Control and Characterization of Line-Addressable Micromirror Arrays

Harris J. Hall

CONTROL AND CHARACTERIZATION OF LINE-ADDRESSABLE MICROMIRROR ARRAYS

THESIS

Harris J. Hall, Second Lieutenant, USAF

AFIT/GEO/ENG/01M-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GEO/ENG/01M-01

AFIT/GEO/ENG/01M-01

CONTROL AND CHARACTERIZATION OF LINE- ADDRESSABLE MICROMIRROR ARRAYS

THESIS

Presented to the faculty of the Graduate School of Engineering & Management

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Harris Joseph Hall, B. S. E. E.

Second Lieutenant, USAF

March 2001

AFIT/GEO/ENG/01M-01

CONTROL AND CHARACTERIZATION OF LINE- ADDRESSABLE MICROMIRROR ARRAYS

THESIS

Harris Joseph Hall, B. S. E. E.
Second Lieutenant, USAF

Approved:

[illegible]	6 MAR 01
Maj Eric P. Magee, Ph.D. Thesis Advisor	Date
James A. Lott	05 MAR 01
Lt Col James A. Lott, Ph.D. Committee member	Date
William D. Cowan	05 Mar 01
Lt Col William D. Cowan, Ph.D. Committee member	Date
[illegible]	5 MAR 01
Dr. Edward A. Watson, Ph.D. Committee member	Date

ACKNOWLEDGMENTS

"Sometimes a scream is better than a thesis."

-Ralph Waldo Emerson (1803-1882) [38]

I first decided to attend AFIT because I wanted to gain additional expertise and confidence in my career field before beginning my first assignment in the "real" Air Force. Looking back upon my experiences, I realize now that I have not only become a better engineer but a better officer, having gained a very valuable perspective on the many technical and organizational challenges that my generation of officers will have to face in our Air Force's future. The many successes and failures that I have achieved and endured throughout the course of this thesis and course curriculum, will undoubtedly serve me well throughout my career, wherever it may take me. My thanks goes to all the faculty, students, and technicians who took time to answer my many questions in an effort to educate me.

I would like to extend special thanks to my advisor Maj Eric Magee, and committee members Lt Col James Lott, Dr. Edward Watson, and Lt Col William Cowan of AFRL/MLPJ who also served as my sponsor. This thesis would not have been possible without additional technical support from Tom Kensky of AFRL/MLPJ; AFIT technicians Bill Trop, Charles McNeely, and Joe Shaw; lab supervisor Charles Powers; and Aaron Babb of SAIC. I would also like to thank Maj Stanley Rogers (USAFR) for assistance during device testing.

While my time at AFIT was far from the proverbial "picnic", I did have my share of very "good times" and I owe these to the great friends that I have made while stationed here. Their ridiculous antics, love of spirits, and general fun-loving, warm demeanor made this assignment truly memorable. The times we spent together provided the atmosphere I needed to let loose and maintain my motivation despite numerous setbacks.

I prepared this document using Microsoft® Word 2000. I made the illustrations using Adobe Illustrator v. 9.0, Microsoft® PowerPoint 2000, and Visio Technical Drawing Editor. All plots were produced using either Microsoft® Excel 2000 or MATLAB® v6.0. Interferometer images were converted from VHS video format courtesy of Lorinda Hull of Anteon Corporation. SEM Photos were collected with the assistance of Bill Trop.

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

This research involved the design and implementation of a complete line-addressable control system for a 32x32 electrostatic piston-actuated micromirror array device. Line addressing reduces the number of control lines from N^2 to 2N making it possible to design larger arrays and arrays with smaller element sizes. The system utilizes the electromechanical bi-stability of individual elements to hold arbitrary bi-stable phase patterns, a technique previously used on tilt arrays.

The control system applies pulse width modulated (PWM) signals to the rows and columns of the device to generate a static phase pattern across the array. Three modes of operation were considered and built into the system. The first was the traditional signal scheme which requires the array to be reset before a new pattern can be applied. The second is an original scheme that allows dynamic switching between bi-stable patterns. The third and final mode applies an effective voltage ramp across the device by operating above mechanical cutoff. Device characterization and control system testing were conducted on samples from two different foundry processes.

The test results showed that the control system was successfully integrated, however individual bi-stable control was not successfully demonstrated on the micromirror arrays tested. The inability to demonstrate bi-stable control is attributed to flaws in the device and variations in snap-down voltage with the application of PWM signals below mechanical cutoff. Methods to correct these flaws for a future redesigned lin-addressable device are proposed.

CONTROL AND CHARACTERIZATION OF LINE-ADDRESSABLE MICROMIRROR ARRAYS

I. Introduction

"...the future of MEMS is more than bright – it is dazzling."
- Kurt Peterson, Ph.D., MEMS pioneer, 1998 [28]

"I think there is a world market for maybe five computers."
- Thomas Watson (1874-1956), Chairman of IBM, 1943[38]

1.1 Problem Background

Microelectromechanical systems (MEMS) devices continue to offer exciting and inexpensive alternatives to various conventional sensor and actuation systems. In the field of optical engineering, the micromirror array is an example of one such MEMS device. When properly controlled, micromirror arrays can modulate light spatially in both static and dynamic settings. Their use has been theorized and, in some cases, already realized in various areas of applied optics including adaptive optics [13], optical switching [27], optical neural networks [8], optical projection [21], and others [25]. The commercial potential of these devices has already been demonstrated, most notably by the immense success of Texas Instruments' Digital Micromirror Device (DMD). Their potential for specific applications in weapon system design, optical networking, and satellite imaging has caught the attention of several military researchers. Consequently, the Defense Advanced Research Projects Agency (DARPA) has sponsored research into micromirror arrays and MEMS for the past several years. The Air Force Institute of Technology (AFIT) has led several pioneering efforts in the study of micromirror arrays from which this research is a direct consequence [4, 9, 13, 19, 32, 37, 40].

The size of the array, granularity of elements, and the adjustability in the position of individual elements are the main features that dictate performance and limitations of a micromirror array. Conventional array designs control the individual elements of the micromirror array by individual wiring of each element. However, this method limits the size of the array since the large number of physical interconnects required becomes impractical. Texas Instruments' DMD overcame this obstacle by monolithically fabricating its micromirror array over an SRAM (Static Random Access Memory) array that would digitally store the position of each element [21]. Each element's control electrode is then addressed by a memory cell directly underneath the element, eliminating the need to route wires underneath the array. However, the fabrication cost necessary for this level of integration is high, especially for applications with smaller expected markets. In addition, this control method limits the number of states a particular element can hold. The desire for increased capability and more cost effective methods has motivated research into implementing alternative control schemes utilizing line-addressing [14, 24]. Line-addressing implies controlling individual elements solely through wiring of the rows and columns of the array. For a square NxN array, line addressing will reduce the number of control lines needed from N^2, in a conventional design, to 2N. Thus line addressing allows increased array sizes while maintaining actuation of individual elements and the ability to fabricate using a low-cost, high-yield process.

Previous work on line-addressable control of micromirror arrays is primarily limited to two papers [14, 24]. The first, from Jaecklin et al. [24], describes a line-addressable torsional micromirror array device. Jaecklin proposed a control scheme for the device such that each element in the array could occupy one of two positions, a "bi-stable" mode of operation. This control scheme utilizes the electromechanical bi-stability inherent in the device to hold positioning of the elements. Essentially it allows the application of an arbitrary static bi-stable pattern to the array. However, this control scheme requires that the array must be reset, before a new pattern can be applied to the array. While Jaecklin proposed this control scheme, it was not implemented for this paper.

The second paper, from Cunningham et al. [14], describes another torsional micromirror array, dubbed a bi-stable array of micromirrors (BAMM). This design was not purely line-addressable since it actually had two control lines for every column and one line for every row. However, these extra lines allowed switching of each element between two different tilted positions, where in the previous work the two bi-stable positions were "tilt" and "no tilt". A control scheme based on the same principle utilized by Jaecklin was designed and implemented, proving that electromechanical bi-stability was indeed a viable means of control. In addition, the extra lines afford the ability to switch dynamically between bi-stable patterns.

This thesis is an extension of work first initiated by Lt Col William Cowan and Capt David Conrad at the Air Force Research Laboratory Hardened Materials Branch (AFRL/MLPJ) at Wright-Patterson AFB, OH [10, 11]. It expands the idea of line-addressable control using electromechanical bi-stability to a piston-actuated micromirror array, and improves upon the previously mentioned control schemes. The line-addressable micromirror array (LAMA) used in this thesis was originally designed by Lt Col William Cowan. The long-term goal behind this work is to successfully develop a line-addressable device and corresponding control scheme such that each mirror element could hold a variety of positions (multi-stable). This would significantly increase the versatility in the aforementioned optical applications.

1.2 Problem Statement

This thesis is a proof-of-concept study with an aim of designing and testing a line-addressable control circuit for a bi-stable micromirror phased array. I explore the implementation and capabilities of a specific line-addressed micromirror system, unlike any other previously attempted. Specifically, a control system was needed for a line-addressable phased micromirror array previously designed at AFIT.

1.3 Scope and Objectives

My objectives for this thesis were 1) to implement and design the remaining components for the intended control system, and 2) demonstrate the system's ability to hold a static bi-stable deflection pattern when connected to a line-addressable micromirror array. The intent was to prove that line-addressable operation of a micromirror phased array could be achieved using pulse-width modulated signals and that this type of system has the potential for use in conventional applications. In addition, alternate modes of operation were considered throughout the design process. The scope was initially limited to demonstrating variable dynamic bi-stable control of a far-field pattern with the micromirror array system.

1.4 Approach and Methodology

My work centered on the achievement of six central milestones. The first milestone was to develop a complete theoretical and experimental understanding of the micromirror arrays to be controlled. This understanding included their physical structure, method of fabrication, and performance characteristics. Testing of individual rows and columns on the array were accomplished with both DC and amplified pulse width modulated (PWM) signals. Effective spring constants were experimentally estimated for use in both steady-state and frequency response models.

The second milestone, was to develop an understanding of the control system intended for these micromirror arrays. This included identifying the necessary components for the control system and ascertaining what needed to be done to create each of them. It was determined that the control system should consist of three central parts: 1) the software control interface; 2) the personal computer (PC) controller board; and 3) the pulse amplification circuit board. After several lengthy discussions with the device designer, a survey of the existing documentation, and some reverse engineering, a full understanding of each of their roles was obtained.

The PC controller board was previously designed and fabricated for me before the start of this project. I needed to design and fabricate the other two components myself. Unfortuantely, little documentation existed on the operation of the PC board and as a result much of its architecture had to be mapped by hand. Ensuring its functionality and understanding its wiring and associated circuitry was the third milestone. Specifically, jumpers had to be correctly inserted into the circuit before it would perform its intended function. Once this was accomplished, testing of the board using previously existing software and appropriate electrical testing equipment, revealed that the board was fully functional.

The design and fabrication of the pulse amplification circuit board was the fourth milestone. This began by first designing a simple transistor amplification circuit that could provide the necessary frequency and gain characteristics for a single channel and could easily be repeated for all 64 channels without exceeding power limitations. The circuit also needed to be robust enough to handle various modes of testing. Once such a circuit was determined and the exact components were chosen, a schematic of the entire board was generated by hand using a computer-aided design program. The actual layout of the components on the board was accomplished by hand through the assistance of design software. After spending several hours optimizing the routing of lines between components the final design files were sent to a foundry and a physical board was received two weeks later.

The creation of a software control interface was the fifth milestone. This involved creating a Graphical User Interface (GUI) environment that could allow the user to easily create, load, and store any bi-stable pattern desired and subsequently apply it to the micromirror array at the click of a button.

The sixth and final milestone was assembling all the components together and demonstrating the ability to hold a desired static deflection pattern across the micromirror array. This first involved soldering all the components to the amplification board and insuring its functionality. Several errors were discovered in the final board design, but modifications were made such that the intended design was realized. Next several line-addressable micromirror array samples were prepared for testing in the fabrication lab. Finally all the system components, and associated test equipment were calibrated and assembled together. Once the control system was found fully functional, various sample test devices were connected to it, and system operation in the various modes was tested. This was done by viewing the test devices under a microscopic laser interferometer. Video clips were collected during testing to document the results.

1.5 Thesis Overview

This thesis is organized into seven chapters and eight supporting appendices. The first chapter introduces the reader to the necessity for line-addressing in micromirror arrays. The second chapter provides a review of supporting theory and pertinent background information. The third chapter discusses the physical design and how characteristics of the line-addressable micromirror array (LAMA) device are measured and used in device modeling. The fourth chapter details the theory used to operate these devices and how this theory was implemented into an actual control system used for the devices. The fifth decribes the experimental setup used during characterization and system testing. The sixth chapter presents the results of this testing and corresponding analysis. The seventh and final chapter highlights the conclusions made during this research and provides recommendations for how future work should be directed.

II. Theoretical Review

"Education is a progressive discovery of our own ignorance."

-Will Durant [38]

2.1 Overview

As discussed in Chapter 1, considerable research has been focused on the design and control of micromirror arrays. This chapter describes some of the existing designs of micromirror arrays and their control schemes. In addition, it highlights several of the rudimentary theoretical principles used in achieving the objectives mentioned in Section 1.3 for the LAMA system that is the subject of this thesis.

Section 2.2 provides an introduction to micromirror arrays and explains why they are classified as spatial light modulators. Section 2.3 highlights the three types of electrostatic micromirror designs previously used in arrays. Section 2.4 describes the two main types of spatial light modulation and which mirror design is best suited for each. These sections are intended to explain why the LAMA design of interest is classified as a spatial light modulator.

Section 2.5 describes the concept of pulse width modulation. Pulse width modulated (PWM) signals are used in the control system for this study, which is described later in Chapter 4.

Section 2.5 reviews the two surface micromachining prototyping technologies used to create the line-addressable micromirror arrays used in this study. This section provides a solid understanding in the material structure of the devices, and the inherent limitations of the processes.

2.2 Micromirror Arrays

Micromirror arrays were some of the first MEMS devices ever conceived. Their development dates back to the mid 1970's, particularly by Peterson [35] (a pioneer of MEMS), and Hornbeck (the inventor of Texas Instruments' DMD) [20, 21]. Simply defined, a micromirror is a very small movable mirror that has been manufactured through a micromachining process, typically on silicon. Micromirrors can range in size and shape, functioning as a group or individually, based on their intended function. When micromirrors are placed together to form an array, they can collectively act as a deformable mirror spatial light modulator (SLM).

Deformable mirror SLM's can be classified as either continuous sheet or segmented. Continuous sheet SLMs consist of some form of membrane or elastomer deposited over an underlying array of actuators. Applications that require very low optical loss, such as adaptive optics, generally use this type of SLM [11]. However continuous phase modulators are typically expensive to construct and control. Segmented deformable mirror SLM's consist of several smaller closely spaced mirror elements that function together. Micromirror arrays are the most commonly discussed segmented deformable SLM's; however, large-scale segmented SLM's do exist [23]. Although micromirror arrays may not provide as high a quality optical signal, due to the inherent gaps between elements, they do have several advantages over their continuous sheet counterparts. Most notably they typically have lower power dissipation, and can be monolithically fabricated on silicon, making them comparatively inexpensive to construct [20].

