
% yeast cell complex absolute permittivities [F/m]
eps_bar_cyt = eps_prime_cyt - j.*cond_cyt./w; %
cytoplasmic sphere
eps_bar_mem = eps_prime_mem - j.*cond_mem./w; % plasma
membrane
eps_bar_pss = eps_prime_pss - j.*cond_pss./w; %
periplasmic space
eps_bar_icw = eps_prime_icw - j.*cond_icw./w; % inner
cell wall
eps_bar_ocw = eps_prime_ocw - j.*cond_ocw./w; % outer
cell wall
% combine the cytoplasmic sphere & the plasma
% membrane into equivalent homogeneous sphere a
a = ( r_cyt + d_mem ) / r_cyt;
K_bar_a = ( eps_bar_cyt - eps_bar_mem ) ...
./ ( eps_bar_cyt + 2.*eps_bar_mem );
eps_bar_a = eps_bar_mem .* ( a^3 + 2.*K_bar_a ) ./ ( a^3
- K_bar_a );
% combine sphere a & the periplasmic space
% layer into equivalent homogeneous sphere b
b = ( a + d_pss ) / a;
K_bar_b = ( eps_bar_a - eps_bar_pss ) ...
./ ( eps_bar_a + 2.*eps_bar_pss );
eps_bar_b = eps_bar_pss .* ( b^3 + 2.*K_bar_b ) ./ ( b^3
- K_bar_b );
% combine sphere b & the inner cell
% wall into equivalent homogeneous sphere c
c = ( b + d_icw ) / b;
K_bar_c = ( eps_bar_b - eps_bar_icw ) ...
./ ( eps_bar_b + 2.*eps_bar_icw );
eps_bar_c = eps_bar_icw .* ( c^3 + 2.*K_bar_c ) ./ ( c^3
- K_bar_c );
% combine sphere c & the outer cell wall into
% equivalent homogeneous sphere d (the final sphere)
d = ( c + d_ocw ) / c;
K_bar_d = ( eps_bar_c - eps_bar_ocw ) ...
./ ( eps_bar_c + 2.*eps_bar_ocw );
eps_bar_d = eps_bar_ocw .* ( d^3 + 2.*K_bar_d ) ...
./ ( d^3 - K_bar_d );
% compute the complex Clausius-Mossotti factor & its real
part
```

```
K_bar_via = ( eps_bar_d - eps_bar_fm ) ...
./ ( eps_bar_d + 2.*eps_bar_fm );
Re_K_bar_via = real(K_bar_via);
% estimate the cross-over frequencies
i_co1 = find( abs(Re_K_bar_via)<0.0002, 1, 'first' );
i_co2 = find( abs(Re_K_bar_via)<0.0002, 1, 'last' );
disp('Viable DEP Spectrum Cross-Over Frequencies [GHz]:')
f_co1 = f(i_co1)
f_co2 = f(i_co2)
% compute and display the approximated RF features
% under the cytoplasmic simplification
disp('1st-Order Approx. of Viable DEP Spectrum
Features:')
K_inf = ( eps_prime_cyt - eps_prime_fm ) ./ (
eps_prime_cyt + 2.*eps_prime_fm )
tau_prime_MW2 = ( eps_prime_cyt + 2.*eps_prime_fm ) ./ (
cond_cyt + 2.*cond_fm );
f_prime_MW2 = 1./(2.*pi.*tau_prime_MW2)
Re_K_MF = ( cond_cyt - cond_fm ) ./ ( cond_cyt +
2.*cond_fm )
%
============================================================
============
% DEP Spectra Plotting
%
============================================================
============
% plot the real part of the complex Clausius-Mossotti
factor
plot(log10(f),Re_K_bar_via,'color','b','LineWidth', 3);
legend('Polystyrene Beads','Yeast')
```

Appendix C: Visual Basic Code for DEP Device Test Automation

```
Imports System
Imports System.Drawing
Imports System.Collections
Imports System.ComponentModel
Imports System.Windows.Forms
Imports System.Data
Imports NationalInstruments.NI4882
Imports System.IO
Imports System.Threading

Namespace NationalInstruments.DEP3
    Public Class DepForm
        Inherits System.Windows.Forms.Form

        'GPIB CONSTANTS
        Private Thurlby As Device
        Const THURLBY_BOARD_ID = 0
        Const THURLBY_PRIMARY_ADD = 5
        Const THURLBY_SECONDARY_ADD = 0

        Private Switch As Device
        Const SWITCH_BOARD_ID = 0
        Const SWITCH_PRIMARY_ADD = 7
        Const SWITCH_SECONDARY_ADD = 0
        Const SWITCH_SETTLING_TIME = 0

        'DEP SET UP CONSTANTS
        Const SIGNAL_AMP_MAX = 20          'volts, peak to peak
        Const SIGNAL_AMP_MIN = 0.1
        Const SIGNAL_FREQ_MAX = 16000      'khz
        Const SIGNAL_FREQ_MIN = 0.01
        Const SIGNAL_PHASE_MAX = 360       'in degrees
        Const SIGNAL_PHASE_MIN = -360
        Const MOVE_TIME_MAX = 100        'in seconds
        Const MOVE_TIME_MIN = 1
        Const SENSE_TIME_MAX = 100
        Const SENSE_TIME_MIN = 10
        Const TOTAL_TIME_MAX = 60        'in minutes
        Const TOTAL_TIME_MIN = 1

        Const NUM_POINTS_MEASURE = 50
        Const INTERVAL_TIME = 0.01
        Const SIGNAL_SETTLE = 5
        Private Path As String
        Private FileExtension As String

        'Threads
        Private DriveTogetherCenterThread As Thread
        Private DriveTogetherRightThread As Thread
        Private DriveTogetherLeftThread As Thread

        Private DriveTogetherNewRightThread As Thread
        Private DriveTogetherNewLeftThread As Thread

        Private StopThread As Thread


#Region " Windows Form Designer generated code "
```

```
Public Sub New()
    MyBase.New()

    'This call is required by the Windows Form Designer.
    InitializeComponent()

    'Add any initialization after the InitializeComponent() call

End Sub

'Form overrides dispose to clean up the component list.
Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
        If Not (components Is Nothing) Then
            components.Dispose()
        End If
    End If
    MyBase.Dispose(disposing)
End Sub

'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents SensorGroupBox As System.Windows.Forms.GroupBox
Friend WithEvents StartStopGroupBox As System.Windows.Forms.GroupBox
Friend WithEvents StartButton As System.Windows.Forms.Button
Friend WithEvents StopButton As System.Windows.Forms.Button
Friend WithEvents ParametersGroupBox As System.Windows.Forms.GroupBox
Friend WithEvents ParametersTabControl As System.Windows.Forms.TabControl
Friend WithEvents ConnectionGroupBox As System.Windows.Forms.GroupBox
Friend WithEvents APadLabel As System.Windows.Forms.Label
Friend WithEvents BPadLabel As System.Windows.Forms.Label
Friend WithEvents CPadLabel As System.Windows.Forms.Label
Friend WithEvents DPadLabel As System.Windows.Forms.Label
Friend WithEvents EPadLabel As System.Windows.Forms.Label
Friend WithEvents FPadLabel As System.Windows.Forms.Label
Friend WithEvents GPadLabel As System.Windows.Forms.Label
'Friend WithEvents DriveApart As System.Windows.Forms.TabPage
Friend WithEvents GroupBox7 As System.Windows.Forms.GroupBox
Friend WithEvents Label21 As System.Windows.Forms.Label
Friend WithEvents Label23 As System.Windows.Forms.Label
Friend WithEvents GroupBox6 As System.Windows.Forms.GroupBox
Friend WithEvents Label18 As System.Windows.Forms.Label
Friend WithEvents Label19 As System.Windows.Forms.Label
Friend WithEvents Label20 As System.Windows.Forms.Label
Friend WithEvents GroupBox5 As System.Windows.Forms.GroupBox
Friend WithEvents Label15 As System.Windows.Forms.Label
Friend WithEvents Label16 As System.Windows.Forms.Label
Friend WithEvents Label17 As System.Windows.Forms.Label
Friend WithEvents Label13 As System.Windows.Forms.Label
Friend WithEvents Label7 As System.Windows.Forms.Label
Friend WithEvents Label3 As System.Windows.Forms.Label
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents Together_TimeBox As System.Windows.Forms.GroupBox
Friend WithEvents Together_DurationNumBox As numberBox.NumericText
Friend WithEvents Together_MoveNumBox As numberBox.NumericText
Friend WithEvents Together_SinBox As System.Windows.Forms.GroupBox
```

```
        Friend WithEvents Together_FreqNumBox As numberBox.NumericText
        Friend WithEvents Together_AmpNumBox As numberBox.NumericText
        Friend WithEvents Together_PhaseNumBox As numberBox.NumericText
        Friend WithEvents DriveTogetherTab As System.Windows.Forms.TabPage
        Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
        Friend WithEvents Label4 As System.Windows.Forms.Label
        Friend WithEvents Label5 As System.Windows.Forms.Label
        Friend WithEvents SS6_Left_FreqNumBox As numberBox.NumericText
        Friend WithEvents SS6_Left_AmpNumBox As numberBox.NumericText
        Friend WithEvents SS6_Left_PhaseNumBox As numberBox.NumericText
        Friend WithEvents Label6 As System.Windows.Forms.Label
        Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
        Friend WithEvents Label8 As System.Windows.Forms.Label
        Friend WithEvents Label9 As System.Windows.Forms.Label
        Friend WithEvents SS6_Right_FreqNumBox As numberBox.NumericText
        Friend WithEvents SS6_Right_AmpNumBox As numberBox.NumericText
        Friend WithEvents SS6_Right_PhaseNumBox As numberBox.NumericText
        Friend WithEvents Label10 As System.Windows.Forms.Label
        Friend WithEvents PictureBox1 As System.Windows.Forms.PictureBox
        Friend WithEvents PictureBox2 As System.Windows.Forms.PictureBox
        Friend WithEvents PictureBox3 As System.Windows.Forms.PictureBox
        Friend WithEvents LeftRadioButton As System.Windows.Forms.RadioButton
        Friend WithEvents CenterRadioButton As System.Windows.Forms.RadioButton
        Friend WithEvents RightRadioButton As System.Windows.Forms.RadioButton
        Friend WithEvents NewRightRadioButton As System.Windows.Forms.RadioButton
        Friend WithEvents NewLeftRadioButton As System.Windows.Forms.RadioButton
        <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
            Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(DepForm))
            Me.SensorGroupBox = New System.Windows.Forms.GroupBox
            Me.NewLeftRadioButton = New System.Windows.Forms.RadioButton
            Me.NewRightRadioButton = New System.Windows.Forms.RadioButton
            Me.PictureBox3 = New System.Windows.Forms.PictureBox
            Me.PictureBox2 = New System.Windows.Forms.PictureBox
            Me.PictureBox1 = New System.Windows.Forms.PictureBox
            Me.LeftRadioButton = New System.Windows.Forms.RadioButton
            Me.RightRadioButton = New System.Windows.Forms.RadioButton
            Me.CenterRadioButton = New System.Windows.Forms.RadioButton
            Me.StartStopGroupBox = New System.Windows.Forms.GroupBox
            Me.StopButton = New System.Windows.Forms.Button
            Me.StartButton = New System.Windows.Forms.Button
            Me.ParametersGroupBox = New System.Windows.Forms.GroupBox
            Me.ParametersTabControl = New System.Windows.Forms.TabControl
            Me.DriveTogetherTab = New System.Windows.Forms.TabPage
            Me.Together_TimeBox = New System.Windows.Forms.GroupBox
            Me.Label13 = New System.Windows.Forms.Label
            Me.Together_DurationNumBox = New numberBox.NumericText
            Me.Label7 = New System.Windows.Forms.Label
            Me.Together_MoveNumBox = New numberBox.NumericText
            Me.Together_SinBox = New System.Windows.Forms.GroupBox
            Me.Label3 = New System.Windows.Forms.Label
            Me.Label1 = New System.Windows.Forms.Label
            Me.Together_FreqNumBox = New numberBox.NumericText
            Me.Together_AmpNumBox = New numberBox.NumericText
            Me.Together_PhaseNumBox = New numberBox.NumericText
            Me.Label2 = New System.Windows.Forms.Label
            Me.GroupBox7 = New System.Windows.Forms.GroupBox
            Me.GroupBox1 = New System.Windows.Forms.GroupBox
            Me.Label4 = New System.Windows.Forms.Label
            Me.Label5 = New System.Windows.Forms.Label
            Me.SS6_Left_FreqNumBox = New numberBox.NumericText
            Me.SS6_Left_AmpNumBox = New numberBox.NumericText
```