Micromirrors can be controlled through various means of actuation. In the computer age, electronic circuitry provides a simple platform to design a control system for a micromirror array. Thus the vast majority of micromirrors used in arrays are designed to utilize voltage inputs. Considerable research had been accomplished at AFIT by Bright and others with mirrors that utilized electrically driven thermal actuators [4, 9]. However, these do not have adequate response times to be used in an array intended for dynamic spatial light modulation. Electrostatic actuation provides simple coupling to controls and fast response times, making it the mechanism of choice.

2.3 Electrostatically Actuated Micromiror Designs

The following subsections highlight three common types of electrostatic micromirrors used in array designs. Note that numerous designs have been previously demonstrated for all three [13, 20, 27].

2.3.1 Piston Actuated

This type of design consists of two parallel plates, typically square, wired to two separate electrodes. The surface of the upper plate acts as the mirror surface and is typically coated with a metal or other reflective material. This upper plate is suspended above the lower plate by flexure beams that are connected to the ground plane and effectively act as springs. The fabrication process used to create them dictates the separation distance. When a voltage difference is created between the two plates, the resulting electrostatic force pulls the upper electrode down towards the bottom plate. The flexures provide a restoring force which balances the electrostatically induced force. This balance allows a stable piston-like motion to be achieved perpendicular to the surface in which the mirror lies. Figure 2-1 graphically depicts this type of design.

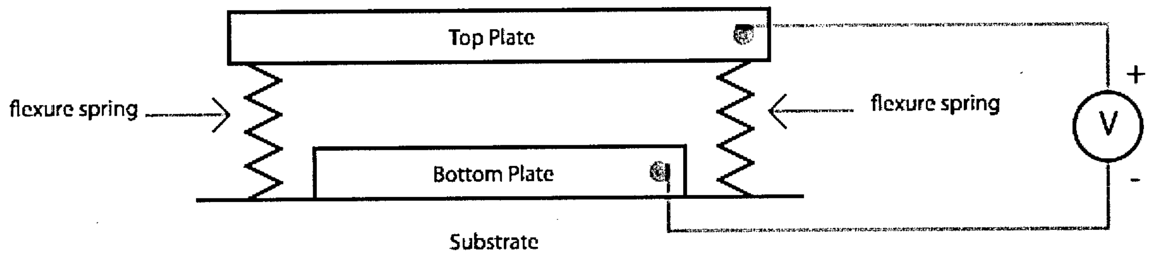


Figure 2-1. 2-D representation of a piston actuated micromirror

2.3.2 Torsional Beam

This type of design causes a tilting or tipping motion of a mirror plate. Once again there are two sets of plates connected to separate electrodes. In most designs of this type, an upper plate is supported by a beam or beams upon which it can pivot [21, 24, 27]. The lower electrode can be on one or both sides of the beam allowing the mirror to tilt in either direction when a voltage difference is applied across electrodes. The restoring force lies in the beam, which acts as a torsional spring. A cross-sectional representation of this type of design is shown in Figure 2-2. The most prominent example of a torsional micromirror device is the Texas Instruments DMD [20, 21].

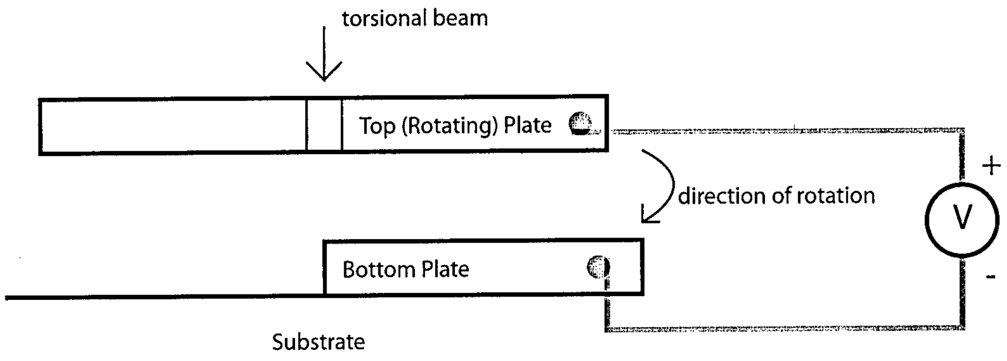


Figure 2-2. 2-D representation of a torsional beam micromirror

2.3.3 Cantilever Beam

The simplest type of micromirror device is the cantilever design. The cantilever consists of an upper electrode suspended above a lower electrode by a beam that is fixed at one end. As in the previous designs, when a voltage is applied between the electrodes the beam is bent down. Once again the structural elasticity of the beam is relied upon to provide the necessary restoring force. Prototype designs of Texas Instrument's DMD predecessor, the Deformable Mirror Device (also abbreviated DMD), were of this type [21]. Figure 2-3 shows a representation of this design.

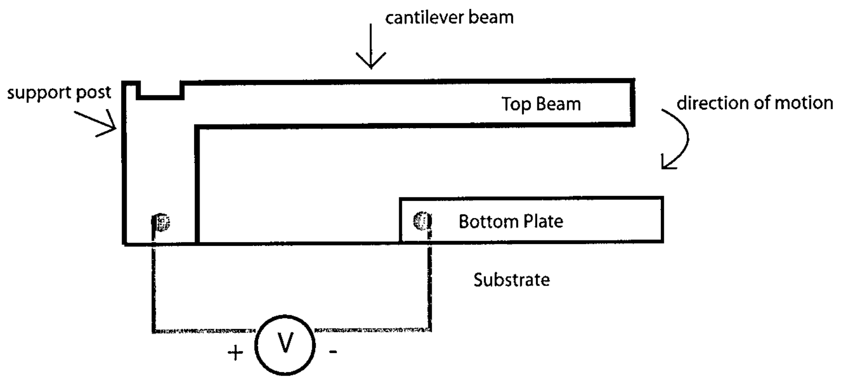


Figure 2-3. 2-D representation of cantilever beam micromirror

2.4 Spatial Light Modulation

As previously stated, micromirror arrays can accomplish spatial light modulation. Spatial Light Modulation is a term that implies manipulating the spatial extent of light, typically in the far-field. There are, however, two distinctly different types of modulation that can be achieved; intensity and phase modulation. The type of modulation desired determines the type of micromirror element design that should be implemented in an array.

Intensity modulation implies controlling the radiation distribution that is incident over a certain area. Portions of a wavefront incident on an intensity modulator are either reflected toward or away from a designated area. The speed at which this switching occurs determines the average intensity of light that is incident on the area. Micromirror arrays utilizing intensity modulation can be used in pixilated imaging systems. In this application each element of the array controls that amount of light incident on a corresponding pixel. Torsional and cantilever beam elements are used for this type of modulation since they can switch reflected light on and off a target by pivoting [19].

Phase modulation implies controlling the amount of phase accumulated across an incident wavefront upon being reflected off of the surface of the modulator. For an incident wave with a phase of $\varphi(\mathbf{x},t)$, the corresponding reflected field off of the phase modulator device can be expressed as:

$$E_r(\mathbf{x},t) = \exp[i(\Phi(\mathbf{x},t) - \Delta(\mathbf{x},t))] \qquad \{2\text{-}1\}$$

where **x** is the spatial coordinate, t is the time, and $\Delta(\mathbf{x}, t)$ is the accumulated phase of the wavefront [19]. When these phase modulation elements are placed in an array the amount of phase accumulation across the array can be varied to change the overall propagation of the reflected wave. Piston actuated elements are used to achieve this type of modulation since the vertical position can control the amount of phase accumulated for that particular element. These types of arrays are used in beam steering and phase correction applications. The micromirror array that is the focus of this research is piston actuated and is thus designed to perform phase modulation.

2.5 Pulse Width Modulation

The line-addressable micromirror array (LAMA) system used in this research utilizes Pulse Width Modulated (PWM) control signals to actuate individual micromirror elements. A PWM signal is a periodic rectangular signal (typically voltage) of fixed frequency and varying duty cycle. Figure 2-4 shows a generalized ideal PWM waveform.

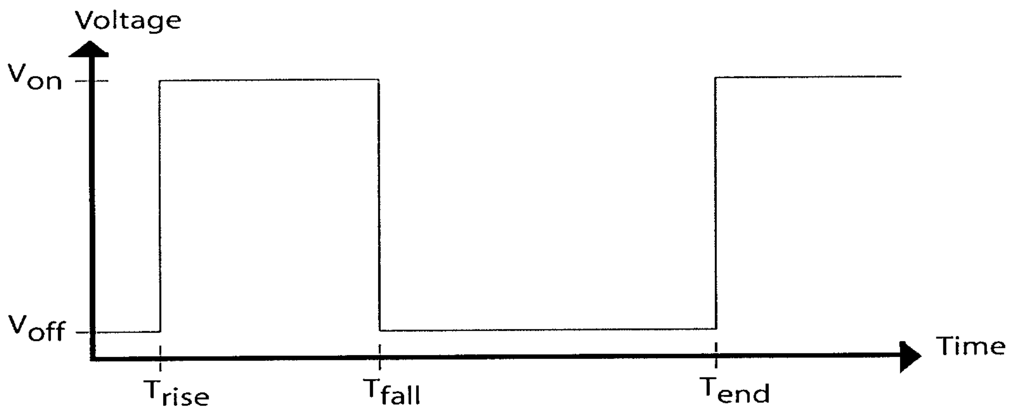


Figure 2-4. Example Pulse Width Modulated Waveform

Duty cycle refers to the percentage of time that the signal is in the active state. The duty cycle of the PWM waveform depicted in Figure 2-4 is defined as [1, 40]:

$$DutyCycle = \frac{T_{fall} - T_{rise}}{T_{end} - T_{rise}} \qquad \{2\text{-}2\}$$

The primary advantage of using pulse width modulation over other control techniques is that the signal can be created and manipulated entirely through digital circuitry, meaning conversion from an analog signal is unnecessary. However, it should be noted, that in communication systems where conversion from an analog signal is required, pulse width modulation remains a viable option [39]. In addition, most modern digital circuitry is fabricated using complementary metal oxide semiconductor (CMOS) logic, which has low power consumption as well as several other benefits. This logic allows pulse width modulation to be used as an extremely efficient means of voltage regulation, which is why it is typically used in power supplies or drive circuitry for motors [1, 40]. When using digital logic, the active state of the signal corresponds to a logic state of 1. For a CMOS digital circuit design, a logic state of 1 usually is 5V, while a logic state of zero is 0V [1].

Although the PWM signals used in the LAMA system are not intended for voltage regulation, power consumption plays an important role in the design of pulse amplification circuitry. It therefore becomes important to understand how energy is transferred in a PWM signal.

When the instantaneous power $p(t)$ drawn by a given load varies periodically, as it does with a PWM signal, the long term average power equals the average over one or more periods. Thus the average power dissipation , P, or rate of energy transferred to a load, is defined as [5]:

$$P = \frac{W}{T} = \frac{1}{T}\int_{t_1}^{t_1+T} p(t)dt \qquad \{2\text{-}3\}$$

where W is the total energy transferred to the load, T is the period of the signal, and t_1 is some arbitrary time. When considered in terms of a time-varying voltage, $v(t)$, applied to a resistance, R, the instantaneous power is equivalent to:

$$p(t) = \frac{v^2(t)}{R} = Ri^2(t) \qquad \{2\text{-}4\}$$

where $i(t)$ is merely the corresponding time-varying current. Thus Equation {2-3} can be expressed as:

$$P = \frac{\left[\frac{1}{T}\int_{t_1}^{t_1+T} v^2(t)dt\right]}{R} \qquad \{2\text{-}5\}$$

However, it is typically easier to think of the power dissipation in terms of a constant voltage (or current), which is why the effective or root-mean-square (rms) value of a periodic signal is commonly used. In the case of a periodic voltage the rms value, V_{rms}, is defined by [5]

$$V_{rms} = \sqrt{\frac{1}{T}\int_{t_1}^{t_1+T} v^2(t)dt}. \qquad \{2\text{-}6\}$$

Thus, an alternate expression for average power is

$$P = \frac{{V_{rms}}^2}{R} \tag{2-7}$$

For the PWM signal defined in Figure 2-2 the rms voltage is given by

$$V_{rms} = \left(\frac{1}{T_{end} - T_{rise}} \cdot \left[\int_{T_{rise}}^{T_{fall}} {V_{on}}^2 dt + \int_{T_{fall}}^{T_{end}} {V_{off}}^2 dt \right] \right)^{1/2} \tag{2-8}$$

which simplifies to

$$V_{rms} = \left(\frac{T_{fall} - T_{rise}}{T_{end} - T_{rise}} \cdot \left(V_{on} - V_{off}\right)^2 + \left(V_{off}\right)^2 \right)^{1/2} \tag{2-9}$$

Note that the initial ratio in Equation {2-9} is the duty cycle. Substituting Equations {2-2} and {2-9} into Equation {2-7} yields

$$P = \left(DutyCycle \cdot \frac{(V_{on} - V_{off})^2}{R} \right) + \frac{(V_{off})^2}{R}, \tag{2-10}$$

which is an expression for the average power dissipation of a PWM signal through a resistance R. Equation {2-10} shows that this average power can be varied linearly by varying the duty cycle.

Pulse width modulated control has been previously used to actuate micromirror arrays whose elements were individually wired. This was first accomplished at AFIT on thermally actuated mirrors by Butler, and later on electrostatically actuated arrays by Rounsavall [15]. In the case of Rounsavall's electrostatically actuated array, the PWM signals were applied at a frequency above the mechanical cutoff of the devices. The micromirror array elements in the system responded to the rms voltage of the signal, which the control system varied through the duty cycle as desired. Previously developed line-addressable control systems [14, 24], as well as the LAMA system that is the subject of this study, utilize PWM signals at frequencies below mechanical cutoff. This mode of operation allows the actual magnitude of the signal at a given time to be applied to the mirror elements.

2.6 Silicon MEMS Prototyping Fabrication Techniques

Microelectromechanical systems can be classified in various ways such as by function (sensors or actuators), method of transduction [28], degree of complexity, or method of fabrication. While there currently is no taxonomy standard for MEMS, it is necessary to understand how these device are created in order to properly use them. Such is the case with the LAMA device used for this study. A great deal of research continues to be devoted to this subject, as it is truly the essence of the field. This section provides only a cursory review of the subject, briefly highlighting the techniques used to fabricate the LAMA device used in this thesis. A much more extensive review of the subject is presented by Madou [31] and Kovacs [28].

In general, when microfabrication, or micromachining, is being discussed it is often in reference to making devices using a silicon substrate, however other materials such as GaAs are used, particularly for integration with light emitting devices, and are being actively researched. This is primarily due to the fact that many of the techniques used in micromachining leverage technology already used for integrated circuit fabrication, which is primarily based on silicon and very well developed. Thus when MEMS devices are manufactured on silicon, they are processed in mass on a wafer, in the same fashion as conventional microelectronics chips. Micromachining is typically classified into two branches. The first, involves etching into the substrate to create structures, and is called *bulk micromachining*. The second, deposits additional layers of material to the substrate and patterns them accordingly to create structures. This method is called *surface micromachining* and is used to create the majority of MEMS devices, since the possibilities afforded by several layers are enormous.

The two primary commercially available foundries for prototyping MEMS devices both employ surface micromachining. They are Sandia's Ultra-Planar Multi-level MEMS Technology (SUMMiT), from Sandia National Laboratory, and Cronos Integrated Microsystem's Multi-User MEMS Process (MUMPs™). Sample LAMA devices fabricated from each technology were used in this thesis, and as a result certain key differences exist between them. However, a basic understanding of surface micromachining is necessary to appreciate them.

In surface micromachining, the mechanical layers used to create the device structure are typically some form of polysilicon. Devices that require a certain degree of freedom to move, are typically suspended above the substrate, as is the case with micromirrors. This is accomplished through the use of sacrificial oxide layers, which are used to separate the polysilicon layers. These layers are later etched away after the device is fabricated, in what is commonly termed the "release" process, allowing the device structure to be suspended as desired. Figure 2-5 illustrates this process through the construction of a simple cantilever beam.

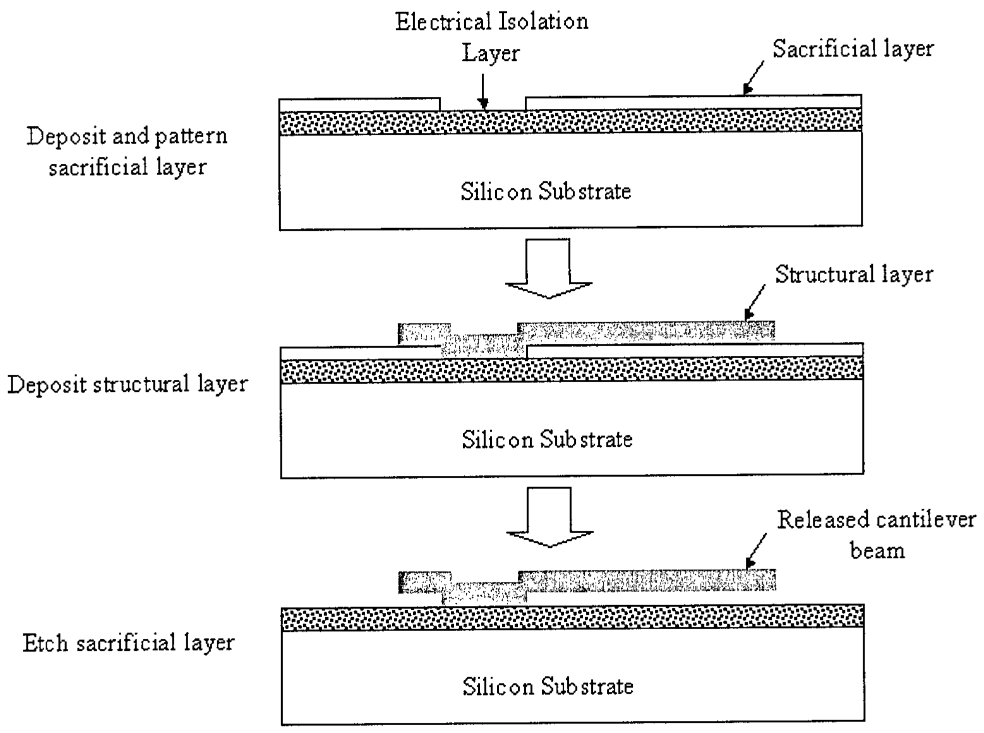


Figure 2-5. Simplified depiction of surface micromachining process, from [12]

Patterning of each layer in the surface micromachining process is typically accomplished through photolithography. First, the material layer is deposited conformally across the entire wafer. The conformal nature of this deposition denotes that the surface topology of the layer being deposited "conforms" to that of the layer beneath it. Next, a thin layer of photoresist is spun onto the wafer. Photoresist is a polymer based syrup-like compound that is sensitive to light. Depending on the type of photoresist, when exposed to light it either strengthens or breaks apart the molecular bonds between polymer chains. Patterns are thus applied to the photoresist by exposing it under a mask that has the desired pattern for that layer. The sample is then dipped in a chemical bath that rinses away the areas where the polymer bonds are weakened. The sample is then etched such that the areas in the layer that aren't covered with photoresist are removed. The photoresist acts effectively as a pattern mask for the etch. Once the etching is completed, the photoresist is stripped and the process is repeated for the next layer. Photolithography is used extensively in integrated circuit (IC) fabrication. As a result, many of the deposition and etching techniques used in IC fabrication can also be used to fabricate MEMS devices. It is important to note that it is the photolithography process that determines the minimum feature size and line spacing. Subsequently, the IC industry is continuously driving research to improve the process. A much more extensive review of photolithography and how it is accomplished are presented by Sze [43].

Process factors such as thermal annealing, deposition, etch method, surface planarization, number of layers, layer thicknesses, and doping of polysilicon are what distinguish a given surface micromachining process. To increase yield and ease the release process additional design factors are also included. For instance, to prevent stiction between suspended layers during device release and operation, dimples are available for the bottom surfaces of the polysilicon structural layers. Another common feature is the inclusion of etch release holes to speed the release of the device. The SUMMiT and MUMPs™ processes differ in several of these aspects.

MUMPs™ utilizes 3 layers of polysilicon (2 releasable) and a gold metal layer. It costs roughly $3,000 for a single 1cm square die site, making it a relatively low cost process. For each die the user receives approximately 15-16 samples. In addition, it has a very fast turnaround time of about 2 months. An additional feature unique to the MUMPs™ process is that a significant amount of residual stress is often exhibited by the releasable polysilicon layers. One final benefit is that the sacrificial oxide used in MUMPs™, a type of phosphosilicate glass (PSG), etches very fast (~2.5 min) in an hydrofluoric acid (HF) based solution.

The SUMMiT process is a four layer polysilicon process (3 releasable). However, no metal layer is available making it difficult to bond wire to the device when packaging. The primary advantage of the SUMMiT process is that it incorporates a chemical mechanical polishing (CMP) step into the creation of the final sacrificial oxide layer. This results in an extremely smooth surface suitable for optical applications. It also utilizes a more advanced photolithography process, compared to the MUMPs™ process, allowing smaller minimum feature sizes for each layer. In addition, the polysilicon layers are nearly stress free. As can be expected, these more advanced features make the process more expensive, roughly $10,000 for a 0.5 cm square die. It also requires a much longer time to etch the sacrificial oxide layers (anywhere from 15-45 min) in a similar HF solution.

Figures 2-6 and 2-7 graphically outline the various layers used in the MUMPs™ and SUMMiT processes respectively. Further details on the processes and their corresponding design rules can be found in design rule handbooks [12, 26].

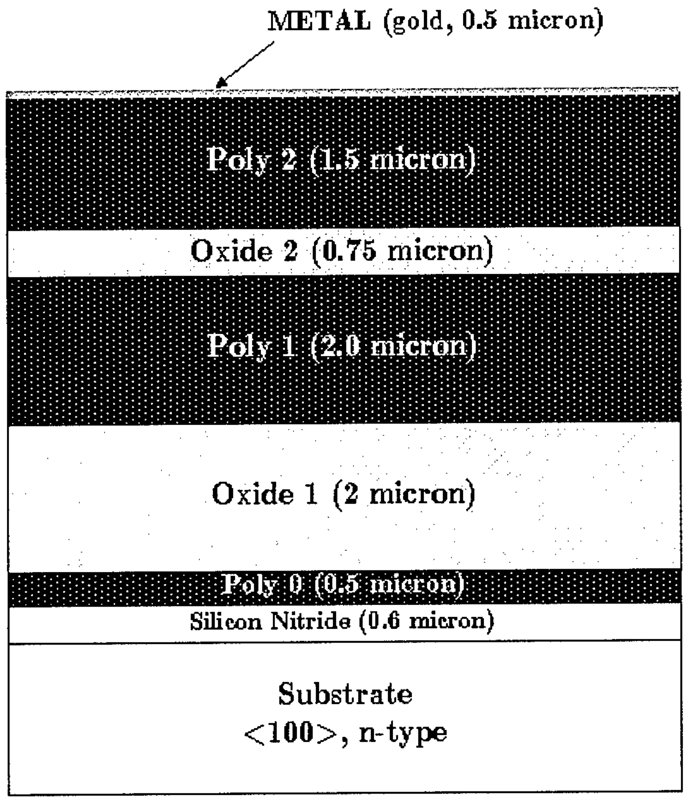


Figure 2-6. Outline of deposition layers for MUMPs ™ process

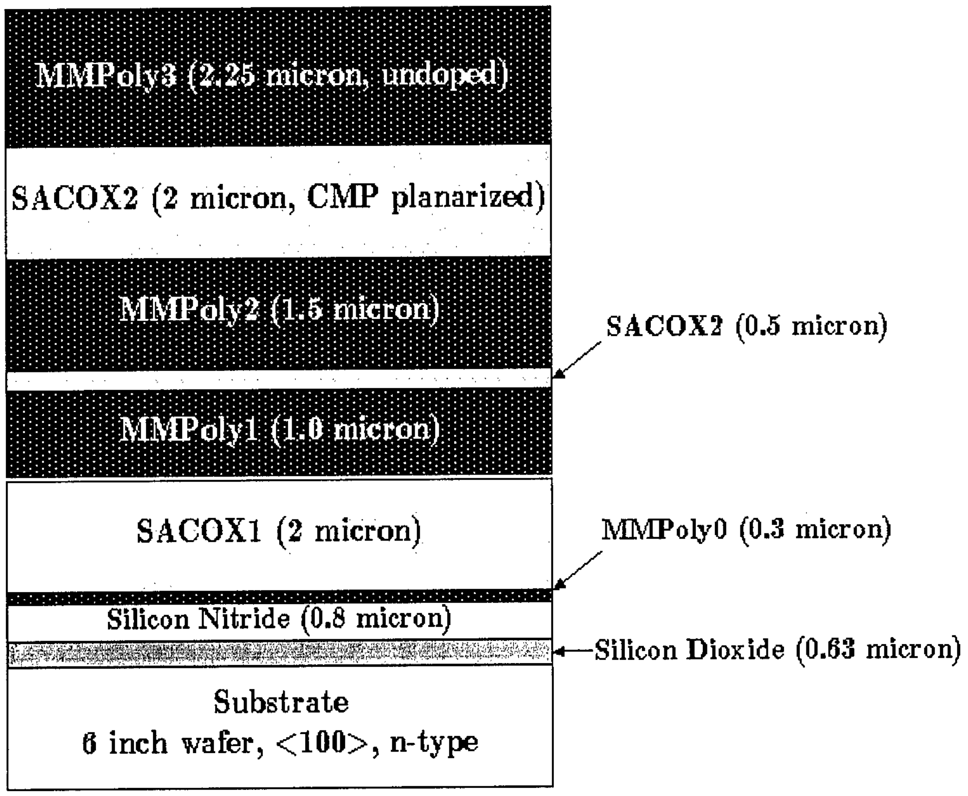

MMPoly3 (2.25 micron, undoped)
SACOX2 (2 micron, CMP planarized)
MMPoly2 (1.5 micron)
SACOX2 (0.5 micron)
MMPoly1 (1.0 micron)
SACOX1 (2 micron)
MMPoly0 (0.3 micron)
Silicon Nitride (0.8 micron)
Silicon Dioxide (0.63 micron)
Substrate
6 inch wafer, <100>, n-type

Figure 2-7. Outline of deposition layers for SUMMiT process

III. Characterization of the Line-Addressable Micromirror Array

"In theory, there is no difference between theory and practice. But, in practice, there is."
- Jan L.A. van de Snepscheut [38]

3.1 Overview

The first step in designing a control system for the LAMA device is understanding its relevant properties and characteristics. The most obvious of these is its physical structure which is described in Section 3.2. It is particularly necessary to explore the electromechanical response of the individual micromirror elements. Section 3.3 describes the models used in this work to accomplish this analysis. Section 3.4 explores the device's functionality as a bi-stable phase modulator by describing a model of its optical response in the far-field. Finally, Section 3.5 describes an experimental method, known as the static fringe method, for estimating the deflection response of an individual element under DC conditions.

3.2 Micromirror Array Design

The line-addressable micromirror arrays used in this study are electrostatically piston actuated, each consisting of 32x32 elements with structure similar to that described in Section 2.3 (see Appendix E for actual design layout). However, as explained in Section 2.6, the actual cross sections for MUMPs™ and SUMMiT devices are significantly different. These are shown in Figure 3.1 and 3.2 respectively.

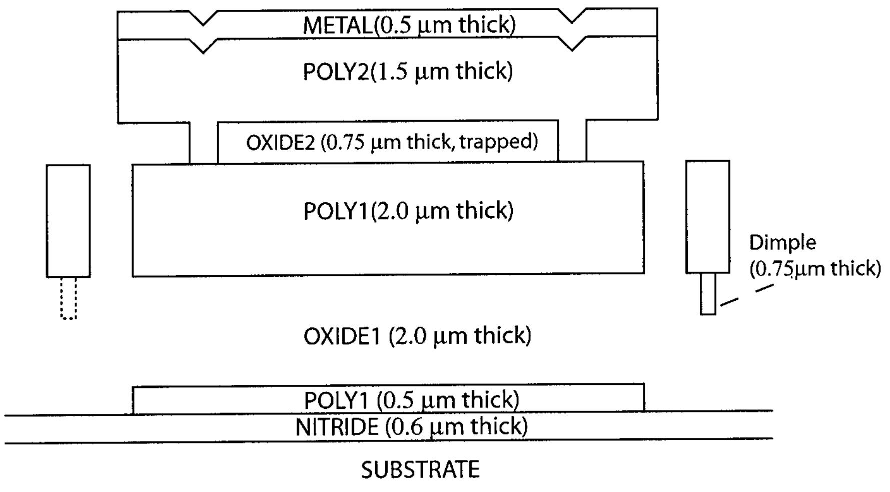


Figure 3-1. Cross Section of MUMPs™ LAMA element

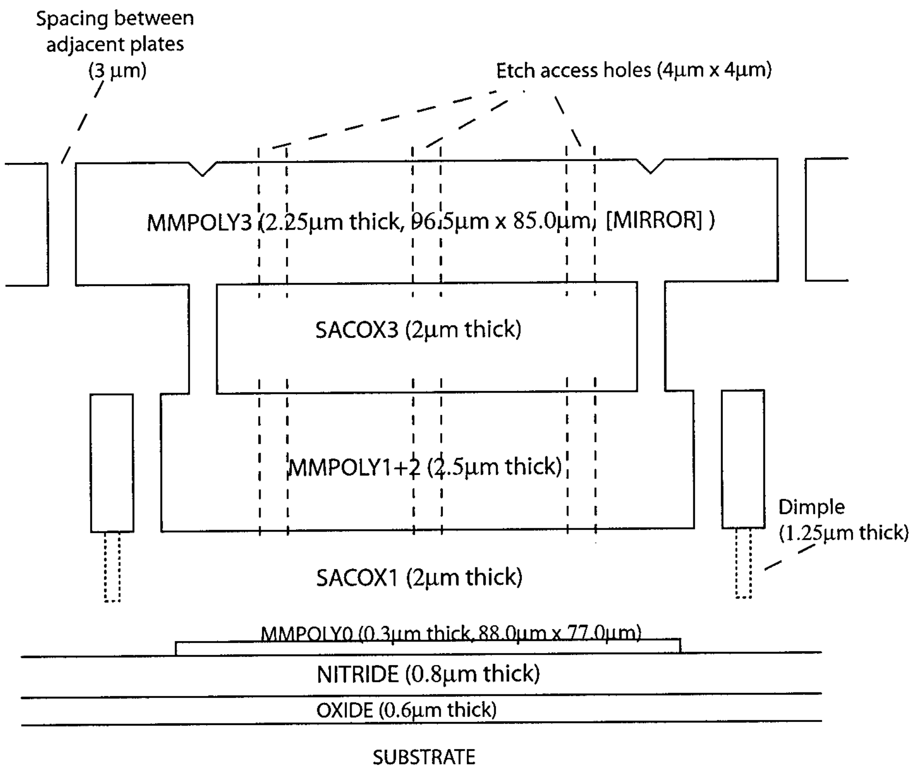

Spacing between adjacent plates (3 μm)
Etch access holes (4μm x 4μm)
MMPOLY3 (2.25μm thick, 96.5μm x 85.0μm, [MIRROR])
SACOX3 (2μm thick)
MMPOLY1+2 (2.5μm thick)
Dimple (1.25μm thick)
SACOX1 (2μm thick)
MMPOLY0 (0.3μm thick, 88.0μm x 77.0μm)
NITRIDE (0.8μm thick)
OXIDE (0.6μm thick)
SUBSTRATE