```
Me.SS6_Left_PhaseNumBox = New numberBox.NumericText
Me.Label6 = New System.Windows.Forms.Label
Me.GroupBox2 = New System.Windows.Forms.GroupBox
Me.Label8 = New System.Windows.Forms.Label
Me.Label9 = New System.Windows.Forms.Label
Me.SS6_Right_FreqNumBox = New numberBox.NumericText
Me.SS6_Right_AmpNumBox = New numberBox.NumericText
Me.SS6_Right_PhaseNumBox = New numberBox.NumericText
Me.Label10 = New System.Windows.Forms.Label
Me.Label21 = New System.Windows.Forms.Label
Me.ApartDurationNumBox = New numberBox.NumericText
Me.Label23 = New System.Windows.Forms.Label
Me.ApartMoveNumBox = New numberBox.NumericText
Me.GroupBox6 = New System.Windows.Forms.GroupBox
Me.Label18 = New System.Windows.Forms.Label
Me.Label19 = New System.Windows.Forms.Label
Me.ApartLeft_FreqNumBox = New numberBox.NumericText
Me.ApartLeft_AmpNumBox = New numberBox.NumericText
Me.ApartLeft_PhaseNumBox = New numberBox.NumericText
Me.Label20 = New System.Windows.Forms.Label
Me.GroupBox5 = New System.Windows.Forms.GroupBox
Me.Label15 = New System.Windows.Forms.Label
Me.Label16 = New System.Windows.Forms.Label
Me.ApartRight_FreqNumBox = New numberBox.NumericText
Me.ApartRight_AmpNumBox = New numberBox.NumericText
Me.ApartRight_PhaseNumBox = New numberBox.NumericText
Me.Label17 = New System.Windows.Forms.Label
Me.ConnectionGroupBox = New System.Windows.Forms.GroupBox
Me.GPadLabel = New System.Windows.Forms.Label
Me.FPadLabel = New System.Windows.Forms.Label
Me.EPadLabel = New System.Windows.Forms.Label
Me.DPadLabel = New System.Windows.Forms.Label
Me.CPadLabel = New System.Windows.Forms.Label
Me.BPadLabel = New System.Windows.Forms.Label
Me.APadLabel = New System.Windows.Forms.Label
Me.SensorGroupBox.SuspendLayout()
Me.StartStopGroupBox.SuspendLayout()
Me.ParametersGroupBox.SuspendLayout()
Me.ParametersTabControl.SuspendLayout()
Me.DriveTogetherTab.SuspendLayout()
Me.Together_TimeBox.SuspendLayout()
Me.Together_SinBox.SuspendLayout()
Me.GroupBox7.SuspendLayout()
Me.GroupBox1.SuspendLayout()
Me.GroupBox2.SuspendLayout()
Me.GroupBox6.SuspendLayout()
Me.GroupBox5.SuspendLayout()
Me.ConnectionGroupBox.SuspendLayout()
Me.SuspendLayout()
'
'SensorGroupBox
'
Me.SensorGroupBox.Controls.Add(Me.NewLeftRadioButton)
Me.SensorGroupBox.Controls.Add(Me.NewRightRadioButton)
Me.SensorGroupBox.Controls.Add(Me.PictureBox3)
Me.SensorGroupBox.Controls.Add(Me.PictureBox2)
Me.SensorGroupBox.Controls.Add(Me.PictureBox1)
Me.SensorGroupBox.Controls.Add(Me.LeftRadioButton)
Me.SensorGroupBox.Controls.Add(Me.RightRadioButton)
Me.SensorGroupBox.Controls.Add(Me.CenterRadioButton)
Me.SensorGroupBox.Location = New System.Drawing.Point(8, 8)
Me.SensorGroupBox.Name = "SensorGroupBox"
```

```
            Me.SensorGroupBox.Size = New System.Drawing.Size(296, 552)
            Me.SensorGroupBox.TabIndex = 0
            Me.SensorGroupBox.TabStop = False
            Me.SensorGroupBox.Text = "Choose Sensor Type"
            '
            'NewLeftRadioButton
            '
            Me.NewLeftRadioButton.Location = New System.Drawing.Point(160, 512)
            Me.NewLeftRadioButton.Name = "NewLeftRadioButton"
            Me.NewLeftRadioButton.TabIndex = 11
            Me.NewLeftRadioButton.Text = "New Left"
            '
            'NewRightRadioButton
            '
            Me.NewRightRadioButton.Location = New System.Drawing.Point(160, 336)
            Me.NewRightRadioButton.Name = "NewRightRadioButton"
            Me.NewRightRadioButton.Size = New System.Drawing.Size(112, 24)
            Me.NewRightRadioButton.TabIndex = 10
            Me.NewRightRadioButton.Text = "New Right"
            '
            'PictureBox3
            '
            Me.PictureBox3.Image = CType(resources.GetObject("PictureBox3.Image"),
System.Drawing.Image)
            Me.PictureBox3.Location = New System.Drawing.Point(24, 360)
            Me.PictureBox3.Name = "PictureBox3"
            Me.PictureBox3.Size = New System.Drawing.Size(248, 152)
            Me.PictureBox3.TabIndex = 6
            Me.PictureBox3.TabStop = False
            '
            'PictureBox2
            '
            Me.PictureBox2.Image = CType(resources.GetObject("PictureBox2.Image"),
System.Drawing.Image)
            Me.PictureBox2.Location = New System.Drawing.Point(24, 184)
            Me.PictureBox2.Name = "PictureBox2"
            Me.PictureBox2.Size = New System.Drawing.Size(256, 152)
            Me.PictureBox2.TabIndex = 5
            Me.PictureBox2.TabStop = False
            '
            'PictureBox1
            '
            Me.PictureBox1.Image = CType(resources.GetObject("PictureBox1.Image"),
System.Drawing.Image)
            Me.PictureBox1.Location = New System.Drawing.Point(24, 16)
            Me.PictureBox1.Name = "PictureBox1"
            Me.PictureBox1.Size = New System.Drawing.Size(248, 144)
            Me.PictureBox1.TabIndex = 4
            Me.PictureBox1.TabStop = False
            '
            'LeftRadioButton
            '
            Me.LeftRadioButton.Location = New System.Drawing.Point(24, 512)
            Me.LeftRadioButton.Name = "LeftRadioButton"
            Me.LeftRadioButton.TabIndex = 2
            Me.LeftRadioButton.Text = "Drive to the Left"
            '
            'RightRadioButton
            '
            Me.RightRadioButton.Location = New System.Drawing.Point(24, 336)
            Me.RightRadioButton.Name = "RightRadioButton"
            Me.RightRadioButton.Size = New System.Drawing.Size(112, 24)
```

```
Me.RightRadioButton.TabIndex = 1
Me.RightRadioButton.Text = "Drive to the Right"
'
'CenterRadioButton
'
Me.CenterRadioButton.Location = New System.Drawing.Point(24, 160)
Me.CenterRadioButton.Name = "CenterRadioButton"
Me.CenterRadioButton.Size = New System.Drawing.Size(152, 24)
Me.CenterRadioButton.TabIndex = 0
Me.CenterRadioButton.Text = "Drive to the Center"
'
'StartStopGroupBox
'
Me.StartStopGroupBox.Controls.Add(Me.StopButton)
Me.StartStopGroupBox.Controls.Add(Me.StartButton)
Me.StartStopGroupBox.Location = New System.Drawing.Point(312, 624)
Me.StartStopGroupBox.Name = "StartStopGroupBox"
Me.StartStopGroupBox.Size = New System.Drawing.Size(512, 168)
Me.StartStopGroupBox.TabIndex = 1
Me.StartStopGroupBox.TabStop = False
Me.StartStopGroupBox.Text = "Start/Stop"
'
'StopButton
'
Me.StopButton.BackColor = System.Drawing.Color.Red
Me.StopButton.Location = New System.Drawing.Point(288, 24)
Me.StopButton.Name = "StopButton"
Me.StopButton.Size = New System.Drawing.Size(208, 112)
Me.StopButton.TabIndex = 1
Me.StopButton.Text = "STOP"
'
'StartButton
'
Me.StartButton.BackColor = System.Drawing.Color.LimeGreen
Me.StartButton.Location = New System.Drawing.Point(16, 24)
Me.StartButton.Name = "StartButton"
Me.StartButton.Size = New System.Drawing.Size(216, 112)
Me.StartButton.TabIndex = 0
Me.StartButton.Text = "START"
'
'ParametersGroupBox
'
Me.ParametersGroupBox.Controls.Add(Me.ParametersTabControl)
Me.ParametersGroupBox.Location = New System.Drawing.Point(312, 8)
Me.ParametersGroupBox.Name = "ParametersGroupBox"
Me.ParametersGroupBox.Size = New System.Drawing.Size(512, 432)
Me.ParametersGroupBox.TabIndex = 2
Me.ParametersGroupBox.TabStop = False
Me.ParametersGroupBox.Text = "Parameters"
'
'ParametersTabControl
'
Me.ParametersTabControl.Controls.Add(Me.DriveTogetherTab)
Me.ParametersTabControl.Location = New System.Drawing.Point(8, 16)
Me.ParametersTabControl.Name = "ParametersTabControl"
Me.ParametersTabControl.SelectedIndex = 0
Me.ParametersTabControl.Size = New System.Drawing.Size(488, 400)
Me.ParametersTabControl.TabIndex = 0
'
'DriveTogetherTab
'
Me.DriveTogetherTab.Controls.Add(Me.Together_TimeBox)
```

```
            Me.DriveTogetherTab.Controls.Add(Me.Together_SinBox)
            Me.DriveTogetherTab.Location = New System.Drawing.Point(4, 22)
            Me.DriveTogetherTab.Name = "DriveTogetherTab"
            Me.DriveTogetherTab.Size = New System.Drawing.Size(480, 374)
            Me.DriveTogetherTab.TabIndex = 0
            Me.DriveTogetherTab.Text = "Drive Together"
            '
            'Together_TimeBox
            '
            Me.Together_TimeBox.Controls.Add(Me.Label13)
            Me.Together_TimeBox.Controls.Add(Me.Together_DurationNumBox)
            Me.Together_TimeBox.Controls.Add(Me.Label7)
            Me.Together_TimeBox.Controls.Add(Me.Together_MoveNumBox)
            Me.Together_TimeBox.Location = New System.Drawing.Point(8, 208)
            Me.Together_TimeBox.Name = "Together_TimeBox"
            Me.Together_TimeBox.Size = New System.Drawing.Size(456, 144)
            Me.Together_TimeBox.TabIndex = 8
            Me.Together_TimeBox.TabStop = False
            Me.Together_TimeBox.Text = "Timing Values"
            '
            'Label13
            '
            Me.Label13.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label13.Location = New System.Drawing.Point(16, 88)
            Me.Label13.Name = "Label13"
            Me.Label13.Size = New System.Drawing.Size(216, 23)
            Me.Label13.TabIndex = 10
            Me.Label13.Text = "Movement Duration (Min)"
            '
            'Together_DurationNumBox
            '
            Me.Together_DurationNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.Together_DurationNumBox.Location = New System.Drawing.Point(240, 80)
            Me.Together_DurationNumBox.Name = "Together_DurationNumBox"
            Me.Together_DurationNumBox.Size = New System.Drawing.Size(192, 29)
            Me.Together_DurationNumBox.TabIndex = 5
            Me.Together_DurationNumBox.Text = ""
            '
            'Label7
            '
            Me.Label7.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label7.Location = New System.Drawing.Point(16, 40)
            Me.Label7.Name = "Label7"
            Me.Label7.Size = New System.Drawing.Size(216, 23)
            Me.Label7.TabIndex = 7
            Me.Label7.Text = "Movement Time Step (S)"
            '
            'Together_MoveNumBox
            '
            Me.Together_MoveNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.Together_MoveNumBox.Location = New System.Drawing.Point(240, 32)
            Me.Together_MoveNumBox.Name = "Together_MoveNumBox"
            Me.Together_MoveNumBox.Size = New System.Drawing.Size(192, 29)
            Me.Together_MoveNumBox.TabIndex = 3
            Me.Together_MoveNumBox.Text = ""
            '
```