Figure 3-2. Cross Section of SUMMiT LAMA element

The dimples shown in Figures 3.1 and 3.2 are added to prevent stiction of the flexures to the substrate during release. They also help prevent the upper and lower plates of the micromirror element from shorting during operation. However, one major flaw in the LAMA design, is that these dimples are not spaced evenly at all four corners of the element. They are spaced only at two opposing corners. Thus if an element is snapped down towards the substrate such that the dimples touch, the element can (and does) actually tilt in one of two diagonal directions. Thus these prototype LAMA chips would not make very good phase modulators in a practical setting. However they do suffice for experimental use in demonstrating line-addressing. Also note the etch access holes in Figure 3.2. These are not necessary in the MUMPs™ design because the fill factor is much smaller and the sacrificial oxide is more easily removed.

While these cross sections are indeed important in predicting the device's behavior, what truly makes the LAMA unique compared to other micromirror arrays is how the elements are connected together. All the bottom plates are connected in one direction by "columns" and all the upper plates are connected together by "rows" using the flexure beams as interconnects, making the design line-addressable. Figure 3-3 shows a schematic of this arrangement.

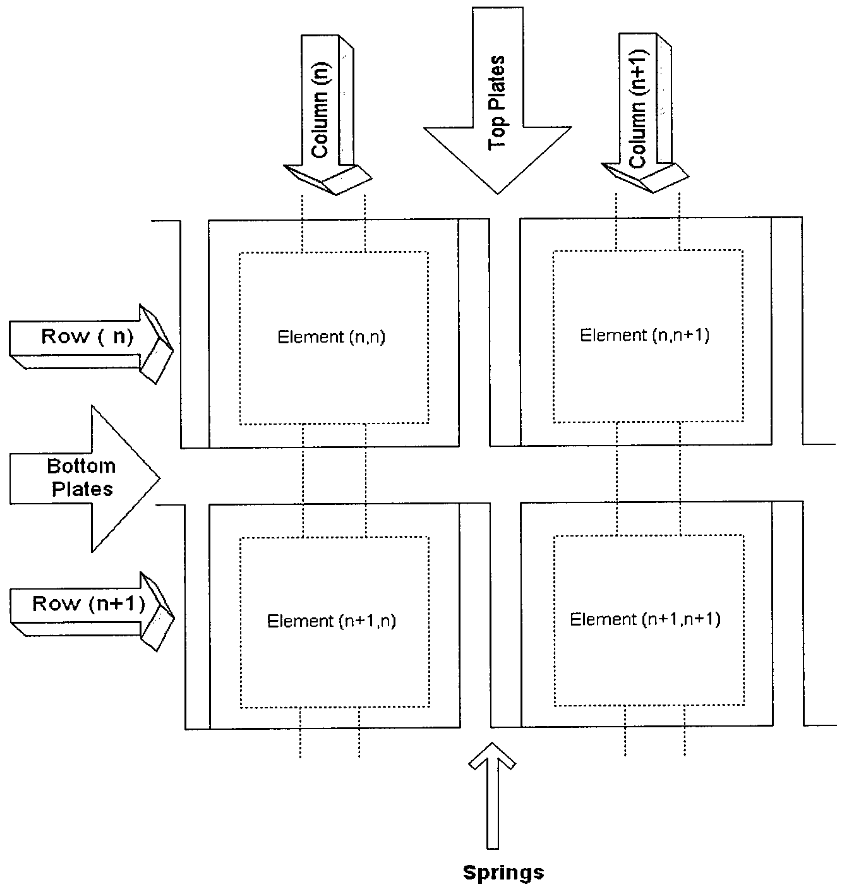


Figure 3-3. Schematic of LAMA element connections

A better grasp of the aforementioned structure can be gained by viewing the LAMA devices themselves using a Scanning Electron Microscope (SEM). Figures 3-4 through 3-6 show overhead SEM micrographs of the MUMPs™ LAMA device. Figure 3-7 shows a SUMMiT LAMA device for visual comparison.

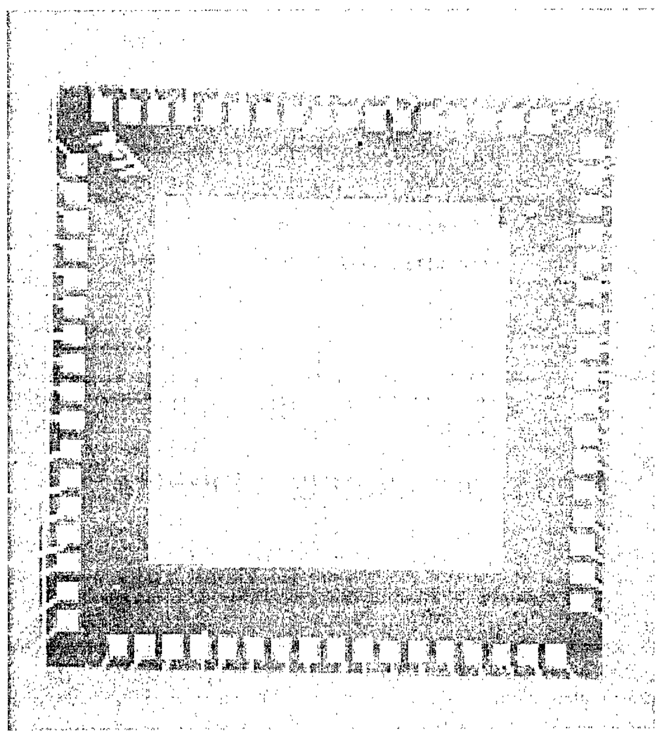


Figure 3-4. Overhead SEM photograph of MUMPs™ LAMA device

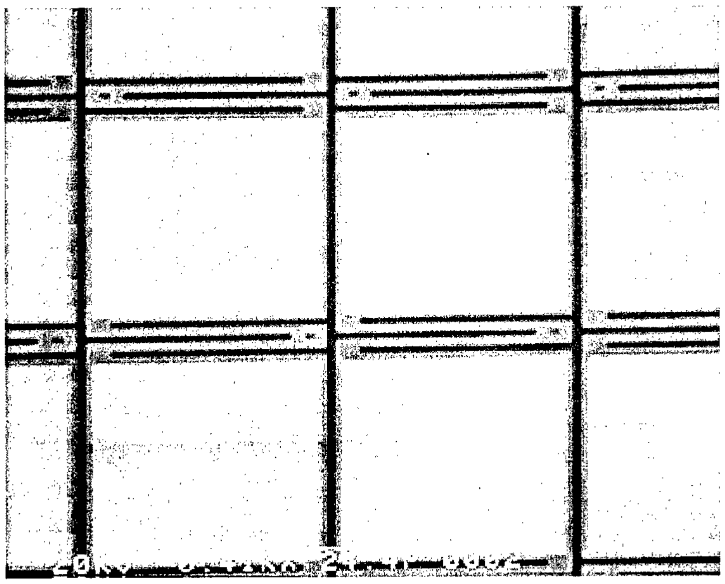


Figure 3-5. SEM photo of MUMPs™ LAMA elements
Photo orientation shows rows connected in the vertical direction.

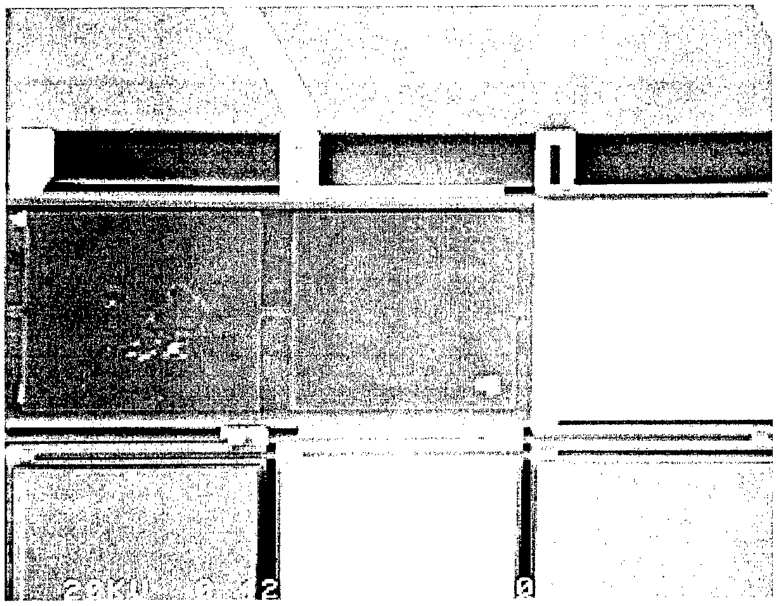


Figure 3-6. SEM photo of MUMPs ™ LAMA elements with top mirror plates removed
Photo orientation shows rows connected in the vertical direction, with POLY 0 columns connected across horizontally underneath. Note rows are connected from opposite sides of the package in an alternating fashion.

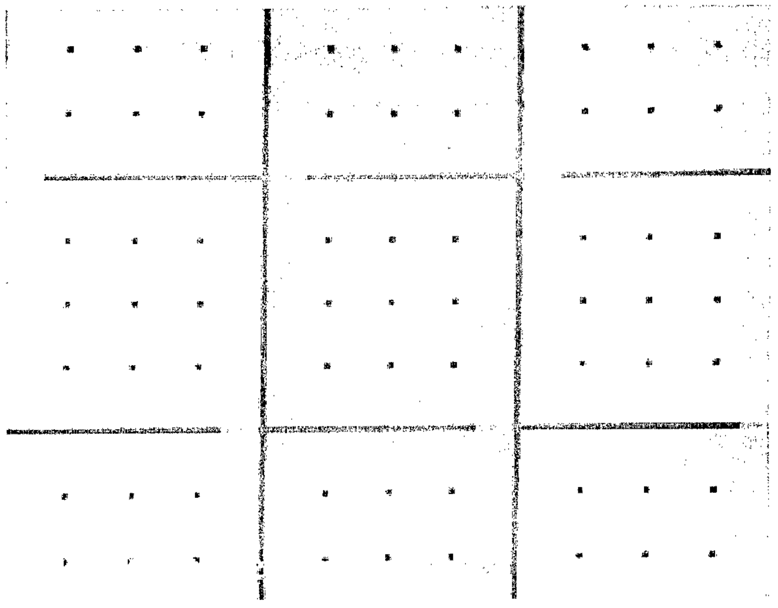


Figure 3-7. SEM photo of SUMMiT LAMA elements
Note existence of hole and high fill factor compared to MUMPs ™ device.

3.3 Electromechanical Model

Understanding how the individual elements will respond to an applied voltage is essential in successfully implementing any form of control system. Thus a great deal of research has been accomplished in the development of models which accurately describe the mechanical response of electrostatically driven MEMS devices, especially in the area of micromirror arrays [17, 22, 30]. Section 3.3.1 describes static DC voltage response of an electrostatically actuated micromirror element initially proposed by Lin [29]. Section 3.3.2 describes the electromechanical bi-stability property that is essential in realizing any form of line-addressable bi-stable control, and continues to be an active area of mathematical modeling research [34]. Section 3.3.3 provides a simplified frequency analysis for a single electrostatic micromirror element, in an attempt to provide an introductory understanding into the dynamic aspects of the system.

3.3.1 Static DC Voltage Response Model

We begin our analysis by first examining the micromirror element as a parallel plate capacitor. The definition of capacitance for this structure, neglecting any electric field fringing effects at the edges of the plates, is given by

$$C = \frac{q}{V} = \frac{\varepsilon_r \varepsilon_o A}{d_{gap}}, \qquad \{3\text{-}1\}$$

where q is the stored charge, V is the instantaneous voltage across the capacitor, A is the overlapping area of the two plates, d_{gap} is the separation of the plates at equilibrium, and ε_r is the dielectric constant or relative permittivity of the material between the plates compared to the permittivity of free space, ε_o. By noting that the displacement current thru the capacitor is, $i = C\frac{\partial V}{\partial t}$, the instantaneous stored energy, W, contained within the capacitor's electric field can be found by integrating the instantaneous power supplied, $P = Vi$, yielding the following relation:

$$W = \int Pdt = \int CVdV = \frac{1}{2}CV^2 + X_{\text{int}} = \frac{1}{2}CV^2 = \frac{1}{2}\frac{q^2}{c} \qquad \{3\text{-}2\}$$

where the constant of integration, $X_{\text{int}} = 0$, since there is no electric field to store energy when $V = 0$ [5]. The instantaneous force, F, in a given direction on the top plate can then be found by taking the derivative of the stored energy with respect to the coordinate direction in which the energy was stored. This is done in the following expression [30]:

$$F = \frac{\partial W}{\partial d_{gap}} = \frac{1}{2}\frac{\partial CV^2}{\partial d_{gap}} = -\frac{1}{2}\frac{\varepsilon_r \varepsilon_o A V^2}{d_{gap}^2}. \qquad \{3\text{-}3\}$$

The negative sign in Equation {3-3} implies that the force is directed downwards since I initially defined the plate separation distance as a positive value above the bottom plate. For an equilibrium deflection to be maintained this downward force must somehow be balanced be another opposing upward force. The mechanical flexure beams provide this upward balancing force, F_s, and can be modeled collectively as a single spring with a spring constant k, such that

$$F_s = k\Delta x = k(d_{gap} - x) \qquad \{3\text{-}4\}$$

where Δx is deflection distance of the top plate and x is its resulting height above the bottom plate. By realizing that d_{gap} in Equation {3-3} should now be replaced by $d_{gap} - \Delta x$ with the introduction of a spring, an expression for Δx as a function of instantaneous voltage, V, can be generated by setting this modified version of {3-3} equivalent to {3-4} and solving for Δx [29]. Doing so yields the following expression for Δx:

$$\Delta x = \frac{A\varepsilon_o \varepsilon_r V^2}{2k(d_{gap} - \Delta x)^2} \qquad \{3\text{-}5\}$$

It is important to note that this equation was derived for *instantaneous voltage*, which by definition does not change with time. Thus, it is also valid for any applied constant DC voltage. However, when the applied voltage is a function of time, Equation {3-2} and thus Equation {3-5}, is no longer valid. Deviation from this DC model increases as the frequency content of the input voltage increases. However, when the frequency of the applied voltage signal is periodic and greater then the mechanical cutoff of the device this equation will hold true by setting the rms voltage of the signal, V_{rms}, equal to the applied voltage V. Once again though if this rms voltage is set as a variable function over time then Equation {3-5} no longer holds true.