```
            'Together_SinBox
            '
            Me.Together_SinBox.Controls.Add(Me.Label3)
            Me.Together_SinBox.Controls.Add(Me.Label1)
            Me.Together_SinBox.Controls.Add(Me.Together_FreqNumBox)
            Me.Together_SinBox.Controls.Add(Me.Together_AmpNumBox)
            Me.Together_SinBox.Controls.Add(Me.Together_PhaseNumBox)
            Me.Together_SinBox.Controls.Add(Me.Label2)
            Me.Together_SinBox.Location = New System.Drawing.Point(8, 8)
            Me.Together_SinBox.Name = "Together_SinBox"
            Me.Together_SinBox.Size = New System.Drawing.Size(456, 192)
            Me.Together_SinBox.TabIndex = 7
            Me.Together_SinBox.TabStop = False
            Me.Together_SinBox.Text = "Sinusoid Values"
            '
            'Label3
            '
            Me.Label3.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label3.Location = New System.Drawing.Point(24, 136)
            Me.Label3.Name = "Label3"
            Me.Label3.Size = New System.Drawing.Size(208, 23)
            Me.Label3.TabIndex = 6
            Me.Label3.Text = "Phase Shift (Degrees)"
            '
            'Label1
            '
            Me.Label1.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label1.Location = New System.Drawing.Point(24, 88)
            Me.Label1.Name = "Label1"
            Me.Label1.Size = New System.Drawing.Size(152, 23)
            Me.Label1.TabIndex = 5
            Me.Label1.Text = "Frequency (kHz)"
            '
            'Together_FreqNumBox
            '
            Me.Together_FreqNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.Together_FreqNumBox.Location = New System.Drawing.Point(240, 80)
            Me.Together_FreqNumBox.Name = "Together_FreqNumBox"
            Me.Together_FreqNumBox.Size = New System.Drawing.Size(192, 29)
            Me.Together_FreqNumBox.TabIndex = 1
            Me.Together_FreqNumBox.Text = ""
            '
            'Together_AmpNumBox
            '
            Me.Together_AmpNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.Together_AmpNumBox.Location = New System.Drawing.Point(240, 32)
            Me.Together_AmpNumBox.Name = "Together_AmpNumBox"
            Me.Together_AmpNumBox.Size = New System.Drawing.Size(192, 29)
            Me.Together_AmpNumBox.TabIndex = 0
            Me.Together_AmpNumBox.Text = ""
            '
            'Together_PhaseNumBox
            '
            Me.Together_PhaseNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
```

```
            Me.Together_PhaseNumBox.Location = New System.Drawing.Point(240, 128)
            Me.Together_PhaseNumBox.Name = "Together_PhaseNumBox"
            Me.Together_PhaseNumBox.Size = New System.Drawing.Size(192, 29)
            Me.Together_PhaseNumBox.TabIndex = 2
            Me.Together_PhaseNumBox.Text = ""
            '
            'Label2
            '
            Me.Label2.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label2.Location = New System.Drawing.Point(24, 40)
            Me.Label2.Name = "Label2"
            Me.Label2.Size = New System.Drawing.Size(160, 23)
            Me.Label2.TabIndex = 4
            Me.Label2.Text = "Amplitude (V - pp)"
            '
            'GroupBox7
            '
            Me.GroupBox7.Controls.Add(Me.GroupBox1)
            Me.GroupBox7.Controls.Add(Me.GroupBox2)
            Me.GroupBox7.Controls.Add(Me.Label21)
            Me.GroupBox7.Controls.Add(Me.ApartDurationNumBox)
            Me.GroupBox7.Controls.Add(Me.Label23)
            Me.GroupBox7.Controls.Add(Me.ApartMoveNumBox)
            Me.GroupBox7.Location = New System.Drawing.Point(8, 312)
            Me.GroupBox7.Name = "GroupBox7"
            Me.GroupBox7.Size = New System.Drawing.Size(456, 120)
            Me.GroupBox7.TabIndex = 12
            Me.GroupBox7.TabStop = False
            Me.GroupBox7.Text = "Timing Values"
            '
            'GroupBox1
            '
            Me.GroupBox1.Controls.Add(Me.Label4)
            Me.GroupBox1.Controls.Add(Me.Label5)
            Me.GroupBox1.Controls.Add(Me.SS6_Left_FreqNumBox)
            Me.GroupBox1.Controls.Add(Me.SS6_Left_AmpNumBox)
            Me.GroupBox1.Controls.Add(Me.SS6_Left_PhaseNumBox)
            Me.GroupBox1.Controls.Add(Me.Label6)
            Me.GroupBox1.Location = New System.Drawing.Point(0, 64)
            Me.GroupBox1.Name = "GroupBox1"
            Me.GroupBox1.Size = New System.Drawing.Size(456, 144)
            Me.GroupBox1.TabIndex = 12
            Me.GroupBox1.TabStop = False
            Me.GroupBox1.Text = "Sinusoid Values Driving Left"
            '
            'Label4
            '
            Me.Label4.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label4.Location = New System.Drawing.Point(24, 104)
            Me.Label4.Name = "Label4"
            Me.Label4.Size = New System.Drawing.Size(184, 23)
            Me.Label4.TabIndex = 6
            Me.Label4.Text = "Phase Shift (Degrees)"
            '
            'Label5
            '
            Me.Label5.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label5.Location = New System.Drawing.Point(24, 64)
            Me.Label5.Name = "Label5"
```

```
            Me.Label5.Size = New System.Drawing.Size(144, 23)
            Me.Label5.TabIndex = 5
            Me.Label5.Text = "Frequency (kHz)"
            '
            'SS6_Left_FreqNumBox
            '
            Me.SS6_Left_FreqNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.SS6_Left_FreqNumBox.Location = New System.Drawing.Point(240, 64)
            Me.SS6_Left_FreqNumBox.Name = "SS6_Left_FreqNumBox"
            Me.SS6_Left_FreqNumBox.Size = New System.Drawing.Size(192, 29)
            Me.SS6_Left_FreqNumBox.TabIndex = 4
            Me.SS6_Left_FreqNumBox.Text = ""
            '
            'SS6_Left_AmpNumBox
            '
            Me.SS6_Left_AmpNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.SS6_Left_AmpNumBox.Location = New System.Drawing.Point(240, 24)
            Me.SS6_Left_AmpNumBox.Name = "SS6_Left_AmpNumBox"
            Me.SS6_Left_AmpNumBox.Size = New System.Drawing.Size(192, 29)
            Me.SS6_Left_AmpNumBox.TabIndex = 3
            Me.SS6_Left_AmpNumBox.Text = ""
            '
            'SS6_Left_PhaseNumBox
            '
            Me.SS6_Left_PhaseNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.SS6_Left_PhaseNumBox.Location = New System.Drawing.Point(240, 104)
            Me.SS6_Left_PhaseNumBox.Name = "SS6_Left_PhaseNumBox"
            Me.SS6_Left_PhaseNumBox.Size = New System.Drawing.Size(192, 29)
            Me.SS6_Left_PhaseNumBox.TabIndex = 5
            Me.SS6_Left_PhaseNumBox.Text = ""
            '
            'Label6
            '
            Me.Label6.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label6.Location = New System.Drawing.Point(24, 24)
            Me.Label6.Name = "Label6"
            Me.Label6.Size = New System.Drawing.Size(152, 23)
            Me.Label6.TabIndex = 4
            Me.Label6.Text = "Amplitude (V - pp)"
            '
            'GroupBox2
            '
            Me.GroupBox2.Controls.Add(Me.Label8)
            Me.GroupBox2.Controls.Add(Me.Label9)
            Me.GroupBox2.Controls.Add(Me.SS6_Right_FreqNumBox)
            Me.GroupBox2.Controls.Add(Me.SS6_Right_AmpNumBox)
            Me.GroupBox2.Controls.Add(Me.SS6_Right_PhaseNumBox)
            Me.GroupBox2.Controls.Add(Me.Label10)
            Me.GroupBox2.Location = New System.Drawing.Point(0, -88)
            Me.GroupBox2.Name = "GroupBox2"
            Me.GroupBox2.Size = New System.Drawing.Size(456, 144)
            Me.GroupBox2.TabIndex = 11
            Me.GroupBox2.TabStop = False
            Me.GroupBox2.Text = "Sinusoid Values Driving Right"
            '
```

```
            'Label8
            '
            Me.Label8.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label8.Location = New System.Drawing.Point(24, 104)
            Me.Label8.Name = "Label8"
            Me.Label8.Size = New System.Drawing.Size(184, 23)
            Me.Label8.TabIndex = 6
            Me.Label8.Text = "Phase Shift (Degrees)"
            '
            'Label9
            '
            Me.Label9.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label9.Location = New System.Drawing.Point(24, 64)
            Me.Label9.Name = "Label9"
            Me.Label9.Size = New System.Drawing.Size(144, 23)
            Me.Label9.TabIndex = 5
            Me.Label9.Text = "Frequency (kHz)"
            '
            'SS6_Right_FreqNumBox
            '
            Me.SS6_Right_FreqNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.SS6_Right_FreqNumBox.Location = New System.Drawing.Point(240, 64)
            Me.SS6_Right_FreqNumBox.Name = "SS6_Right_FreqNumBox"
            Me.SS6_Right_FreqNumBox.Size = New System.Drawing.Size(192, 29)
            Me.SS6_Right_FreqNumBox.TabIndex = 1
            Me.SS6_Right_FreqNumBox.Text = ""
            '
            'SS6_Right_AmpNumBox
            '
            Me.SS6_Right_AmpNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.SS6_Right_AmpNumBox.Location = New System.Drawing.Point(240, 24)
            Me.SS6_Right_AmpNumBox.Name = "SS6_Right_AmpNumBox"
            Me.SS6_Right_AmpNumBox.Size = New System.Drawing.Size(192, 29)
            Me.SS6_Right_AmpNumBox.TabIndex = 0
            Me.SS6_Right_AmpNumBox.Text = ""
            '
            'SS6_Right_PhaseNumBox
            '
            Me.SS6_Right_PhaseNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.SS6_Right_PhaseNumBox.Location = New System.Drawing.Point(240, 104)
            Me.SS6_Right_PhaseNumBox.Name = "SS6_Right_PhaseNumBox"
            Me.SS6_Right_PhaseNumBox.Size = New System.Drawing.Size(192, 29)
            Me.SS6_Right_PhaseNumBox.TabIndex = 2
            Me.SS6_Right_PhaseNumBox.Text = ""
            '
            'Label10
            '
            Me.Label10.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label10.Location = New System.Drawing.Point(24, 24)
            Me.Label10.Name = "Label10"
            Me.Label10.Size = New System.Drawing.Size(184, 23)
            Me.Label10.TabIndex = 4
            Me.Label10.Text = "Amplitude (V - pp)"
```