Also important to note is that Equation {3-5} is only accurate for Δx roughly less than $d_{gap}/3$. Deflections greater than $d_{gap}/3$ gap generally become unstable and exhibit an effect called "snap-down" or "snap-in". This effect is attributed to the fact that the electrostatic force increases much more rapidly than that restoring force of the springs as the deflection increases [12]. The following section explains this property in further detail.

3.3.2 Electromechanical Bi-stability

The effect of "snap down" actually introduces more than just a sharp discontinuity to the static deflection response model presented. In fact, it introduces an extremely non-linear hysteresis response effect, which we can make use of in controlling the LAMA device. It has been observed that when an element is "snapped-down" by the application of a voltage greater then the snap down voltage, it will actually remain snapped down until the applied voltage is brought to a voltage level much less then the original snap down voltage, which is appropriately called the "snap-up" voltage. Figure 3-8. illustrates this effect graphically by showing idealized response curves for the "attraction" and "release" cycles.

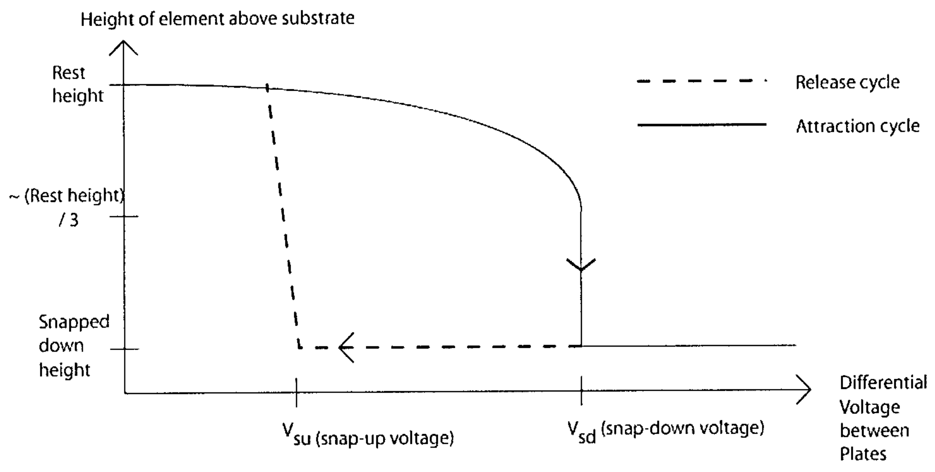


Figure 3-8. Idealized plot of element height vs. applied voltage for a micromirror The attraction and release cycles indicate a hysteresis in the device response.

In addition, while the snap-down voltage will remain constant, the snap-up voltage will vary if the initial applied voltage is stepped greater than the snap-down voltage. For instance, if snap-down voltage is 25 Volts, and the applied voltage steps from 10 Volts to 27 Volts, the snap-up voltage will be larger then if the applied voltage steps from 10 Volts to 31 Volts. This effect occurs because while the amount of stored charge increases linearly, as stated by Equation {3-1}, the electrostatic attraction upon snap-down increases in a non-linear fashion with charge. This is an important effect to note since the PWM signals applied by the control system described later in Chapter 4, are a continuous stream of stepped voltages. However, if a certain fixed applied voltage step is chosen to induce snap-down, the snap-up voltage should remain relatively constant. Thus the potential exists for a system were an element can retain its position for a varying voltage, as long as this voltage falls between snap-up and snap-down. This region can be thought of as a mechanical hold region for the element.

A significant amount of research has been driven towards eliminating this snap-down instability in an effort to increase the operating region of micromirror and other electrostatically controlled devices. Specific efforts in implementing voltage control and capacitive feedback have also been tried [7, 34, 41]. The most recent success has been achieved by Chan et al [6]. However, this effect has yet to be fully mitigated and still remains an active area of research.

3.3.3 Frequency Analysis

While the static model described by Equation {3-5} is of considerable use, and can actually provide a reasonable estimate of deflection for low frequency input, an understanding of the device limitations requires a more complete model of the system for time-varying voltage input. A common approach to modeling electromechanical systems such as the LAMA device is to conduct what is called a lumped-parameter analysis of the system. Such an analysis implies completely describing the system by a set of linear constant-coefficient differential equations. Such a system is said to be linear time-invariant (LTI) and its output can subsequently be solved using linear algebra matrix methods.

Constructing such a model and solving its system of equations was beyond the scope of this thesis. However a great deal of insight was gained by simply exploring the frequency response of the electrical and mechanical system separately. Figure 3-9. shows an overview of how separate electrical and mechanical models of the micromirror system could be integrated into a larger analysis. An excellent reference on how to conduct such an analysis for the system as a whole, particularly through the use of Langrage's Equation, is provided by D'Azzo and Houpis [15].

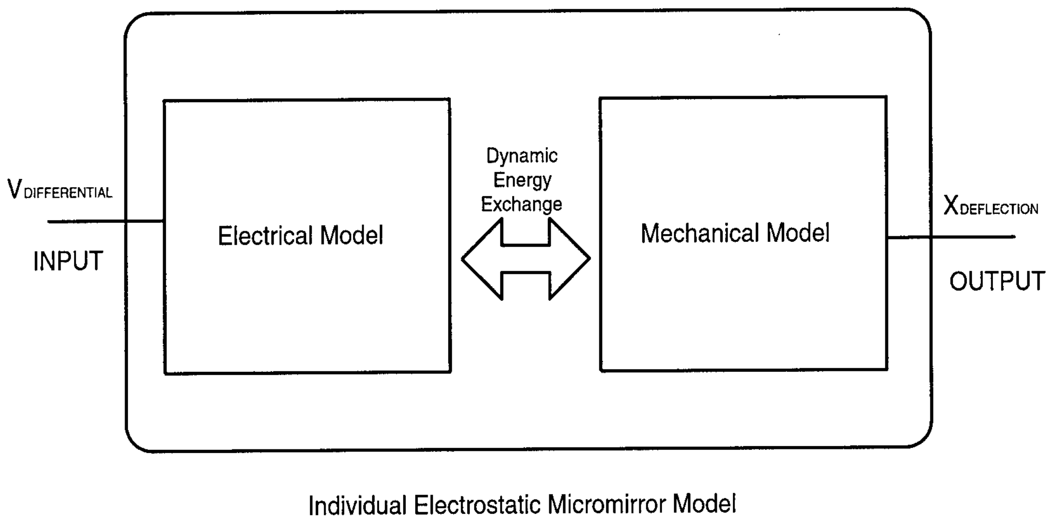


Figure 3-9. Overview of proposed electromechanical model for electrostatic micromirror

Mechanical Model

A simple way to model the frequency response of the mechanical portion of the micromirror system is to describe it in terms of a second-order mass-spring system. This model consists of a mass (the top mirror plate), suspended by a spring (flexure beams), and a dampener, which would represent any viscous dampening caused by motion through the air in the gap between the plates. Figure 3-10 shows a schematic depicting such a system.

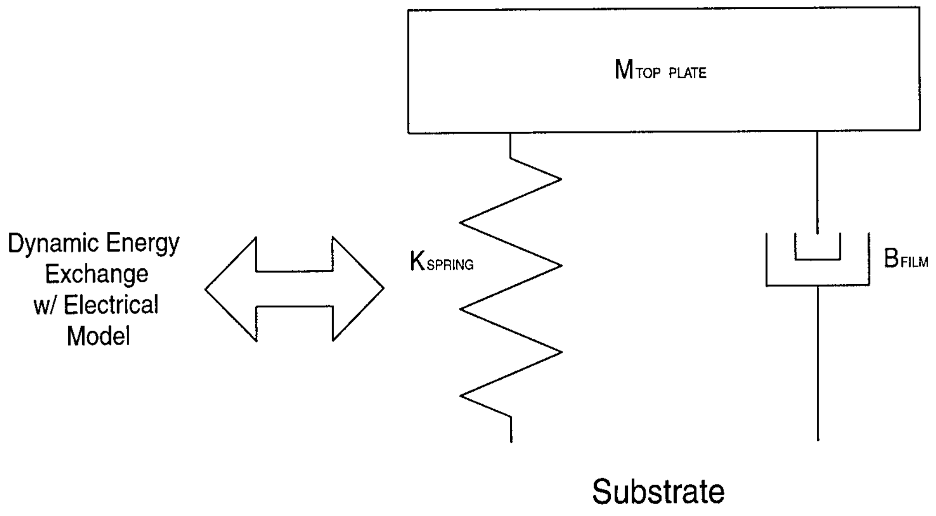


Figure 3-10. Schematic of second-order mechanical model of micromirror

The differential equation describing the dynamic response from an arbitrary forcing function, $F_{external}(t)$, is.

$$M\frac{d^2x}{dt}+B\frac{dx}{dt}+kx=F_{external}(t) \qquad \{3\text{-}6\}$$

where M is the mass, B is the dampening coefficient, and k is the spring constant. By setting $F_{external}(t)$ equivalent to the unit impulse function $u_o(t)$, and taking the Laplace transform of both sides of Equation {3-6}, the transfer function, H(s), of the system can be described as [28]:

$$H(s)=\frac{Y(s)}{X(s)}=\frac{\frac{1}{M}}{s^2+\frac{B}{M}s+\frac{k}{M}}=\frac{K\omega_o^2}{s^2+\frac{\omega_o}{Q}s+\omega_o{}^2}, \qquad \{3\text{-}7\}$$

where s is the complex frequency, $X(s)$ and $Y(s)$ are the transforms of the system inputs and outputs respectively, K is the DC gain factor, ω_o is angular natural frequency, and Q is the quality factor, which is a measure of system losses. Expressions can be found for these later three variables through appropriate equating of the coefficients [28].

$$K = \frac{1}{k} \qquad \omega_o = \sqrt{\frac{k}{M}} \qquad Q = \omega_o \frac{M}{B} \tag{3-8}$$

Equation {3-7} basically depicts the mechanical system as a second-order low-pass filter, where the Equations {3-8} describe its frequency characteristics.

There are three possible homogeneous solutions ($F_{external} = 0$) to Equation {3-6}, depending on the values of k and B. These are commonly termed the "over damped" (B > k), "under damped" (B < k), and "critically dampened" (B = k) solutions. The under damped case will produce a decaying oscillation about the equilibrium point, while the other two cases will exponentially decay towards it [36].

It has been widely reported that the resonant frequency of MEMS devices can actually shift with the application of a DC bias voltage [44]. Thus, the non-angular resonant frequency, f_o, is more accurately described by

$$f_o = \frac{1}{2\pi}\sqrt{\frac{k - k_{Vdc}}{M}} \tag{3-9}$$

where k_{Vdc} is the amount of spring constant lost with the application of a DC bias voltage.

The mass M of the top plate can calculated using

$$M = \rho A \Upsilon \qquad \{3\text{-}10\}$$

where ρ is density of polysilicon used to construct the layer, A is area of the plate, and Υ is the thickness of the layer. Note for the SUMMiT device the mass of both top plates should be calculated separately and then summed together.

Electrical Model

A similar approach can be taken for modeling the frequency characteristics of the electrical model. In fact if modeled as a simple RLC circuit the transfer function would be nearly identical in form to that given by Equation {3-7}. However, with little additional effort a more complete model was derived for this thesis by accounting for all the parasitic capacitances inherent in the micromirror circuit. A schematic of such an electrical model is depicted in Figure 3-11.

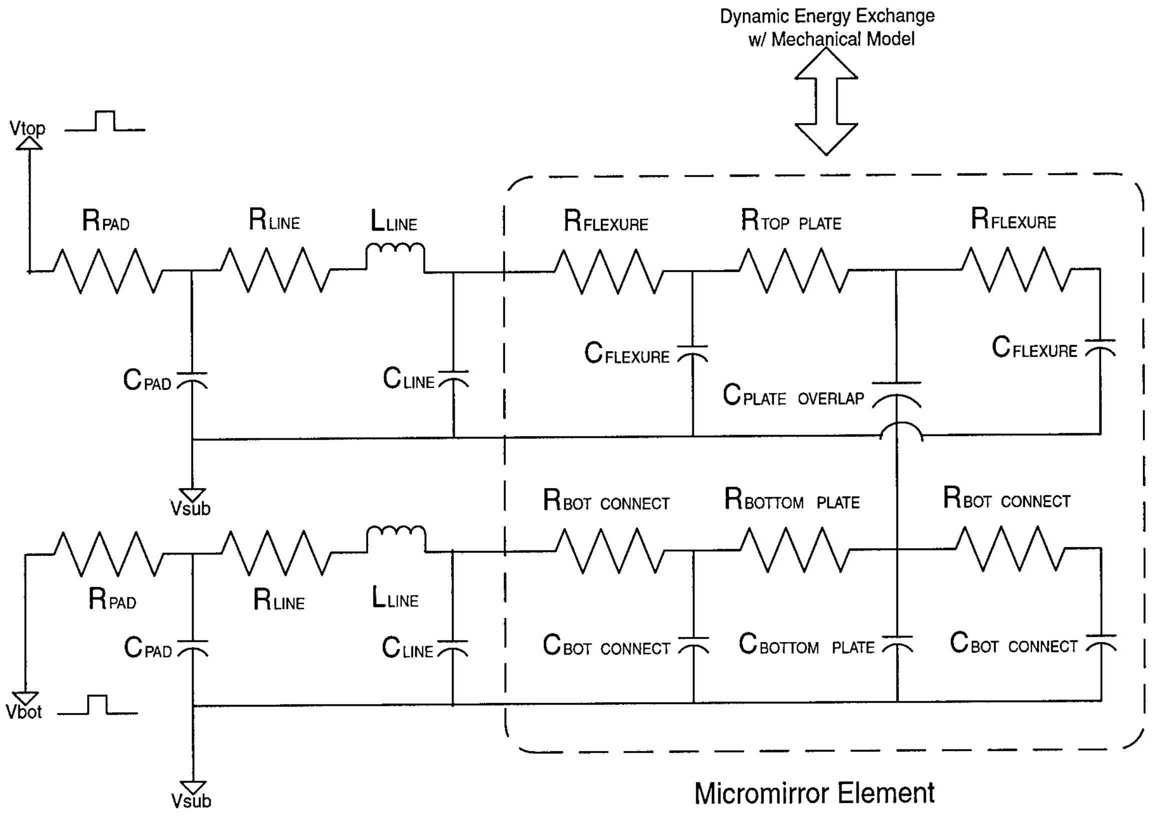


Figure 3-11. Schematic of higher-order electrical model of a micromirror

For accurate modeling of an element in the LAMA device, the array actually needs to be modeled as a whole since every column affects a part of every row and vice-versa. However, since the columns are only providing a sweeping pulse select (see Chapter 4), whose frequency content is most likely much less then that of the rows, a fairly good approximation for the LAMA device would merely look at the response of an entire row, with the all the voltages on the bottom plates of the overlapping columns set to the same voltage level. With this approximation the device model consists of 32 single element models (dashed box in Figure 3-11) in series. This would be the worst case scenario in terms of frequency response since it maximizes the capacitance and series resistances seen by each signal. Thus, it becomes a useful model for designing any future LAMA devices.

This electrical model can be analyzed by accomplishing a nodal analysis and utilizing matrix methods to solve for a response, which can then be translated into a transfer function as before. However solving such a large network by hand is a mind numbing exercise, therefore its is easiest to accomplish a frequency analysis using an electronic simulation package such a SPICE.

While the development of a complete model is well beyond the scope of this thesis, it is worthwhile to note some of the effects such a model could account for. These effects would include surface variations in the plates, a mapping of the force distribution across the top plate, corresponding deformations induced across the plate as deflection varies, fringing effects in the electric field, coupling between adjacent elements, snap-down and snap-up instabilities, parasitic capacitances inherent in the device, transmission line effects inherent in packaging, and an accurate formulism depicting the transient and frequency response of the mirror, to name a few. Compile this list of effects with a variety of mathematical methods that can be used to analyze their effects and one becomes quickly aware of the challenges inherent in developing a rigorous model. A variety of computer simulation programs, such as IntelliCAD™ and ANSYS, which utilize finite element techniques, have been constructed in contribution to these research efforts [44, 46]. The development of an accurate model is a valuable exercise because it will allow more accurate testing and simulation of MEMS device performance prior to fabrication. In a commercial setting modeling saves MEMS device development costs, and speeds research efforts. Hence it remains a challenging problem in the applied mathematics community.

3.4 Device Modeling in Optical Far-field

Another important characteristic of the array to consider is how it can potentially act as a bi-stable phase modulator. More specifically, it is useful to model the far-field distribution of a laser beam, reflected off of the surface of the micromirror array.

Undeflected, the mirror array essentially acts as an aperature function, $t_o(\xi,\eta)$, defined two-dimensionally as follows,

$$t_o(\xi,\eta) = \left(rect(\frac{\xi}{a},\frac{\eta}{a}) \otimes \frac{1}{d^2} comb(\frac{\xi}{d},\frac{\eta}{d}) \right) rect(\frac{\xi}{W},\frac{\eta}{W}) \qquad \{3\text{-}11\}$$

where a is width of the elements, d is period between elements, $\otimes$ denotes convolution, W is the width of the entire array, and ξ and η are the horizontal and vertical coordinates in the plane of the aperature. Equation {3-11} is an accurate description of the micromirror array since the light incident on the spaces between mirror elements undergoes so much diffraction passing through the slit virtually none of it is reflected. Assuming the incident laser is a perfectly monochromatic plane wave of unit amplitude, the far-field pattern of the reflected field is simply the Fourier transform of this function. This translates to the following intensity profile described by,

$$I(u,v) = \left| \left(a^2 \mathrm{sinc}(a\frac{u}{\lambda r}, a\frac{v}{\lambda r}) comb(d\frac{u}{\lambda r}, d\frac{v}{\lambda r}) \right) \otimes \mathrm{sinc}(W\frac{u}{\lambda r}, W\frac{v}{\lambda r}) \right|^2 \qquad \{3\text{-}12\}$$

where r is the distance from the mirror, and u and v are the horizontal and vertical coordinates in the image plane [18].

When the mirror array is set to hold an arbitrary bi-stable phase pattern the aperture function can no longer be described by an analytical formula. Thus computation of the far-field pattern requires a discretized analysis of the system. This essentially involves generating a complex phase map to represent the positioning within the array and then running a discrete Fourier transform algorithm, such as a fast-Fourier transform (FFT), to generate a projected far-field pattern. This can be easily accomplished using a simulation software package such as MATLAB®.

3.5 Static Fringe Method

The experimental method used to estimate the total spring constant of the flexure beams that support the micromirror elements is termed the static fringe method. This method uses the positioning of fringes produced by the devices when seen through a laser interferometric microscope. The fringe pattern seen on the microscope is effectively the result of constructive and destructive interference of the light reflected off of the device with that of the reference beam internal to the interferometer. The fringe spacing is determined by relative tilt of the device with respect to the incident laser light. The amount of phase accrued by the laser light reflected off of the device directly relates to the position of the fringes. When an element on the mirror is deflected, a certain amount of phase delay is accrued in the light reflected off of that element with respect to the other elements (this is the definition of phase modulation). This phase delay is seen as a shifting of the fringe lines of that element with respect to the fringe lines of the other elements. When a mirror element is deflected such that the phase delay accrued by the reflected light is 2π radians its fringe lines will be shifted such that they overlap their original position when the element was undeflected. Thus at these deflections the fringes across the surface will remain *static* or unchanged with respect to zero deflection. The 2π phase delay translates to a change in distance of one wavelength, λ, of the light being reflected. So the actual amount of deflection at this state, dubbed the static fringe deflection d_{static}, will be

$$d_{static} = \frac{\lambda}{2} \qquad \{3\text{-}13\}$$

Using this knowledge, one can experimentally step up the DC voltage difference across a set of elements until this static fringe phenomenon is observed, and then record this voltage. From Equation {3-13}, given the knowledge of the actual wavelength of the laser light the static fringe deflection can be obtained. This value and the recorded voltage can be used to estimate the total spring constant of the mirror elements by applying Equation {3-5}. This in turn allows us to use Equations {3-5} and {3-8} to project a static deflection response for DC voltages and estimate the resonant frequency.

IV. Theory and Implementation of Line-Addressable Control

"Make everything as simple as possible, but not simpler."

-Albert Einstein (1879-1955) [38]

4.1 Overview

The design and implementation of the control system for the LAMA device is the heart of this thesis. Initially, three modes of operation of the device were conceived. Each of these is described in detail in Section 4.2. The actual control system implementation used to accommodate each of these modes, and each of the system's core building blocks is described in Section 4.3.

4.2 Theory of Line-Addressable Operation

Bi-stable control of individual elements in the LAMA device is predicated on the electromechanical bi-stability inherent in the micromirrors as described in Section 3.3.2. Specifically, I conceived of two separate control modes that utilize this property. The first is based on a scheme first proposed by Jaecklin for a line-addressable tilt array [24]. This mode allows the application of an arbitrary static bi-stable pattern, however requires the array to be reset to its original position before a new pattern can be applied. The second is an original scheme that eliminates this difficulty, allowing dynamic switching between patterns. In addition, this second mode affords additional benefits upon implementation. Each of these modes are respectively described in Sections 4.2.1 and 4.2.2, respectively.

The primary limitation inherent with each of these bi-stable control modes is that they require operation below the mechanical cutoff frequency of the device. Section 4.2.3 discusses the possibilities afforded with operation above the mechanical cutoff.

4.2.1 Conventional Bi-stable Operation

In the first mode of operation, two different sets of PWM signals are applied to the rows and columns of the LAMA device. Both of these sets of signals operate at the same frequency, f, which is below the mechanical cutoff of the device. Figure 4-1. provides an example pair of these signals that will be referred to for the remainder of this discussion. Although the LAMA device described in Chapter 3 is square, it will become apparent that this scheme can be applied to any arbitrary MxN rectangular array with a similar structure.

In this mode, the PWM signals applied to the rows are at a DC offset above that of the signals applied to the columns, however the difference between active and inactive states for each signal is not necessarily the same. This is done to maintain a net voltage difference across the individual elements such that the electric field is always directed towards the substrate. The substrate itself is held at the active state voltage of the columns. This is done to minimize the voltage difference between the bottom mirror plate and the substrate, as well as reduce the voltage difference between the hidden flexure beams and the substrate.

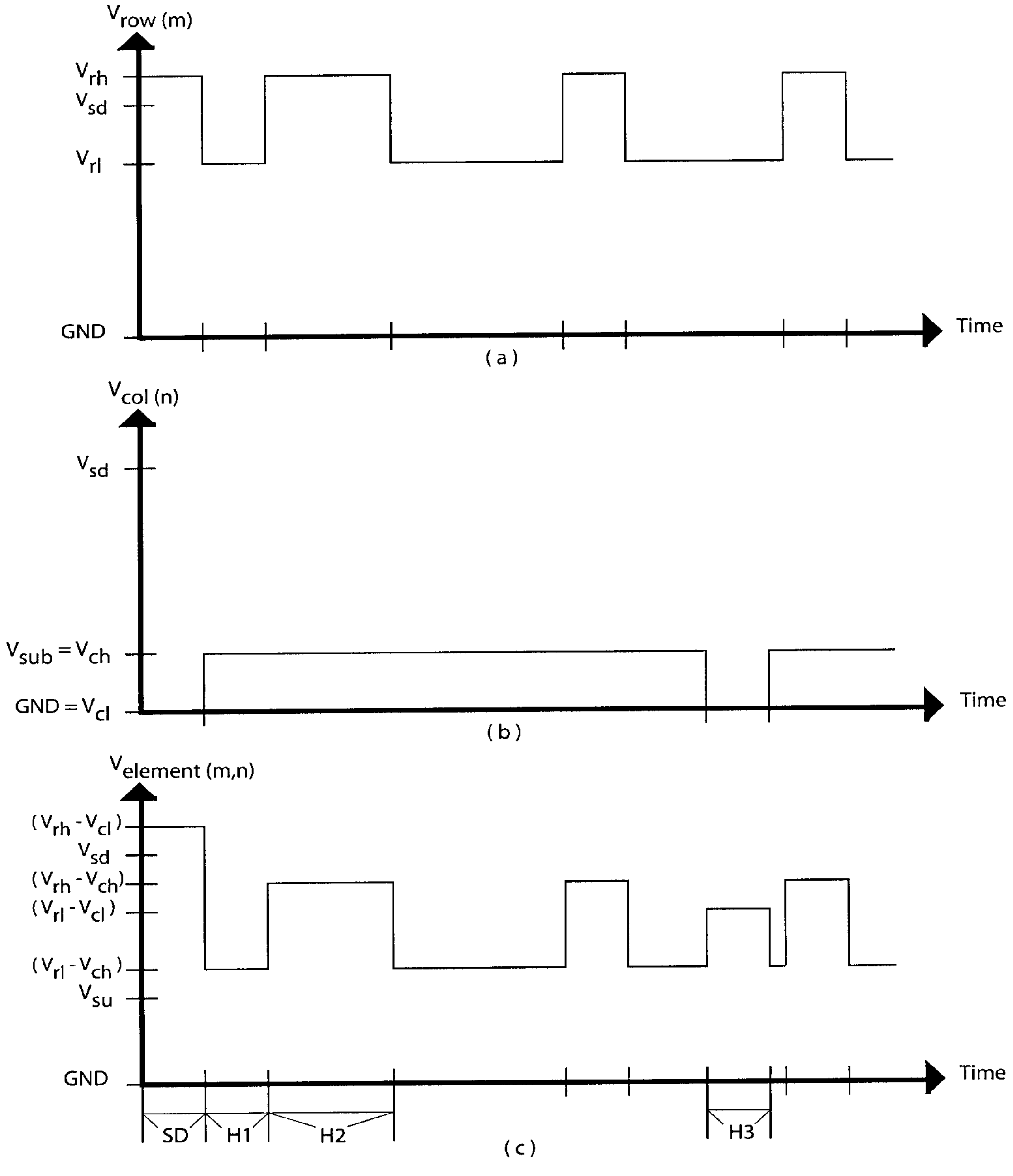


Figure 4-1. Example PWM signals for conventional bi-stable operation.
An arbitrary row voltage signal (a) and column signal (b) produce the voltage difference signal (c) for the intersecting element. In (c), SD indicates snap down state, GND indicates ground (0 Volts), and H1, H2, and H3 indicate each of the hold states. In this example element (m,n) is positioned in the snap down position, where V_{sd} represents the snap down voltage. Note the time the column signal spends in the low state has been exaggerated to ease visualization.

The column PWM signals, $V_{COL(n)}$, are applied as a means of scanning across the array and isolating each column separately for a specific period of time. Thus they each have the same duty cycle, $DutyCycle_{COL}$, defined as

$$DutyCycle_{COL} = \frac{1}{N} \tag{4-1}$$

where N is the number of columns in the device. Adjacent column signals are separated by a temporal phase difference, $\Delta\tau_{COL}$, defined as

$$\Delta\tau_{COL} = DutyCycle_{COL} \cdot T \tag{4-2}$$

where T is period of each signal $(T = 1/f)$.

The PWM signals applied to each of the rows, $V_{row(m)}$, effectively determine which elements in that row are snapped down and which remain in the upright position. This becomes apparent when one looks at the voltage difference seen by an individual element, $V_{element(m,n)}$, as shown in Figure 4-1 (c). The voltage difference between the plates of an individual element can have 4 possible states. The first of these is called the snap-down state, since the voltage difference is greater than the snap down voltage V_{sd} and causes an undeflected element to assume its snapped down position. It is important to note that this state can only occur when the column state is low, which is how individual element selection is accomplished. The remaining three states are all labeled as hold states since the voltage differences associated with each state are all in between the snap up, V_{su}, and snap down voltage levels. Thus once an element experiences a snap down state it remains snapped down. However, the undeflected elements will experience a significant amount of "wiggle" as the voltage difference between plates varies. Thus, to hold the pattern with minimum "wiggle" in the elements, all the row signals should be set to a constant low state, V_{rl}, after the pattern is applied. To completely reset the array such that all elements are undeflected, one can either raise the active state of the column, V_{ch}, such that V_{rl}-V_{ch} is less then $V_{su,}$ or completely ground both sets of signals.

Using the signal definitions shown in Figure 4-1., the aforementioned conditions for each of these four states is summarized by

$$\begin{aligned} &V_{rh} - V_{cl} > V_{sd} \rightarrow SD \\ &V_{su} < V_{rl} - V_{ch} < V_{sd} \rightarrow H1 \\ &V_{su} < V_{rh} - V_{ch} < V_{sd} \rightarrow H2 \\ &V_{su} < V_{rl} - V_{cl} < V_{sd} \rightarrow H3. \end{aligned} \qquad \{4\text{-}3\}$$

These conditions are greatly simplified when V_{cl} is set to ground (0V) and states H2 and H3 are set equivalent to each other by setting the voltage difference, Δ, between high and low voltage states, equal for the rows and columns $[\Delta = (V_{ch} - V_{cl}) = V_{ch} = (V_{rh} - V_{rl})]$. Under these conditions Equation {4-3} simplifies to

$$\begin{aligned} &V_{rh} > V_{sd} \rightarrow SD \\ &V_{su} < V_{rl} - \Delta < V_{sd} \rightarrow H1 \\ &V_{su} < V_{rl} < V_{sd} \rightarrow (H2 \,\&\, H3). \end{aligned} \qquad \{4\text{-}4\}$$

Since reliable operation is dependent upon these conditions, it becomes apparent why accurate characterization of the snap up and snap down voltages of the LAMA device is important.