```
            '
            'Label21
            '
            Me.Label21.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label21.Location = New System.Drawing.Point(24, 72)
            Me.Label21.Name = "Label21"
            Me.Label21.Size = New System.Drawing.Size(200, 23)
            Me.Label21.TabIndex = 10
            Me.Label21.Text = "Total Duration (Min)"
            '
            'ApartDurationNumBox
            '
            Me.ApartDurationNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.ApartDurationNumBox.Location = New System.Drawing.Point(240, 72)
            Me.ApartDurationNumBox.Name = "ApartDurationNumBox"
            Me.ApartDurationNumBox.Size = New System.Drawing.Size(192, 29)
            Me.ApartDurationNumBox.TabIndex = 8
            Me.ApartDurationNumBox.Text = ""
            '
            'Label23
            '
            Me.Label23.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label23.Location = New System.Drawing.Point(24, 24)
            Me.Label23.Name = "Label23"
            Me.Label23.Size = New System.Drawing.Size(216, 23)
            Me.Label23.TabIndex = 7
            Me.Label23.Text = "Movement Time Step (S)"
            '
            'ApartMoveNumBox
            '
            Me.ApartMoveNumBox.Font = New System.Drawing.Font("Microsoft Sans Serif",
14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
            Me.ApartMoveNumBox.Location = New System.Drawing.Point(240, 24)
            Me.ApartMoveNumBox.Name = "ApartMoveNumBox"
            Me.ApartMoveNumBox.Size = New System.Drawing.Size(192, 29)
            Me.ApartMoveNumBox.TabIndex = 6
            Me.ApartMoveNumBox.Text = ""
            '
            'GroupBox6
            '
            Me.GroupBox6.Controls.Add(Me.Label18)
            Me.GroupBox6.Controls.Add(Me.Label19)
            Me.GroupBox6.Controls.Add(Me.ApartLeft_FreqNumBox)
            Me.GroupBox6.Controls.Add(Me.ApartLeft_AmpNumBox)
            Me.GroupBox6.Controls.Add(Me.ApartLeft_PhaseNumBox)
            Me.GroupBox6.Controls.Add(Me.Label20)
            Me.GroupBox6.Location = New System.Drawing.Point(8, 160)
            Me.GroupBox6.Name = "GroupBox6"
            Me.GroupBox6.Size = New System.Drawing.Size(456, 144)
            Me.GroupBox6.TabIndex = 11
            Me.GroupBox6.TabStop = False
            Me.GroupBox6.Text = "Sinusoid Values Driving Left"
            '
            'Label18
            '
            Me.Label18.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
            Me.Label18.Location = New System.Drawing.Point(24, 104)
            Me.Label18.Name = "Label18"
            Me.Label18.Size = New System.Drawing.Size(184, 23)
            Me.Label18.TabIndex = 6
            Me.Label18.Text = "Phase Shift (Degrees)"
            '
            'Label19
            '
            Me.Label19.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label19.Location = New System.Drawing.Point(24, 64)
            Me.Label19.Name = "Label19"
            Me.Label19.Size = New System.Drawing.Size(144, 23)
            Me.Label19.TabIndex = 5
            Me.Label19.Text = "Frequency (kHz)"
            '
            'ApartLeft_FreqNumBox
            '
            Me.ApartLeft_FreqNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.ApartLeft_FreqNumBox.Location = New System.Drawing.Point(240, 64)
            Me.ApartLeft_FreqNumBox.Name = "ApartLeft_FreqNumBox"
            Me.ApartLeft_FreqNumBox.Size = New System.Drawing.Size(192, 29)
            Me.ApartLeft_FreqNumBox.TabIndex = 4
            Me.ApartLeft_FreqNumBox.Text = ""
            '
            'ApartLeft_AmpNumBox
            '
            Me.ApartLeft_AmpNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.ApartLeft_AmpNumBox.Location = New System.Drawing.Point(240, 24)
            Me.ApartLeft_AmpNumBox.Name = "ApartLeft_AmpNumBox"
            Me.ApartLeft_AmpNumBox.Size = New System.Drawing.Size(192, 29)
            Me.ApartLeft_AmpNumBox.TabIndex = 3
            Me.ApartLeft_AmpNumBox.Text = ""
            '
            'ApartLeft_PhaseNumBox
            '
            Me.ApartLeft_PhaseNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.ApartLeft_PhaseNumBox.Location = New System.Drawing.Point(240, 104)
            Me.ApartLeft_PhaseNumBox.Name = "ApartLeft_PhaseNumBox"
            Me.ApartLeft_PhaseNumBox.Size = New System.Drawing.Size(192, 29)
            Me.ApartLeft_PhaseNumBox.TabIndex = 5
            Me.ApartLeft_PhaseNumBox.Text = ""
            '
            'Label20
            '
            Me.Label20.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label20.Location = New System.Drawing.Point(24, 24)
            Me.Label20.Name = "Label20"
            Me.Label20.Size = New System.Drawing.Size(152, 23)
            Me.Label20.TabIndex = 4
            Me.Label20.Text = "Amplitude (V - pp)"
            '
            'GroupBox5
            '
            Me.GroupBox5.Controls.Add(Me.Label15)
```

```
            Me.GroupBox5.Controls.Add(Me.Label16)
            Me.GroupBox5.Controls.Add(Me.ApartRight_FreqNumBox)
            Me.GroupBox5.Controls.Add(Me.ApartRight_AmpNumBox)
            Me.GroupBox5.Controls.Add(Me.ApartRight_PhaseNumBox)
            Me.GroupBox5.Controls.Add(Me.Label17)
            Me.GroupBox5.Location = New System.Drawing.Point(8, 8)
            Me.GroupBox5.Name = "GroupBox5"
            Me.GroupBox5.Size = New System.Drawing.Size(456, 144)
            Me.GroupBox5.TabIndex = 10
            Me.GroupBox5.TabStop = False
            Me.GroupBox5.Text = "Sinusoid Values Driving Right"
            '
            'Label15
            '
            Me.Label15.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label15.Location = New System.Drawing.Point(24, 104)
            Me.Label15.Name = "Label15"
            Me.Label15.Size = New System.Drawing.Size(184, 23)
            Me.Label15.TabIndex = 6
            Me.Label15.Text = "Phase Shift (Degrees)"
            '
            'Label16
            '
            Me.Label16.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label16.Location = New System.Drawing.Point(24, 64)
            Me.Label16.Name = "Label16"
            Me.Label16.Size = New System.Drawing.Size(144, 23)
            Me.Label16.TabIndex = 5
            Me.Label16.Text = "Frequency (kHz)"
            '
            'ApartRight_FreqNumBox
            '
            Me.ApartRight_FreqNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.ApartRight_FreqNumBox.Location = New System.Drawing.Point(240, 64)
            Me.ApartRight_FreqNumBox.Name = "ApartRight_FreqNumBox"
            Me.ApartRight_FreqNumBox.Size = New System.Drawing.Size(192, 29)
            Me.ApartRight_FreqNumBox.TabIndex = 1
            Me.ApartRight_FreqNumBox.Text = ""
            '
            'ApartRight_AmpNumBox
            '
            Me.ApartRight_AmpNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.ApartRight_AmpNumBox.Location = New System.Drawing.Point(240, 24)
            Me.ApartRight_AmpNumBox.Name = "ApartRight_AmpNumBox"
            Me.ApartRight_AmpNumBox.Size = New System.Drawing.Size(192, 29)
            Me.ApartRight_AmpNumBox.TabIndex = 0
            Me.ApartRight_AmpNumBox.Text = ""
            '
            'ApartRight_PhaseNumBox
            '
            Me.ApartRight_PhaseNumBox.Font = New System.Drawing.Font("Microsoft Sans
Serif", 14.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
            Me.ApartRight_PhaseNumBox.Location = New System.Drawing.Point(240, 104)
            Me.ApartRight_PhaseNumBox.Name = "ApartRight_PhaseNumBox"
            Me.ApartRight_PhaseNumBox.Size = New System.Drawing.Size(192, 29)
```

```
            Me.ApartRight_PhaseNumBox.TabIndex = 2
            Me.ApartRight_PhaseNumBox.Text = ""
            '
            'Label17
            '
            Me.Label17.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label17.Location = New System.Drawing.Point(24, 24)
            Me.Label17.Name = "Label17"
            Me.Label17.Size = New System.Drawing.Size(184, 23)
            Me.Label17.TabIndex = 4
            Me.Label17.Text = "Amplitude (V - pp)"
            '
            'ConnectionGroupBox
            '
            Me.ConnectionGroupBox.Controls.Add(Me.GPadLabel)
            Me.ConnectionGroupBox.Controls.Add(Me.FPadLabel)
            Me.ConnectionGroupBox.Controls.Add(Me.EPadLabel)
            Me.ConnectionGroupBox.Controls.Add(Me.DPadLabel)
            Me.ConnectionGroupBox.Controls.Add(Me.CPadLabel)
            Me.ConnectionGroupBox.Controls.Add(Me.BPadLabel)
            Me.ConnectionGroupBox.Controls.Add(Me.APadLabel)
            Me.ConnectionGroupBox.Location = New System.Drawing.Point(312, 456)
            Me.ConnectionGroupBox.Name = "ConnectionGroupBox"
            Me.ConnectionGroupBox.Size = New System.Drawing.Size(512, 152)
            Me.ConnectionGroupBox.TabIndex = 4
            Me.ConnectionGroupBox.TabStop = False
            Me.ConnectionGroupBox.Text = "Connections"
            '
            'GPadLabel
            '
            Me.GPadLabel.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.GPadLabel.Location = New System.Drawing.Point(240, 120)
            Me.GPadLabel.Name = "GPadLabel"
            Me.GPadLabel.Size = New System.Drawing.Size(184, 23)
            Me.GPadLabel.TabIndex = 6
            Me.GPadLabel.Text = "Pad G:"
            '
            'FPadLabel
            '
            Me.FPadLabel.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.FPadLabel.Location = New System.Drawing.Point(240, 88)
            Me.FPadLabel.Name = "FPadLabel"
            Me.FPadLabel.Size = New System.Drawing.Size(184, 23)
            Me.FPadLabel.TabIndex = 5
            Me.FPadLabel.Text = "Pad F:"
            '
            'EPadLabel
            '
            Me.EPadLabel.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.EPadLabel.Location = New System.Drawing.Point(240, 56)
            Me.EPadLabel.Name = "EPadLabel"
            Me.EPadLabel.Size = New System.Drawing.Size(184, 23)
            Me.EPadLabel.TabIndex = 4
            Me.EPadLabel.Text = "Pad E:"
            '
            'DPadLabel
            '
```

```
            Me.DPadLabel.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.DPadLabel.Location = New System.Drawing.Point(240, 24)
            Me.DPadLabel.Name = "DPadLabel"
            Me.DPadLabel.Size = New System.Drawing.Size(184, 23)
            Me.DPadLabel.TabIndex = 3
            Me.DPadLabel.Text = "Pad D:"
            '
            'CPadLabel
            '
            Me.CPadLabel.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.CPadLabel.Location = New System.Drawing.Point(16, 88)
            Me.CPadLabel.Name = "CPadLabel"
            Me.CPadLabel.Size = New System.Drawing.Size(176, 23)
            Me.CPadLabel.TabIndex = 2
            Me.CPadLabel.Text = "Pad C:"
            '
            'BPadLabel
            '
            Me.BPadLabel.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.BPadLabel.Location = New System.Drawing.Point(16, 56)
            Me.BPadLabel.Name = "BPadLabel"
            Me.BPadLabel.Size = New System.Drawing.Size(176, 23)
            Me.BPadLabel.TabIndex = 1
            Me.BPadLabel.Text = "Pad B:"
            '
            'APadLabel
            '
            Me.APadLabel.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.APadLabel.Location = New System.Drawing.Point(16, 24)
            Me.APadLabel.Name = "APadLabel"
            Me.APadLabel.Size = New System.Drawing.Size(176, 23)
            Me.APadLabel.TabIndex = 0
            Me.APadLabel.Text = "Pad A:"
            '
            'DepForm
            '
            Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
            Me.ClientSize = New System.Drawing.Size(848, 814)
            Me.Controls.Add(Me.ConnectionGroupBox)
            Me.Controls.Add(Me.ParametersGroupBox)
            Me.Controls.Add(Me.StartStopGroupBox)
            Me.Controls.Add(Me.SensorGroupBox)
            Me.Name = "DepForm"
            Me.Text = "Dep Form"
            Me.SensorGroupBox.ResumeLayout(False)
            Me.StartStopGroupBox.ResumeLayout(False)
            Me.ParametersGroupBox.ResumeLayout(False)
            Me.ParametersTabControl.ResumeLayout(False)
            Me.DriveTogetherTab.ResumeLayout(False)
            Me.Together_TimeBox.ResumeLayout(False)
            Me.Together_SinBox.ResumeLayout(False)
            Me.GroupBox7.ResumeLayout(False)
            Me.GroupBox1.ResumeLayout(False)
            Me.GroupBox2.ResumeLayout(False)
            Me.GroupBox6.ResumeLayout(False)
            Me.GroupBox5.ResumeLayout(False)
            Me.ConnectionGroupBox.ResumeLayout(False)
            Me.ResumeLayout(False)
```