4.2.2 Enhanced Bi-stable Operation

In the second mode of operation two similar sets of PWM signals are applied to the rows and columns of the LAMA device. The theory for this mode also extends to any MxN array. The row and column signals applied, in this mode, are identical in function to those described for the previous mode, so Equations {4-1} and {4-2} continue to hold. The difference is that both sets of PWM signals are applied with their low states set to the same voltage level, $V_{rl} = V_{cl}$. The obvious choice for this uniform low state is ground, which is how the example signals are presented in Figure 4-2.

The voltage difference between active and inactive states is not the same for these signals. In addition, since the electric field is not always directed towards the substrate, the substrate is also grounded. This is done to prevent any voltage difference between the top mirror plate and the substrate from having any significant effect on the control of each element.

The primary advantage of this mode over the previous is that it allows dynamic switching between patterns. Once again this becomes apparent when viewing the voltage difference seen by an individual element, as shown in Figure 4-2. (c). In addition, since the signals are all referenced to ground and some other high state the direct CMOS method of amplification can be used. This affords a simpler more efficient means of amplification.

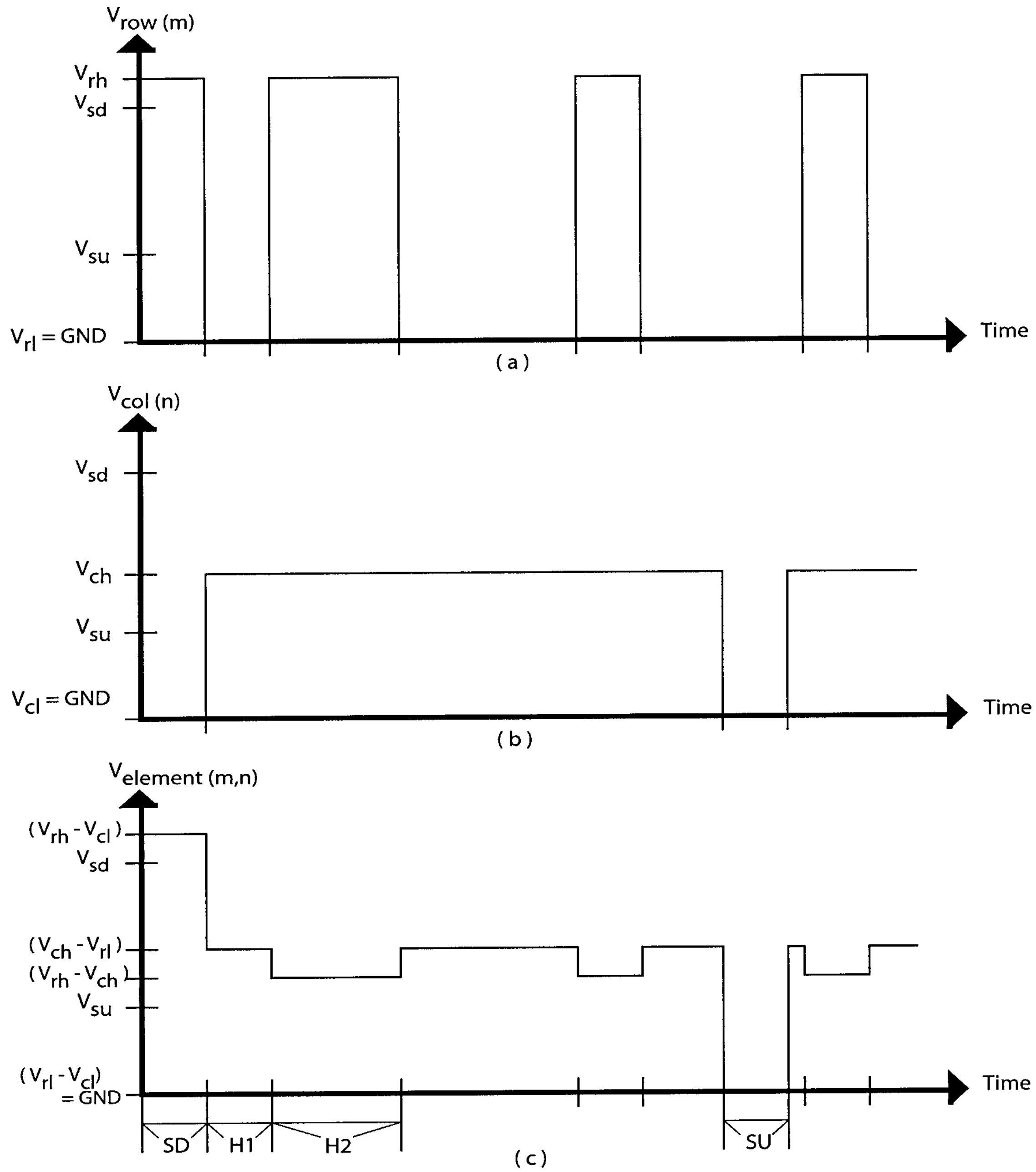


Figure 4-2. Example PWM signals for enhanced bi-stable operation.
An arbitrary row voltage signal (a) and column signal (b) produce the voltage difference signal (c) for the intersecting element. In (c), SD indicates snap down state, SU indicates snap up state, GND indicates ground (0 Volts) and H1 and H2 indicate each of the hold states. In this example element(m,n) is first positioned in the snap down position, where Vsd represents the snap down voltage and then it is snapped back up. Note the time the column signal spends in the low state has been exaggerated to ease visualization.

Based on the signal definitions supplied in Figure 4-2. the conditions for operation in this mode can be defined by Equations {4-5}.

$$\begin{aligned}
&(V_{rh} - V_{cl}) > V_{sd} \rightarrow SD \\
&V_{su} < (V_{ch} - V_{rl}) < V_{sd} \rightarrow H1 \\
&V_{su} < (V_{rh} - V_{ch}) < V_{sd} \rightarrow H2 \\
&(V_{rl} - V_{cl}) < V_{su} \rightarrow SU
\end{aligned} \qquad \{4\text{-}5\}$$

The primary drawback for this system is that once again there is potential for "wiggle" in elements that are not snapped down. For the previous mode, this is not a difficulty since once a static pattern is set, the row signals are turned off and only a slight "wiggle" is introduced by the column scan. If absolutely necessary, this "wiggle" in could be eliminated by turning up the frequency. In this mode, however the "wiggle" truly matters since we want the signals to be applied continuously to allow dynamic changing of patterns. Thus to eliminate any "wiggle" the active states of the row and column signals can be set such that H1=H2. By algebraically solving $V_{ch} - V_{rl} = V_{rh} - V_{ch}$ for V_{ch}, and setting the low states equal to ground, a simplified set of conditions for "no wiggle" is found:

$$\begin{aligned}
&V_{rh} > V_{sd} \rightarrow SD \\
&V_{su} < \left(V_{ch} = \frac{V_{rh}}{2} \right) < V_{sd} \rightarrow H1, H2 \\
&V_{cl} = V_{rl} = GND \rightarrow SU
\end{aligned} \qquad \{4\text{-}6\}$$

It is important to note that the rest state assumed when the mirror elements are not snapped down is not completely undeflected, unlike the previous mode. The mirror elements in the "up" state are deflected according to the hold voltage. However, since the mirror response is nonlinear it is likely that the extent of this deflection may not be significant. Of course this is entirely dependent upon the application in which this device would be used.

4.2.3 Operation Above Mechanical Cutoff

The previous two modes of operation provide the ability to apply arbitrary bi-stable patterns to the LAMA device. However, the inherent limitation in these modes is that they must both operate at a frequency below the mechanical cutoff of the micromirror device. There may be applications were the ability to operate at higher frequencies is imperative. Therefore it is worthwhile to explore the potential patterns that can be applied by varying the rms-voltages applied to each line.

The analysis begins by assuming that we are dealing with DC line voltages for simplicity. Any actual implementation would utilize PWM signals, whose rms-voltage, as defined by Equation {2-9}, would take the place of these DC signals. The manner in which this is done is inconsequential, and can be modeled as applying various DC voltages to the device. Specifically our analysis will focus on how the line voltages applied to the array overlap to produce a differential voltage pattern across the array.

We begin by first defining the input voltages to a line addressable MxN array as vector quantities:

$$V_{row} = \begin{bmatrix} a_m \\ a_{m+1} \\ a_{m+2} \\ \dots \\ a_{m+M} \end{bmatrix} \qquad \{4\text{-}7\}$$

$$V_{col} = \begin{bmatrix} b_n & b_{n+1} & b_{n+2} & \dots & b_{n+N} \end{bmatrix}$$

where V_{row} and V_{col} are the vectors defining the row and column voltages respectively. To compute the voltage difference between elements across the array, we define the matrix operation "Ξ" as follows:

$$V_{row} \Xi V_{col} = V_{array} = \begin{bmatrix} |a_m - b_n| & |a_m - b_{n+1}| & \dots & |a_m - b_{n+N}| \\ |a_{m+1} - b_n| & |a_{m+1} - b_{n+1}| & \dots & |a_{m+1} - b_{n+N}| \\ \dots & \dots & \dots & \dots \\ |a_{m+M} - b_n| & |a_{m+M} - b_{n+1}| & \dots & |a_{m+M} - b_{n+N}| \end{bmatrix} = \begin{bmatrix} c_{m,n} & c_{m,n+1} & \dots & c_{m,n+N} \\ c_{m+1,n} & c_{m+1,n+1} & \dots & c_{m+1,n+N} \\ \dots & \dots & \dots & \dots \\ c_{m+M,n} & c_{m+M,n+1} & \dots & c_{m+M,n+N} \end{bmatrix} \qquad \{4\text{-}8\}$$

where V_{array} is the differential voltage map across the array. If we assume a steady state solution and that these are DC voltages we can recursively apply Equation {3-5} to each element in V_{array} and generate a deflection map for the array. However, since iterating solutions for each element would be computationally intensive, a faster method of generating a deflection map would be through the use of a pre-generated lookup table listing the deflection response for a specified voltage.

Figure 4-3. shows a simple voltage ramp across the rows as an example differential voltage map. Figure 4-4. shows the corresponding deflection map that is created using the above method. Both were created using the MATLAB® code included in Appendix G which performs the above analysis for any set of line inputs for the LAMA device.

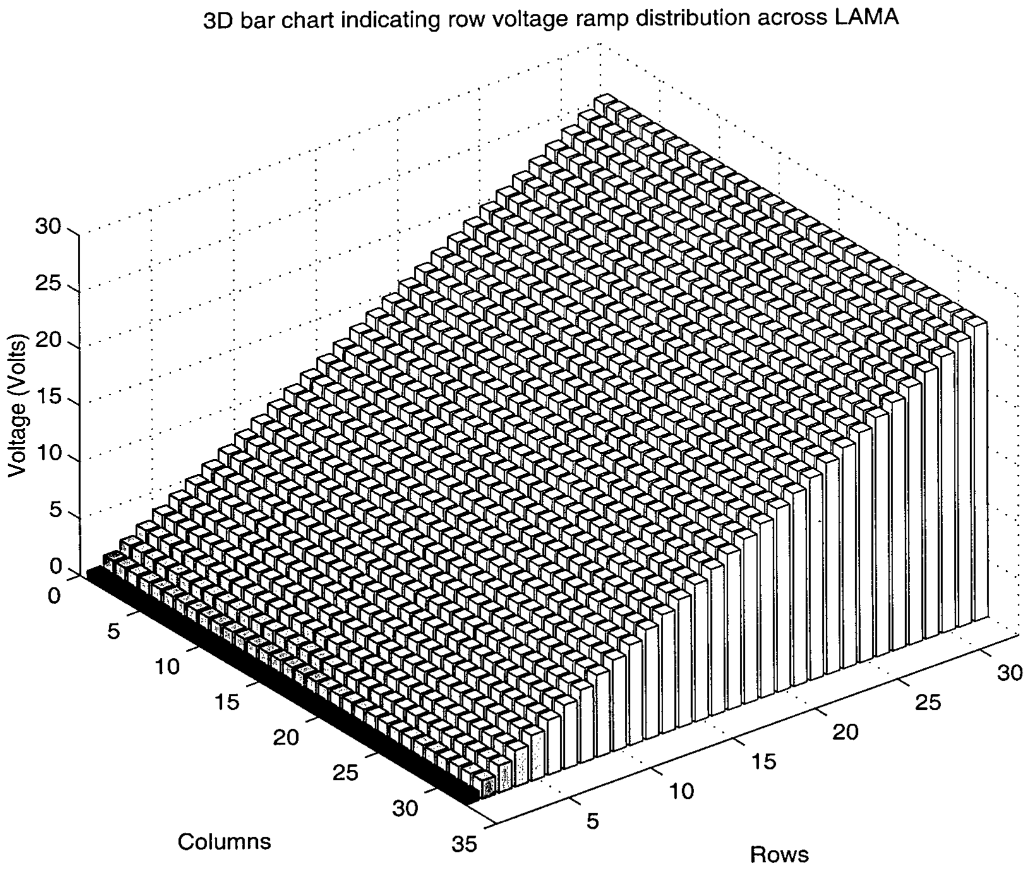


Figure 4-3. 3D bar chart indicating example differential voltage ramp across LAMA. The map corresponds to a voltage ramp across the rows with the columns grounded.

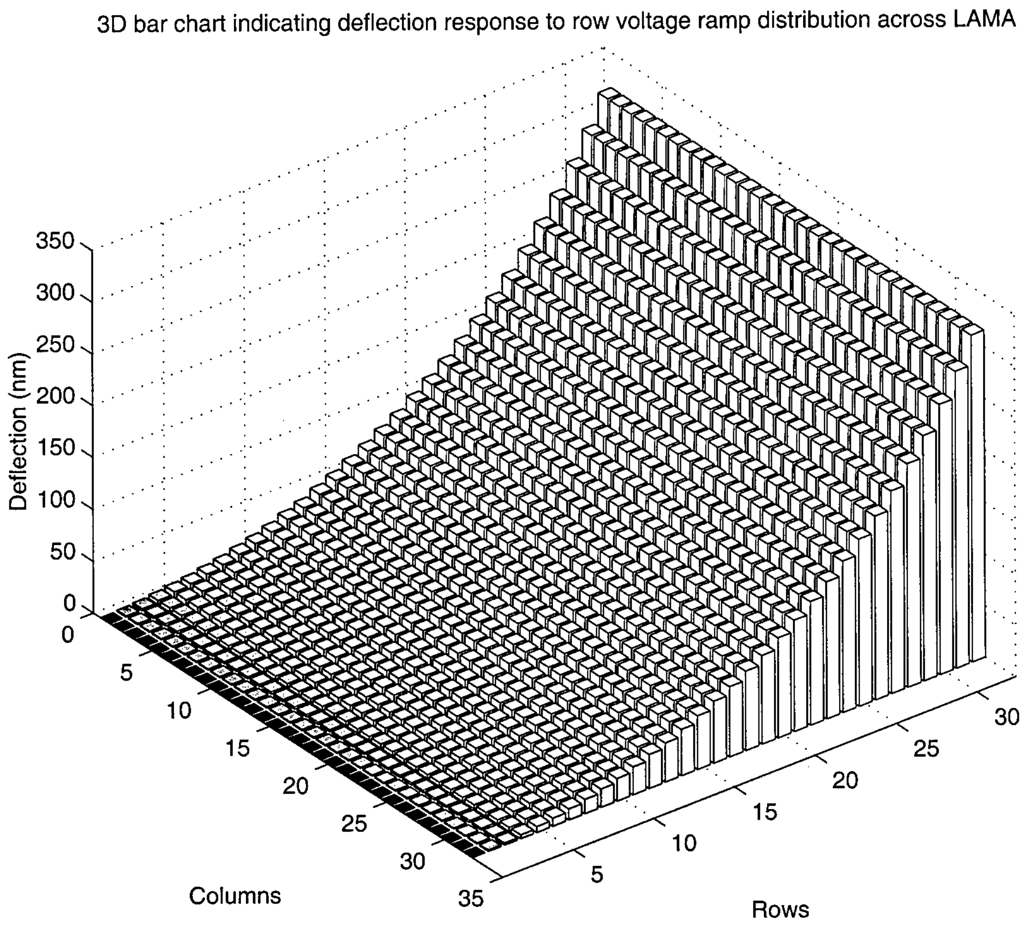


Figure 4-4. 3D bar chart indicating a SUMMiT LAMA device response to a differential voltage ramp. The deflection response was estimated for a static fringe voltage of 26V.