```
        End Sub

#End Region

        Private Sub StartButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles StartButton.Click

            Dim FoundErrors As Boolean
            FoundErrors = False

            SensorGroupBox.Enabled = False
            ParametersGroupBox.Enabled = False
            StartButton.Enabled = False

            FoundErrors = FoundErrorsDriveTogetherTab()

            If FoundErrors = False Then
                InitializeDevices()
            End If

            If FoundErrors = False Then

                If CenterRadioButton.Checked = True Then
                    DriveTogetherCenterThread = New Thread(New ThreadStart(AddressOf
Me.DriveTogetherCenterThreadProc))
                    DriveTogetherCenterThread.Priority = ThreadPriority.AboveNormal
                    DriveTogetherCenterThread.IsBackground = True
                    DriveTogetherCenterThread.Start()
                End If

                If RightRadioButton.Checked = True Then
                    DriveTogetherRightThread = New Thread(New ThreadStart(AddressOf
Me.DriveTogetherRightThreadProc))
                    DriveTogetherRightThread.Priority = ThreadPriority.AboveNormal
                    DriveTogetherRightThread.IsBackground = True
                    DriveTogetherRightThread.Start()
                End If


                If NewRightRadioButton.Checked = True Then
                    DriveTogetherNewRightThread = New Thread(New ThreadStart(AddressOf
Me.DriveTogetherNewRightThreadProc))
                    DriveTogetherNewRightThread.Priority = ThreadPriority.AboveNormal
                    DriveTogetherNewRightThread.IsBackground = True
                    DriveTogetherNewRightThread.Start()
                End If


                If LeftRadioButton.Checked = True Then
                    DriveTogetherLeftThread = New Thread(New ThreadStart(AddressOf
Me.DriveTogetherLeftThreadProc))
                    DriveTogetherLeftThread.Priority = ThreadPriority.AboveNormal
                    DriveTogetherLeftThread.IsBackground = True
                    DriveTogetherLeftThread.Start()
                End If

                If NewLeftRadioButton.Checked = True Then
                    DriveTogetherNewLeftThread = New Thread(New ThreadStart(AddressOf
Me.DriveTogetherNewLeftThreadProc))
                    DriveTogetherNewLeftThread.Priority = ThreadPriority.AboveNormal
                    DriveTogetherNewLeftThread.IsBackground = True
```

```
                    DriveTogetherNewLeftThread.Start()
                End If
            End If


            SensorGroupBox.Enabled = True
            ParametersGroupBox.Enabled = True
            StartButton.Enabled = True

        End Sub

        Private Sub DepForm_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Load
            CenterRadioButton.Checked = True

            'set path
            Path = Directory.GetCurrentDirectory + "\"


        End Sub

        Private Function roundToTenThou(ByVal num As Double) As Double
            'rounds value to the nearest ten-thousandths and returns
            roundToTenThou = (System.Math.Round(num * 10000)) / 10000
        End Function

        Private Function roundToThou(ByVal num As Double) As Double
            'rounds value to the nearest thousandths and returns
            roundToThou = (System.Math.Round(num * 1000)) / 1000
        End Function

        Private Function roundToHund(ByVal num As Double) As Double
            'rounds value to the nearest hundredths and returns
            roundToHund = (System.Math.Round(num * 100)) / 100
        End Function

        Private Function roundToTenth(ByVal num As Double) As Double
            'rounds value to the nearest tenths and returns
            roundToTenth = (System.Math.Round(num * 10)) / 10
        End Function

        Private Function FieldEmpty(ByVal s As String) As Boolean
            'returns true if the string sent is empty, false if the field is not empty
            If s.Length = 0 Then
                FieldEmpty = True
            Else
                FieldEmpty = False
            End If

        End Function

        Private Function MultiDec(ByVal s As String) As Boolean
            'returns true if the string sent has multiple decmial points, false if it
doesn't
            If s.IndexOf(".") <> s.LastIndexOf(".") Then
                MultiDec = True
            Else
                MultiDec = False
            End If
        End Function

        Private Function MultiNeg(ByVal s As String) As Boolean
```

```
            'returns true if the string sent has multiple negative signs, false if it
doesn't
            If s.IndexOf("-") <> s.LastIndexOf("-") Then
                MultiNeg = True
            Else
                MultiNeg = False
            End If
        End Function

        Private Function DecExist(ByVal s As String) As Boolean
            'returns true if the string sent has a decmial point, false if it doesn't
            If s.IndexOf(".") = -1 Then
                DecExist = False
            Else
                DecExist = True
            End If
        End Function 'DecExist

        Private Function NegExist(ByVal s As String) As Boolean
            'returns true if the string stored sent has a negative sign, false if it
doesn't
            If s.IndexOf("-") = -1 Then
                NegExist = False
            Else
                NegExist = True
            End If
        End Function 'NegExist

        Private Function SpaceExist(ByVal s As String) As Boolean
            'returns true if the string stored sent has a space, false if it doesn't
            If s.IndexOf(" ") = -1 Then
                SpaceExist = False
            Else
                SpaceExist = True
            End If
        End Function 'SpaceExist

        Private Function NegFirst(ByVal s As String) As Boolean
            'returns true if the string stored sent has a negative sign to begin w/,
false if it doesn't
            If s.IndexOf("-") = 0 Then
                NegFirst = True
            Else
                NegFirst = False
            End If
        End Function 'NegFirst

        Private Function FoundErrorsDriveTogetherTab() As Boolean
            FoundErrorsDriveTogetherTab = False

            Dim current = New Object
            Dim field As String
            Dim fielddbl As Double
            Dim swap As Double

            Try

                ' clear all coloured fields
                Together_AmpNumBox.BackColor = System.Drawing.Color.White
                Together_FreqNumBox.BackColor = System.Drawing.Color.White
                Together_PhaseNumBox.BackColor = System.Drawing.Color.White
                Together_MoveNumBox.BackColor = System.Drawing.Color.White
```

```
                Together_DurationNumBox.BackColor = System.Drawing.Color.White

                '----amp
                current = Together_AmpNumBox
                field = current.text

                If FieldEmpty(field) = True Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And NegExist(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And MultiDec(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And DecExist(field) = True
Then

                    'if there is a decimal point found in the last place in the field,
remove it
                    If (field.IndexOf(".") = field.Length - 1) Then
                        current.text = field.Substring(0, field.Length - 1)
                    End If

                    ' make sure field just doesnt have a "."
                    If field.Length = 0 Or field.Length = 1 Then
                        current.backcolor = System.Drawing.Color.LightSteelBlue
                        FoundErrorsDriveTogetherTab = True
                    End If
                End If

                '----freq
                current = Together_FreqNumBox
                field = current.text

                If FieldEmpty(field) = True Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And NegExist(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And MultiDec(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And DecExist(field) = True
Then

                    'if there is a decimal point found in the last place in the field,
remove it
                    If (field.IndexOf(".") = field.Length - 1) Then
                        current.text = field.Substring(0, field.Length - 1)
                    End If

                    ' make sure field just doesnt have a "."
                    If field.Length = 0 Or field.Length = 1 Then
                        current.backcolor = System.Drawing.Color.LightSteelBlue
                        FoundErrorsDriveTogetherTab = True
                    End If
```

```
                End If


                '----shift
                current = Together_PhaseNumBox
                field = current.text

                If FieldEmpty(field) = True Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And MultiNeg(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And DecExist(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And NegExist(field) = True
Then

                    ' make sure the negative sign is placed in front
                    If NegFirst(field) = False Then
                        current.backcolor = System.Drawing.Color.LightSteelBlue
                        FoundErrorsDriveTogetherTab = True
                    End If

                    ' make sure the field doesnt just have a "-"
                    If field.Length = 1 Then
                        current.backcolor = System.Drawing.Color.LightSteelBlue
                        FoundErrorsDriveTogetherTab = True
                    End If
                End If

                '---- move time step
                current = Together_MoveNumBox
                field = current.text

                If FieldEmpty(field) = True Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And NegExist(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And DecExist(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                End If

                '---- total duration
                current = Together_DurationNumBox
                field = current.text

                If FieldEmpty(field) = True Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                ElseIf FoundErrorsDriveTogetherTab = False And NegExist(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
```

```
                ElseIf FoundErrorsDriveTogetherTab = False And DecExist(field) = True
Then
                    current.backcolor = System.Drawing.Color.LightSteelBlue
                    FoundErrorsDriveTogetherTab = True
                End If

                If FoundErrorsDriveTogetherTab = False Then
                    ' specific value checks
                    fielddbl = Together_AmpNumBox.Text
                    If fielddbl < SIGNAL_AMP_MIN Or fielddbl > SIGNAL_AMP_MAX Then
                        Together_AmpNumBox.BackColor =
System.Drawing.Color.LightSteelBlue
                        FoundErrorsDriveTogetherTab = True
                    End If

                    fielddbl = Together_FreqNumBox.Text
                    If fielddbl < SIGNAL_FREQ_MIN Or fielddbl > SIGNAL_FREQ_MAX Then
                        Together_FreqNumBox.BackColor =
System.Drawing.Color.LightSteelBlue
                        FoundErrorsDriveTogetherTab = True
                    End If

                    fielddbl = Together_PhaseNumBox.Text
                    If fielddbl < SIGNAL_PHASE_MIN Or fielddbl > SIGNAL_PHASE_MAX Then
                        Together_PhaseNumBox.BackColor =
System.Drawing.Color.LightSteelBlue
                        FoundErrorsDriveTogetherTab = True
                    End If

                    fielddbl = Together_MoveNumBox.Text
                    If fielddbl < MOVE_TIME_MIN Or fielddbl > MOVE_TIME_MAX Then
                        Together_MoveNumBox.BackColor =
System.Drawing.Color.LightSteelBlue
                        FoundErrorsDriveTogetherTab = True
                    End If

                    fielddbl = Together_DurationNumBox.Text
                    If fielddbl < TOTAL_TIME_MIN Or fielddbl > TOTAL_TIME_MAX Then
                        Together_DurationNumBox.BackColor =
System.Drawing.Color.LightSteelBlue
                        FoundErrorsDriveTogetherTab = True
                    End If

                End If

                If FoundErrorsDriveTogetherTab = False Then
                    ' round off values if needed

                    fielddbl = Together_AmpNumBox.Text
                    swap = roundToHund(fielddbl)
                    If fielddbl <> swap Then
                        Together_AmpNumBox.Text = swap
                        Together_AmpNumBox.BackColor = System.Drawing.Color.Wheat
                    End If

                    fielddbl = Together_FreqNumBox.Text
                    swap = roundToHund(fielddbl)
                    If fielddbl <> swap Then
                        Together_FreqNumBox.Text = swap
                        Together_FreqNumBox.BackColor = System.Drawing.Color.Wheat
                    End If
```

```
                    'If MeasureRadioButton.Checked = True Then
                    '    fielddbl = SenseNumBox.Text
                    '    swap = roundToTenth(fielddbl)
                    '    If fielddbl <> swap Then
                    '        SenseNumBox.Text = swap
                    '        SenseNumBox.BackColor = System.Drawing.Color.Wheat
                    '    End If
                    'End If
                End If

            Catch ex As Exception
                MessageBox.Show(ex.Message)
                FoundErrorsDriveTogetherTab = True
            Finally
            End Try


        End Function


        Private Sub CenterRadioButton_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles CenterRadioButton.CheckedChanged
            DriveTogetherTab.Enabled = True

            APadLabel.Text = "A: Line 3"
            BPadLabel.Text = "B: Line 1"
            CPadLabel.Text = "C: Lid"
            DPadLabel.Text = "D: Line 2"
            EPadLabel.Text = "E: Center"
            FPadLabel.Text = "F: Lid"
            GPadLabel.Text = " "

            FileExtension = "_center.dat"
        End Sub

        Private Sub RightRadioButton_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RightRadioButton.CheckedChanged
            DriveTogetherTab.Enabled = True

            APadLabel.Text = "A: Line 3"
            BPadLabel.Text = "B: Line 1"
            CPadLabel.Text = "C: Lid"
            DPadLabel.Text = " "
            EPadLabel.Text = "E: Line 2"
            FPadLabel.Text = "F: Lid"
            GPadLabel.Text = "G: Right Most"

            FileExtension = "_right.dat"
        End Sub

        Private Sub LeftRadioButton_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles LeftRadioButton.CheckedChanged
            DriveTogetherTab.Enabled = True

            APadLabel.Text = "A: Line 1"
            BPadLabel.Text = "B: Line 3"
            CPadLabel.Text = "C: Lid"
            DPadLabel.Text = "D: Left Most"
            EPadLabel.Text = "E: Line 2"
            FPadLabel.Text = "F: Lid"
            GPadLabel.Text = " "
```

```
    FileExtension = "_left.dat"
End Sub


Private Sub DriveTogetherCenterThreadProc()

    SensorGroupBox.Enabled = False
    ParametersGroupBox.Enabled = False

    'There are seven touch pads to these sensors.  From left to right
    'on top of the sensor there's A, B and C.  On the bottom, from left
    'to right there's D, E and F.  Some sensors will have an additional pad
    'to the left of F called G.

    'Each sensor has a lid and an target electrode where
    'all the particles are driven towards.  This target electrode
    'is either at the far right, far left or center of the sensor.

    'In addition, there are three other lines that help drive
    'particles towards the target. Line one is closest to the
    'target electrode, then line two and the furthest line is line 3

    'MOVEMENT:
    'Lid and target electrode stay at: SIN (WT + PHASE)
    'Other lines loop as following:
    'STEP       LINE 1              LINE 2              LINE 3
    '1          sin(wt + phase)     sin(wt)             sin(wt + phase)
    '2          sin(wt)             sin(wt)             sin(wt + phase)
    '3          sin(wt)             sin(wt + phase)     sin(wt + phase)
    '4          sin(wt)             sin(wt + phase)     sin(wt)
    '5          sin(wt + phase)     sin(wt + phase)     sin(wt)
    '6          sin(wt + phase)     sin(wt)             sin(wt)
    'then back to step 1

    Dim send As String
    Dim recv As String
    Dim InHz As Double
    Dim fielddbl As Double
    Dim finish As Double

    'card 1 rows and columms
    Dim CARD1 As Integer
    Dim B_ROW_CARD1 As Integer
    Dim E_ROW_CARD1 As Integer
    Dim A_ROW_CARD1 As Integer
    Dim SIN_COL_CARD1 As Integer
    Dim SINPHASE_COL_CARD1 As Integer

    CARD1 = 1
    B_ROW_CARD1 = 1
    E_ROW_CARD1 = 2
    A_ROW_CARD1 = 3
    SIN_COL_CARD1 = 1
    SINPHASE_COL_CARD1 = 2

    'card 2 rows and columns
    Dim CARD2 As Integer
    Dim CENTER_ROW_CARD2 As Integer
    Dim LID_ROW_CARD2 As Integer
    Dim SINPHASE_COLA_CARD2 As Integer
    Dim SINPHASE_COLB_CARD2 As Integer
```

```
CARD2 = 2
CENTER_ROW_CARD2 = 1
LID_ROW_CARD2 = 3
SINPHASE_COLA_CARD2 = 1
SINPHASE_COLB_CARD2 = 2


Try

    'setting up thurlby thandar
    send = "SETUPCH 1"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 2"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 3"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 4"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    'set up phase
    send = "SETUPCH 1"
    Thurlby.Write(send)
    send = "LOCKMODE MASTER"
    Thurlby.Write(send)
    send = "PHASE 0"
    Thurlby.Write(send)
    send = "LOCKSTAT ON"
    Thurlby.Write(send)

    send = "SETUPCH 2"
    Thurlby.Write(send)
    send = "LOCKMODE SLAVE"
    Thurlby.Write(send)
    send = "PHASE " + Together_PhaseNumBox.Text
    Thurlby.Write(send)
    send = "LOCKSTAT ON"
    Thurlby.Write(send)

    send = "SETUPCH 3"
    Thurlby.Write(send)
    send = "LOCKMODE SLAVE"
    Thurlby.Write(send)
    send = "PHASE " + Together_PhaseNumBox.Text
```

```
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

'set up amplitude and frequency
send = "SETUPCH 1"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 2"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 3"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)


'output on
send = "SETUPCH 1"
Thurlby.Write(send)
```

```
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 3"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)

                Thread.Sleep(SIGNAL_SETTLE * 1000) ''wait for signal to settle

                'open all switch matrix lines
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)


                finish = Microsoft.VisualBasic.DateAndTime.Timer +
(Together_DurationNumBox.Text * 60)

                'throught out movement lid stays at sin(wt + phase)
                send = ":ROUT:CLOS (@" + CARD2.ToString + "!" + LID_ROW_CARD2.ToString
+ "!" + SINPHASE_COLA_CARD2.ToString + ")"
                Switch.Write(send)

                'throught out movement target electrode stays at sin(wt + phase)
                send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
CENTER_ROW_CARD2.ToString + "!" + SINPHASE_COLB_CARD2.ToString + ")"
                Switch.Write(send)


                Do

                    'STEP       LINE 1              LINE 2              LINE 3
                    '1          sin(wt + phase)     sin(wt)             sin(wt +
phase)
                    'step 1 ----------------------------------------
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '2          sin(wt)             sin(wt)             sin(wt +
phase)
                    'step 2 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
```

```
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '3          sin(wt)             sin(wt + phase)     sin(wt +
phase)
                    'step 3 ------------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '4          sin(wt)             sin(wt + phase)     sin(wt)
                    'step 4 ------------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
```

```
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '5          sin(wt + phase)     sin(wt + phase)     sin(wt)
                    'step 5 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '6          sin(wt + phase)     sin(wt)             sin(wt)
                    'step 6 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)
```

```
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                Loop While (Microsoft.VisualBasic.DateAndTime.Timer < finish)

                'all switch connections are openned
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)

                'output off
                send = "SETUPCH 1"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 3"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)

            Catch ex As ThreadAbortException
                MessageBox.Show(ex.Message)
            End Try

            SensorGroupBox.Enabled = True
            ParametersGroupBox.Enabled = True

        End Sub
        Private Sub DriveTogetherRightThreadProc()

            SensorGroupBox.Enabled = False
            ParametersGroupBox.Enabled = False

            'There are seven touch pads to these sensors.  From left to right
            'on top of the sensor there's A, B and C.  On the bottom, from left
            'to right there's D, E and F.  Some sensors will have an additional pad
            'to the left of F called G.

            'Each sensor has a lid and an target electrode where
            'all the particles are driven towards.  This target electrode
            'is either at the far right, far left or center of the sensor.

            'In addition, there are three other lines that help drive
            'particles towards the target. Line one is closest to the
            'target electrode, then line two and the furthest line is line 3

            'MOVEMENT:
            'Lid and target electrode stay at: SIN (WT + PHASE)
```

```
'Other lines loop as following:
'STEP       LINE 1              LINE 2              LINE 3
'1          sin(wt + phase)     sin(wt)             sin(wt + phase)
'2          sin(wt)             sin(wt)             sin(wt + phase)
'3          sin(wt)             sin(wt + phase)     sin(wt + phase)
'4          sin(wt)             sin(wt + phase)     sin(wt)
'5          sin(wt + phase)     sin(wt + phase)     sin(wt)
'6          sin(wt + phase)     sin(wt)             sin(wt)
'then back to step 1

Dim send As String
Dim recv As String
Dim InHz As Double
Dim fielddbl As Double
Dim finish As Double

'card 1 rows and columms
Dim CARD1 As Integer
Dim B_ROW_CARD1 As Integer
Dim E_ROW_CARD1 As Integer
Dim A_ROW_CARD1 As Integer
Dim SIN_COL_CARD1 As Integer
Dim SINPHASE_COL_CARD1 As Integer

CARD1 = 1
B_ROW_CARD1 = 1
E_ROW_CARD1 = 2
A_ROW_CARD1 = 3
SIN_COL_CARD1 = 1
SINPHASE_COL_CARD1 = 2

'card 2 rows and columns
Dim CARD2 As Integer
Dim RIGHTMOST_ROW_CARD2 As Integer
Dim LID_ROW_CARD2 As Integer
Dim SINPHASE_COLA_CARD2 As Integer
Dim SINPHASE_COLB_CARD2 As Integer

CARD2 = 2
RIGHTMOST_ROW_CARD2 = 1
LID_ROW_CARD2 = 3
SINPHASE_COLA_CARD2 = 1
SINPHASE_COLB_CARD2 = 2


Try

    'setting up thurlby thandar
    send = "SETUPCH 1"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 2"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)
```

```
send = "SETUPCH 3"
Thurlby.Write(send)
send = "MODE CONT"
Thurlby.Write(send)
send = "WAVE SINE"
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "MODE CONT"
Thurlby.Write(send)
send = "WAVE SINE"
Thurlby.Write(send)

'set up phase
send = "SETUPCH 1"
Thurlby.Write(send)
send = "LOCKMODE MASTER"
Thurlby.Write(send)
send = "PHASE 0"
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 2"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 3"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

'set up amplitude and frequency
send = "SETUPCH 1"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)
```

```
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "AMPUNIT VPP"
                Thurlby.Write(send)
                send = "AMPL " + Together_AmpNumBox.Text
                Thurlby.Write(send)
                fielddbl = Together_FreqNumBox.Text
                InHz = fielddbl * 1000
                send = "WAVFREQ " + InHz.ToString
                Thurlby.Write(send)

                send = "SETUPCH 3"
                Thurlby.Write(send)
                send = "AMPUNIT VPP"
                Thurlby.Write(send)
                send = "AMPL " + Together_AmpNumBox.Text
                Thurlby.Write(send)
                fielddbl = Together_FreqNumBox.Text
                InHz = fielddbl * 1000
                send = "WAVFREQ " + InHz.ToString
                Thurlby.Write(send)

                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "AMPUNIT VPP"
                Thurlby.Write(send)
                send = "AMPL " + Together_AmpNumBox.Text
                Thurlby.Write(send)
                fielddbl = Together_FreqNumBox.Text
                InHz = fielddbl * 1000
                send = "WAVFREQ " + InHz.ToString
                Thurlby.Write(send)


                'output on
                send = "SETUPCH 1"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 3"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)

                Thread.Sleep(SIGNAL_SETTLE * 1000) ''wait for signal to settle

                'open all switch matrix lines
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)


                finish = Microsoft.VisualBasic.DateAndTime.Timer +
(Together_DurationNumBox.Text * 60)
```

```
                'throught out movement lid stays at sin(wt + phase)
                send = ":ROUT:CLOS (@" + CARD2.ToString + "!" + LID_ROW_CARD2.ToString
+ "!" + SINPHASE_COLA_CARD2.ToString + ")"
                Switch.Write(send)

                'throught out movement target electrode stays at sin(wt + phase)
                send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
RIGHTMOST_ROW_CARD2.ToString + "!" + SINPHASE_COLB_CARD2.ToString + ")"
                Switch.Write(send)


                Do

                    'STEP       LINE 1              LINE 2              LINE 3
                    '1          sin(wt + phase)     sin(wt)             sin(wt +
phase)
                    'step 1 ----------------------------------------
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '2          sin(wt)             sin(wt)             sin(wt +
phase)
                    'step 2 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '3          sin(wt)             sin(wt + phase)     sin(wt +
phase)
                    'step 3 ----------------------------------------
```

```
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '4          sin(wt)             sin(wt + phase)     sin(wt)
                    'step 4 ------------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '5          sin(wt + phase)     sin(wt + phase)     sin(wt)
                    'step 5 ------------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
```

```
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '6          sin(wt + phase)     sin(wt)             sin(wt)
                    'step 6 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                Loop While (Microsoft.VisualBasic.DateAndTime.Timer < finish)

                'all switch connections are openned
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)

                'output off
                send = "SETUPCH 1"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
```

```
                Thurlby.Write(send)
                send = "SETUPCH 3"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)

            Catch ex As ThreadAbortException
                MessageBox.Show(ex.Message)
            End Try

            SensorGroupBox.Enabled = True
            ParametersGroupBox.Enabled = True

        End Sub

        Private Sub DriveTogetherNewRightThreadProc()

            SensorGroupBox.Enabled = False
            ParametersGroupBox.Enabled = False

            'There are seven touch pads to these sensors.  From left to right
            'on top of the sensor there's A, B and C.  On the bottom, from left
            'to right there's D, E and F.  Some sensors will have an additional pad
            'to the left of F called G.

            'Each sensor has a lid and an target electrode where
            'all the particles are driven towards.  This target electrode
            'is either at the far right, far left or center of the sensor.

            'In addition, there are three other lines that help drive
            'particles towards the target. Line one is closest to the
            'target electrode, then line two and the furthest line is line 3

            'MOVEMENT:
            'Other lines loop as following:
            'STEP       LINE B              LINE E              LINE A
LID
            '1          sin(wt)             sin(wt)             sin(wt + phase)
user specified sin(wt + phase)
            '2          sin(wt)             sin(wt + phase)     sin(wt)             1V
sin(wt + phase)
            '3          sin(wt)             sin(wt + phase)     sin(wt)
user specified sin(wt + phase)
            '4          sin(wt + phase)     sin(wt)             sin(wt)             1V
sin(wt + phase)
            '5          sin(wt + phase)     sin(wt)             sin(wt)
user specified sin(wt + phase)
            '6          sin(wt)             sin(wt)             sin(wt + phase)     1V
sin(wt + phase)
            'then back to step 1

            Dim send As String
            Dim recv As String
            Dim InHz As Double
            Dim fielddbl As Double
            Dim finish As Double

            'card 1 rows and columms
```

```
Dim CARD1 As Integer
Dim B_ROW_CARD1 As Integer
Dim E_ROW_CARD1 As Integer
Dim A_ROW_CARD1 As Integer
Dim SIN_COL_CARD1 As Integer
Dim SINPHASE_COL_CARD1 As Integer

CARD1 = 1
B_ROW_CARD1 = 1
E_ROW_CARD1 = 2
A_ROW_CARD1 = 3
SIN_COL_CARD1 = 1
SINPHASE_COL_CARD1 = 2

'card 2 rows and columns
Dim CARD2 As Integer
Dim TARGET_ROW_CARD2 As Integer
Dim LID_ROW_CARD2 As Integer
Dim SINPHASE_1V_CARD2 As Integer
Dim SINPHASE_USER_CARD2 As Integer

CARD2 = 2
TARGET_ROW_CARD2 = 1
LID_ROW_CARD2 = 3
SINPHASE_1V_CARD2 = 1         'WILL BE FIXED TO 1 V
SINPHASE_USER_CARD2 = 2         'WILL BE USER SPECIFIED


Try

    'setting up thurlby thandar
    send = "SETUPCH 1"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 2"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 3"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 4"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    'set up phase
    send = "SETUPCH 1"
    Thurlby.Write(send)
```

```
send = "LOCKMODE MASTER"
Thurlby.Write(send)
send = "PHASE 0"
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 2"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 3"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

'set up amplitude and frequency
send = "SETUPCH 1"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 2"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 3"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL 1" '+ Together_AmpNumBox.Text
```

```
                Thurlby.Write(send)
                fielddbl = Together_FreqNumBox.Text
                InHz = fielddbl * 1000
                send = "WAVFREQ " + InHz.ToString
                Thurlby.Write(send)

                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "AMPUNIT VPP"
                Thurlby.Write(send)
                send = "AMPL " + Together_AmpNumBox.Text
                Thurlby.Write(send)
                fielddbl = Together_FreqNumBox.Text
                InHz = fielddbl * 1000
                send = "WAVFREQ " + InHz.ToString
                Thurlby.Write(send)


                'output on
                send = "SETUPCH 1"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 3"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)

                Thread.Sleep(SIGNAL_SETTLE * 1000) ''wait for signal to settle

                'open all switch matrix lines
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)


                finish = Microsoft.VisualBasic.DateAndTime.Timer +
(Together_DurationNumBox.Text * 60)

                ''throught out movement lid stays at sin(wt + phase)
                'send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_COLA_CARD2.ToString + ")"
                'Switch.Write(send)

                'throught out movement target electrode stays at sin(wt + phase)
                send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
TARGET_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                Switch.Write(send)


                Do

                    'STEP       LINE B              LINE E              LINE A
LID
```

```
                    '1          sin(wt)             sin(wt)             sin(wt +
phase)     user specified sin(wt + phase)
                    'step 1 ----------------------------------------
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE B              LINE E              LINE A
LID
                    '2          sin(wt)             sin(wt + phase)     sin(wt)
1V sin(wt + phase)
                    'step 2 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE B              LINE E              LINE A
LID
                    '3          sin(wt)             sin(wt + phase)     sin(wt)
user specified sin(wt + phase)
                    'step 3 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
```

```
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE B              LINE E              LINE A
LID
                    '4          sin(wt + phase)     sin(wt)             sin(wt)
1V sin(wt + phase)
                    'step 4 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE B              LINE E              LINE A
LID
```

```
                    '5          sin(wt + phase)     sin(wt)             sin(wt)
user specified sin(wt + phase)
                    'step 5 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE B              LINE E              LINE A
LID
                    '6          sin(wt)             sin(wt)             sin(wt +
phase)     1V sin(wt + phase)
                    'step 6 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
```

```
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                Loop While (Microsoft.VisualBasic.DateAndTime.Timer < finish)

                'all switch connections are openned
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)

                'output off
                send = "SETUPCH 1"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 3"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)

            Catch ex As ThreadAbortException
                MessageBox.Show(ex.Message)
            End Try

            SensorGroupBox.Enabled = True
            ParametersGroupBox.Enabled = True

        End Sub
        Private Sub DriveTogetherNewLeftThreadProc()

            SensorGroupBox.Enabled = False
            ParametersGroupBox.Enabled = False

            'There are seven touch pads to these sensors.  From left to right
            'on top of the sensor there's A, B and C.  On the bottom, from left
            'to right there's D, E and F.  Some sensors will have an additional pad
            'to the left of F called G.

            'Each sensor has a lid and a target electrode where
            'all the particles are driven towards.  This target electrode
            'is either at the far right, far left or center of the sensor.
```

```
            'In addition, there are three other lines that help drive
            'particles towards the target. Line one is closest to the
            'target electrode, then line two and the furthest line is line 3

            'MOVEMENT:
            'Other lines loop as following:
            'STEP       LINE A              LINE E              LINE B
LID
            '1          sin(wt)             sin(wt)             sin(wt + phase)
user specified sin(wt + phase)
            '2          sin(wt)             sin(wt + phase)     sin(wt)             1V
sin(wt + phase)
            '3          sin(wt)             sin(wt + phase)     sin(wt)
user specified sin(wt + phase)
            '4          sin(wt + phase)     sin(wt)             sin(wt)             1V
sin(wt + phase)
            '5          sin(wt + phase)     sin(wt)             sin(wt)
user specified sin(wt + phase)
            '6          sin(wt)             sin(wt)             sin(wt + phase)     1V
sin(wt + phase)
            'then back to step 1

            Dim send As String
            Dim recv As String
            Dim InHz As Double
            Dim fielddbl As Double
            Dim finish As Double

            'card 1 rows and columms
            Dim CARD1 As Integer
            Dim B_ROW_CARD1 As Integer
            Dim E_ROW_CARD1 As Integer
            Dim A_ROW_CARD1 As Integer
            Dim SIN_COL_CARD1 As Integer
            Dim SINPHASE_COL_CARD1 As Integer

            CARD1 = 1
            B_ROW_CARD1 = 1
            E_ROW_CARD1 = 2
            A_ROW_CARD1 = 3
            SIN_COL_CARD1 = 1
            SINPHASE_COL_CARD1 = 2

            'card 2 rows and columns
            Dim CARD2 As Integer
            Dim TARGET_ROW_CARD2 As Integer
            Dim LID_ROW_CARD2 As Integer
            Dim SINPHASE_1V_CARD2 As Integer
            Dim SINPHASE_USER_CARD2 As Integer

            CARD2 = 2
            TARGET_ROW_CARD2 = 1
            LID_ROW_CARD2 = 3
            SINPHASE_1V_CARD2 = 1         'WILL BE FIXED TO 1 V
            SINPHASE_USER_CARD2 = 2         'WILL BE USER SPECIFIED


            Try

                'setting up thurlby thandar
                send = "SETUPCH 1"
```

```
Thurlby.Write(send)
send = "MODE CONT"
Thurlby.Write(send)
send = "WAVE SINE"
Thurlby.Write(send)

send = "SETUPCH 2"
Thurlby.Write(send)
send = "MODE CONT"
Thurlby.Write(send)
send = "WAVE SINE"
Thurlby.Write(send)

send = "SETUPCH 3"
Thurlby.Write(send)
send = "MODE CONT"
Thurlby.Write(send)
send = "WAVE SINE"
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "MODE CONT"
Thurlby.Write(send)
send = "WAVE SINE"
Thurlby.Write(send)

'set up phase
send = "SETUPCH 1"
Thurlby.Write(send)
send = "LOCKMODE MASTER"
Thurlby.Write(send)
send = "PHASE 0"
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 2"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 3"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
```

```
Thurlby.Write(send)

'set up amplitude and frequency
send = "SETUPCH 1"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 2"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 3"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL 1" '+ Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)


'output on
send = "SETUPCH 1"
Thurlby.Write(send)
send = "OUTPUT ON"
Thurlby.Write(send)
send = "SETUPCH 2"
Thurlby.Write(send)
send = "OUTPUT ON"
Thurlby.Write(send)
send = "SETUPCH 3"
Thurlby.Write(send)
send = "OUTPUT ON"
Thurlby.Write(send)
send = "SETUPCH 4"
```

```
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)

                Thread.Sleep(SIGNAL_SETTLE * 1000) ''wait for signal to settle

                'open all switch matrix lines
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)


                finish = Microsoft.VisualBasic.DateAndTime.Timer +
(Together_DurationNumBox.Text * 60)

                ''throught out movement lid stays at sin(wt + phase)
                'send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_COLA_CARD2.ToString + ")"
                'Switch.Write(send)

                'throught out movement target electrode stays at sin(wt + phase)
                send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
TARGET_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                Switch.Write(send)


                Do

                    'STEP       LINE A              LINE E              LINE B
LID
                    '1          sin(wt)             sin(wt)             sin(wt +
phase)     user specified sin(wt + phase)
                    'step 1 ----------------------------------------
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE A              LINE E              LINE B
LID
                    '2          sin(wt)             sin(wt + phase)     sin(wt)
1V sin(wt + phase)
                    'step 2 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
```

```
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE A              LINE E              LINE B
LID
                    '3          sin(wt)             sin(wt + phase)     sin(wt)
user specified sin(wt + phase)
                    'step 3 ------------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE A              LINE E              LINE B
LID
                    '4          sin(wt + phase)     sin(wt)             sin(wt)
1V sin(wt + phase)
                    'step 4 ------------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
```

```
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE A              LINE E              LINE B
LID
                    '5          sin(wt + phase)     sin(wt)             sin(wt)
user specified sin(wt + phase)
                    'step 5 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE A              LINE E              LINE B
LID
```

```
                    '6          sin(wt)             sin(wt)             sin(wt +
phase)     1V sin(wt + phase)
                    'step 6 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_USER_CARD2.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD2.ToString + "!" +
LID_ROW_CARD2.ToString + "!" + SINPHASE_1V_CARD2.ToString + ")"
                    Switch.Write(send)

                Loop While (Microsoft.VisualBasic.DateAndTime.Timer < finish)

                'all switch connections are openned
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)

                'output off
                send = "SETUPCH 1"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 3"
                Thurlby.Write(send)
```

```
            send = "OUTPUT OFF"
            Thurlby.Write(send)
            send = "SETUPCH 4"
            Thurlby.Write(send)
            send = "OUTPUT OFF"
            Thurlby.Write(send)

        Catch ex As ThreadAbortException
            MessageBox.Show(ex.Message)
        End Try

        SensorGroupBox.Enabled = True
        ParametersGroupBox.Enabled = True

    End Sub
    Private Sub DriveTogetherLeftThreadProc()

        SensorGroupBox.Enabled = False
        ParametersGroupBox.Enabled = False

        'There are seven touch pads to these sensors.  From left to right
        'on top of the sensor there's A, B and C.  On the bottom, from left
        'to right there's D, E and F.  Some sensors will have an additional pad
        'to the left of F called G.

        'Each sensor has a lid and an target electrode where
        'all the particles are driven towards.  This target electrode
        'is either at the far right, far left or center of the sensor.

        'In addition, there are three other lines that help drive
        'particles towards the target. Line one is closest to the
        'target electrode, then line two and the furthest line is line 3

        'MOVEMENT:
        'Lid and target electrode stay at: SIN (WT + PHASE)
        'Other lines loop as following:
        'STEP       LINE 1              LINE 2              LINE 3
        '1          sin(wt + phase)     sin(wt)             sin(wt + phase)
        '2          sin(wt)             sin(wt)             sin(wt + phase)
        '3          sin(wt)             sin(wt + phase)     sin(wt + phase)
        '4          sin(wt)             sin(wt + phase)     sin(wt)
        '5          sin(wt + phase)     sin(wt + phase)     sin(wt)
        '6          sin(wt + phase)     sin(wt)             sin(wt)
        'then back to step 1

        Dim send As String
        Dim recv As String
        Dim InHz As Double
        Dim fielddbl As Double
        Dim finish As Double

        'card 1 rows and columms
        Dim CARD1 As Integer
        Dim B_ROW_CARD1 As Integer
        Dim E_ROW_CARD1 As Integer
        Dim A_ROW_CARD1 As Integer
        Dim SIN_COL_CARD1 As Integer
        Dim SINPHASE_COL_CARD1 As Integer

        CARD1 = 1
        B_ROW_CARD1 = 1
        E_ROW_CARD1 = 2
```

```
A_ROW_CARD1 = 3
SIN_COL_CARD1 = 1
SINPHASE_COL_CARD1 = 2

'card 2 rows and columns
Dim CARD2 As Integer
Dim LEFTMOST_ROW_CARD2 As Integer
Dim LID_ROW_CARD2 As Integer
Dim SINPHASE_COLA_CARD2 As Integer
Dim SINPHASE_COLB_CARD2 As Integer

CARD2 = 2
LEFTMOST_ROW_CARD2 = 2
LID_ROW_CARD2 = 3
SINPHASE_COLA_CARD2 = 1
SINPHASE_COLB_CARD2 = 2


Try

    'setting up thurlby thandar
    send = "SETUPCH 1"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 2"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 3"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    send = "SETUPCH 4"
    Thurlby.Write(send)
    send = "MODE CONT"
    Thurlby.Write(send)
    send = "WAVE SINE"
    Thurlby.Write(send)

    'set up phase
    send = "SETUPCH 1"
    Thurlby.Write(send)
    send = "LOCKMODE MASTER"
    Thurlby.Write(send)
    send = "PHASE 0"
    Thurlby.Write(send)
    send = "LOCKSTAT ON"
    Thurlby.Write(send)

    send = "SETUPCH 2"
    Thurlby.Write(send)
    send = "LOCKMODE SLAVE"
```

```
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 3"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "LOCKMODE SLAVE"
Thurlby.Write(send)
send = "PHASE " + Together_PhaseNumBox.Text
Thurlby.Write(send)
send = "LOCKSTAT ON"
Thurlby.Write(send)

'set up amplitude and frequency
send = "SETUPCH 1"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 2"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 3"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
send = "AMPL " + Together_AmpNumBox.Text
Thurlby.Write(send)
fielddbl = Together_FreqNumBox.Text
InHz = fielddbl * 1000
send = "WAVFREQ " + InHz.ToString
Thurlby.Write(send)

send = "SETUPCH 4"
Thurlby.Write(send)
send = "AMPUNIT VPP"
Thurlby.Write(send)
```

```
                send = "AMPL " + Together_AmpNumBox.Text
                Thurlby.Write(send)
                fielddbl = Together_FreqNumBox.Text
                InHz = fielddbl * 1000
                send = "WAVFREQ " + InHz.ToString
                Thurlby.Write(send)


                'output on
                send = "SETUPCH 1"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 3"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)
                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "OUTPUT ON"
                Thurlby.Write(send)

                Thread.Sleep(SIGNAL_SETTLE * 1000) ''wait for signal to settle

                'open all switch matrix lines
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)


                finish = Microsoft.VisualBasic.DateAndTime.Timer +
(Together_DurationNumBox.Text * 60)

                'throught out movement lid stays at sin(wt + phase)
                send = ":ROUT:CLOS (@" + CARD2.ToString + "!" + LID_ROW_CARD2.ToString
+ "!" + SINPHASE_COLA_CARD2.ToString + ")"
                Switch.Write(send)

                'throught out movement target electrode stays at sin(wt + phase)
                send = ":ROUT:CLOS (@" + CARD2.ToString + "!" +
LEFTMOST_ROW_CARD2.ToString + "!" + SINPHASE_COLB_CARD2.ToString + ")"
                Switch.Write(send)


                Do

                    'STEP       LINE 1              LINE 2              LINE 3
                    '1          sin(wt + phase)     sin(wt)             sin(wt +
phase)
                    'step 1 ----------------------------------------
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
```

```
                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '2          sin(wt)             sin(wt)             sin(wt +
phase)
                    'step 2 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '3          sin(wt)             sin(wt + phase)     sin(wt +
phase)
                    'step 3 ----------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '4          sin(wt)             sin(wt + phase)     sin(wt)
                    'step 4 ----------------------------------------
```

```
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '5          sin(wt + phase)     sin(wt + phase)     sin(wt)
                    'step 5 ------------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    'STEP       LINE 1              LINE 2              LINE 3
                    '6          sin(wt + phase)     sin(wt)             sin(wt)
                    'step 6 ------------------------------------------

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
```

```
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:CLOS (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                    Thread.Sleep(Together_MoveNumBox.Text * 1000)

                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
A_ROW_CARD1.ToString + "!" + SINPHASE_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
E_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)
                    send = ":ROUT:OPEN (@" + CARD1.ToString + "!" +
B_ROW_CARD1.ToString + "!" + SIN_COL_CARD1.ToString + ")"
                    Switch.Write(send)

                Loop While (Microsoft.VisualBasic.DateAndTime.Timer < finish)

                'all switch connections are openned
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)

                'output off
                send = "SETUPCH 1"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 2"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 3"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)
                send = "SETUPCH 4"
                Thurlby.Write(send)
                send = "OUTPUT OFF"
                Thurlby.Write(send)

            Catch ex As ThreadAbortException
                MessageBox.Show(ex.Message)
            End Try

            SensorGroupBox.Enabled = True
            ParametersGroupBox.Enabled = True

        End Sub

        Private Sub StopButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles StopButton.Click
            Dim send As String

            If Not (Switch Is Nothing) Then
                send = ":ROUT:OPEN ALL"
                Switch.Write(send)
```

```
            End If

            StopThread = New Thread(New ThreadStart(AddressOf Me.StopThreadProc))
            StopThread.Priority = ThreadPriority.AboveNormal
            StopThread.IsBackground = True
            StopThread.Start()

            SensorGroupBox.Enabled = True
            ParametersGroupBox.Enabled = True
        End Sub

        Private Sub StopThreadProc()

            Try
                If Not (DriveTogetherCenterThread Is Nothing) Then
                    DriveTogetherCenterThread.Abort()
                    DriveTogetherCenterThread.Join()
                    DriveTogetherCenterThread = Nothing
                End If

                If Not (DriveTogetherRightThread Is Nothing) Then
                    DriveTogetherRightThread.Abort()
                    DriveTogetherRightThread.Join()
                    DriveTogetherRightThread = Nothing
                End If

                If Not (DriveTogetherNewRightThread Is Nothing) Then
                    DriveTogetherNewRightThread.Abort()
                    DriveTogetherNewRightThread.Join()
                    DriveTogetherNewRightThread = Nothing
                End If

                If Not (DriveTogetherLeftThread Is Nothing) Then
                    DriveTogetherLeftThread.Abort()
                    DriveTogetherLeftThread.Join()
                    DriveTogetherLeftThread = Nothing
                End If

                If Not (DriveTogetherNewLeftThread Is Nothing) Then
                    DriveTogetherNewLeftThread.Abort()
                    DriveTogetherNewLeftThread.Join()
                    DriveTogetherNewLeftThread = Nothing
                End If

                SensorGroupBox.Enabled = True
                ParametersGroupBox.Enabled = True

            Catch ex As ThreadAbortException
                MessageBox.Show(ex.Message)
            Catch ex As Exception
                MessageBox.Show(ex.Message)
            End Try
        End Sub

        Private Sub InitializeDevices()

            If Switch Is Nothing Then
                Switch = New Device(CInt(SWITCH_BOARD_ID), CByte(SWITCH_PRIMARY_ADD),
CByte(SWITCH_SECONDARY_ADD))
                Switch.EndOfStringCharacter = System.Convert.ToByte(ControlChars.Lf)
                Switch.TerminateReadOnEndOfString = True
                Switch.IOTimeout = TimeoutValue.T3s
```

```
            End If

            If Thurlby Is Nothing Then
                Thurlby = New Device(CInt(THURLBY_BOARD_ID),
CByte(THURLBY_PRIMARY_ADD), CByte(THURLBY_SECONDARY_ADD))
                Thurlby.EndOfStringCharacter = System.Convert.ToByte(ControlChars.Lf)
                Thurlby.TerminateReadOnEndOfString = True
                Thurlby.IOTimeout = TimeoutValue.T3s
            End If

        End Sub

        Private Function InsertCommonEscapeSequences(ByVal s As String) As String
            'takes the end of transmission characters and replaces them
            Return s.Replace(ControlChars.Lf, "\n").Replace(ControlChars.Cr, "\r")
        End Function 'InsertCommonEscapeSequences



        Private Sub NewRightRadioButton_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles NewRightRadioButton.CheckedChanged
            DriveTogetherTab.Enabled = True

            APadLabel.Text = "A: Line 3"
            BPadLabel.Text = "B: Line 1"
            CPadLabel.Text = "C: Lid"
            DPadLabel.Text = " "
            EPadLabel.Text = "E: Line 2"
            FPadLabel.Text = "F: Lid"
            GPadLabel.Text = "G: Right Most"

            FileExtension = "_newright.dat"
        End Sub

        Private Sub NewLeftRadioButton_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles NewLeftRadioButton.CheckedChanged
            DriveTogetherTab.Enabled = True

            APadLabel.Text = "A: Line 1"
            BPadLabel.Text = "B: Line 3"
            CPadLabel.Text = "C: Lid"
            DPadLabel.Text = "D: Left Most"
            EPadLabel.Text = "E: Line 2"
            FPadLabel.Text = "F: Lid"
            GPadLabel.Text = " "

            FileExtension = "_newleft.dat"
        End Sub
    End Class
End Namespace
```