It should be apparent that such a ramp can be generated in 8 different directions across the array (up, down, left, right, and four diagonals), when both positive and negative voltage are used. The appearance of negative voltages can be achieved through purely positive voltages by raising the voltage level applied to the substrate. Thus the LAMA device can potentially be used in small-angle beam steering applications. More aggressive beam steering can be accomplished by having the LAMA device function as a reflective rectangular phase grating, where the width of the grating can be adjusted. Several other additional patterns can be created by manipulating the line inputs and applying Equation {4-8}. All of these patterns can manipulate the reflected far-field pattern of a laser which makes operation above mechanical cutoff potentially useful in an optical communication system.

4.3 Control System Design and Implementation

To experimentally demonstrate LAMA device operation in each of these three modes, an appropriate control system utilizing PWM signals was developed. Its concept and architecture was initially devised in part by W. Cowan and can be separated into three main parts. The first part is the software component, described in Section 4.3.1. The software allows the user to create, load, and save any desired bi-stable deflection pattern and subsequently apply it to the micromirror array when desired. The second part is the PC controller board, which generates the the PWM signals as defined by the software. Section 4.3.2 describes the details in its design. Finally, to achieve the voltages necessary to achieve actual deflection these PWM signals must be amplified before they are applied to the LAMA device. Section 4.3.3 describes the pulse amplification board to which the actual micromirror device is mounted. Figure 4-5 shows a graphical overview of this system.

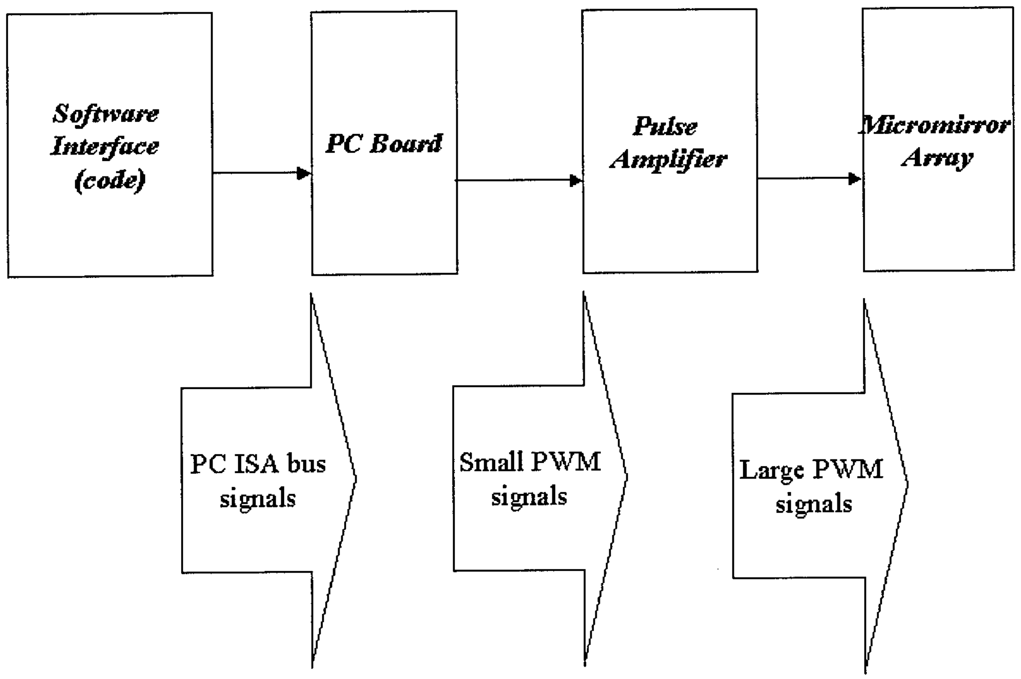
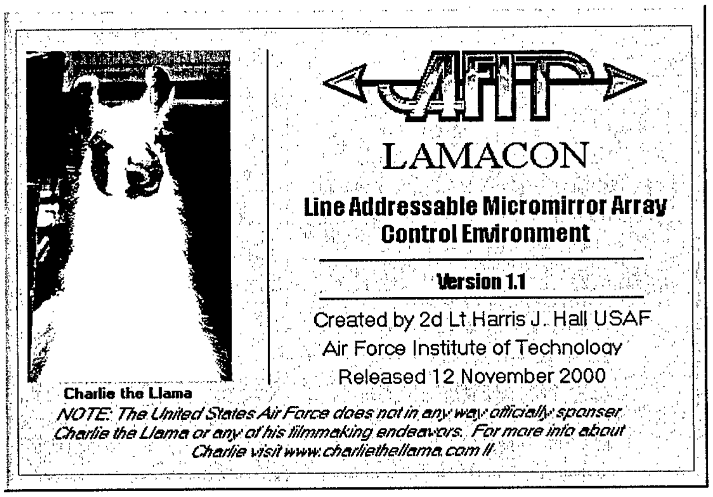


Figure 4-5. Overview of Experimental Control System Design

4.3.1 Software Interface

Many of the applications in which micromirrors are utilized require high speed control and the obvious method of achieving this is by using a digital computer. Pulse width modulation, as previously explained in Section 2.5, can be easily accomplished using digital logic. To generate the PWM signals needed, the shape of the pulse streams is stored in the computer memory, which is then transferred to the memory on the PC Control Board. This memory is physically created using CMOS logic. To generate physical signals once a pattern is downloaded to memory, the electrical contents of that memory are streamed out. The function of the software interface of course, is to store the appropriate patterns to memory.

The software control interface designed for the LAMA device is appropriately called LAMACON. It is designed to run in a Windows 95/98 operating system on a PC platform and consists of two parts. The first is a Graphical User Interface (GUI) that offers simple point and click control for the user. This allows the user to choose which mode of operation is to be tested. Specifically, the options include setting the array low, setting the array high, applying a forward-slash pattern to the memory for testing, applying voltage ramps across the rows of the device (to test operation above cutoff), and finally independent bi-stable operation. This final selection allows the user to create, load, and save arbitrary bi-stable patterns and then apply them when desired. Since it can directly switch from one pattern to the next, both bi-stable modes of operation can be tested. Figure 4-6 shows the splash screen that appears on start up and Figure 4-7 shows the actual GUI environment as seen by the user.


LAMACON

Line Addressable Micromirror Array Control Environment

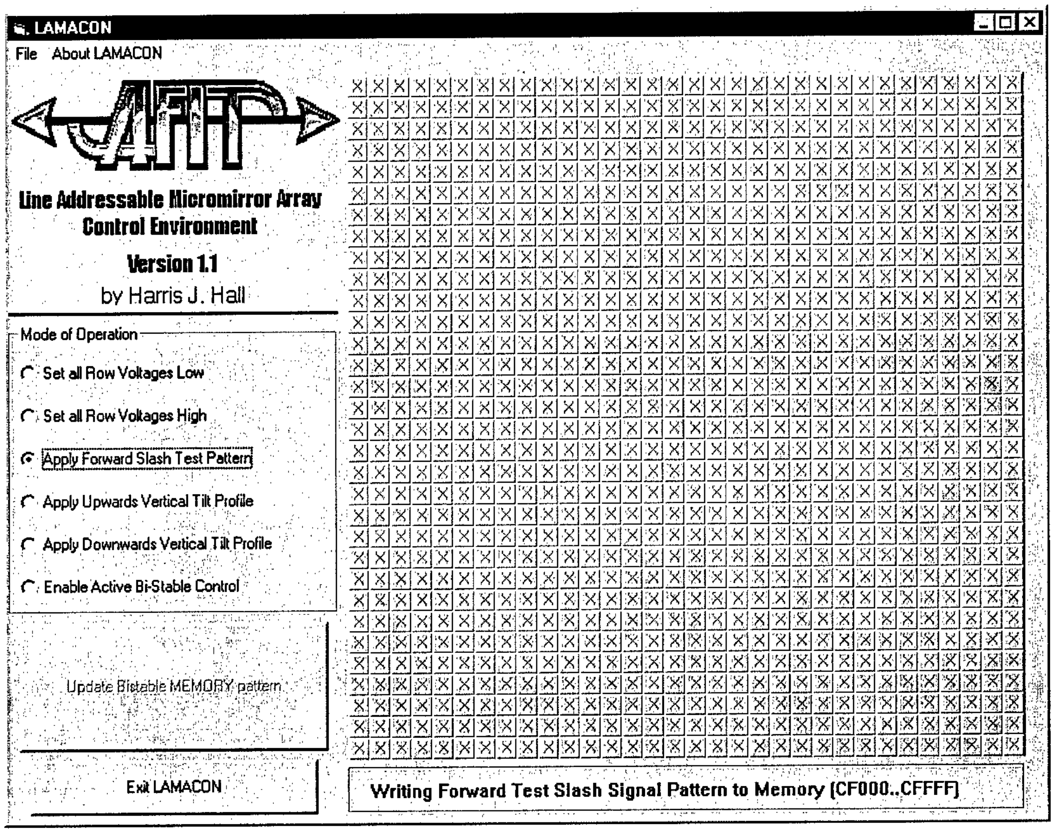
Version 1.1

Created by 2d Lt Harris J. Hall USAF
Air Force Institute of Technology
Released 12 November 2000

Charlie the Llama

NOTE: The United States Air Force does not in any way officially sponser Charlie the Llama or any of his filmmaking endeavors. For more info about Charlie visit www.charliethellama.com !!

Figure 4-6. Image of LAMACON Splash Screen


LAMACON

File About LAMACON

Line Addressable Micromirror Array Control Environment

Version 1.1

by Harris J. Hall

Mode of Operation

( ) Set all Row Voltages Low

( ) Set all Row Voltages High

(•) Apply Forward Slash Test Pattern

( ) Apply Upwards Vertical Tilt Profile

( ) Apply Downwards Vertical Tilt Profile

( ) Enable Active Bi-Stable Control

Update Bistable MEMORY pattern

Exit LAMACON

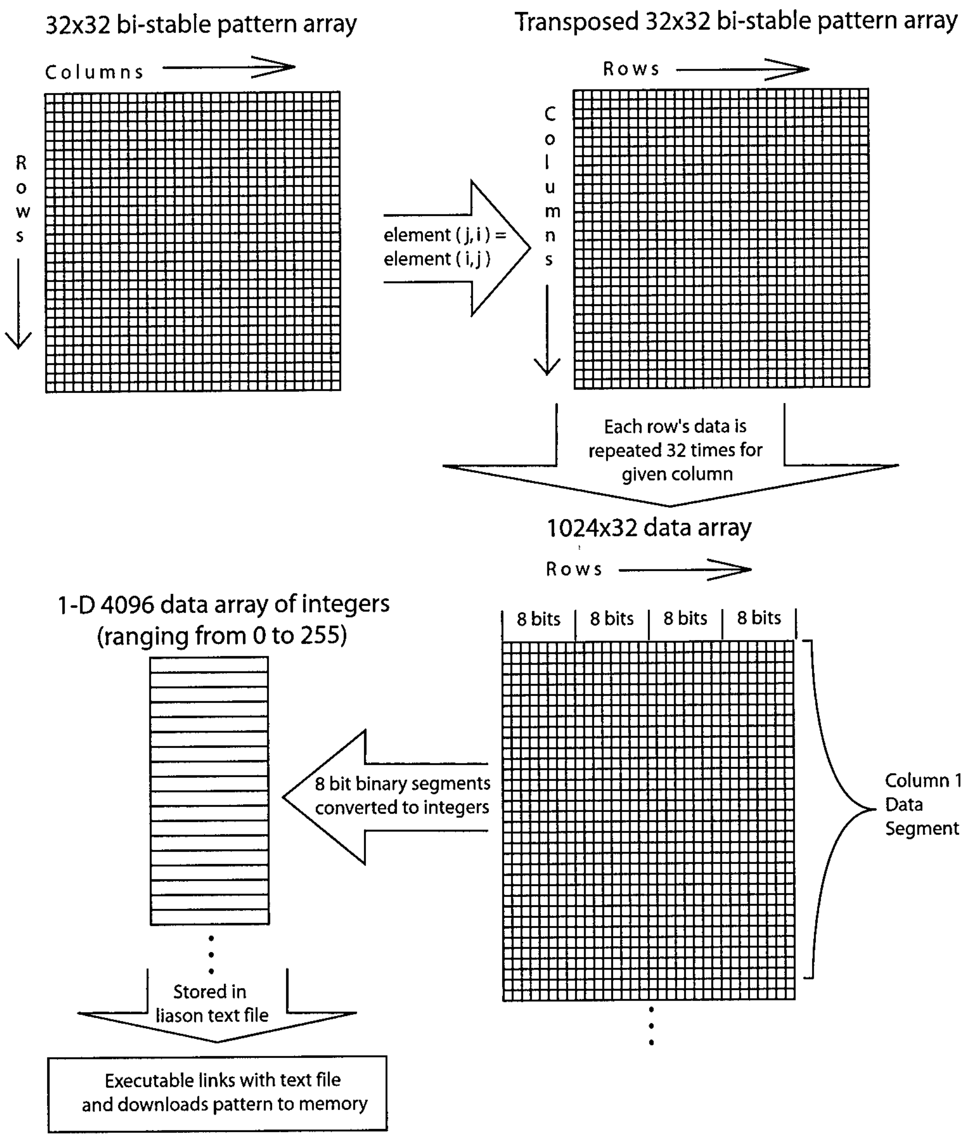
Writing Forward Test Slash Signal Pattern to Memory (CF000..CFFFF)

Figure 4-7. Image of LAMACON GUI screen.
This view indicates the application of a forward slash test pattern. "X's" displayed across the array signify a bi-stable pattern is not being applied.

The second part of LAMACON is a series of MS-DOS executable files that are called by the GUI to physically download these patterns to memory. These executable files were created using Borland C++ version 3.1 and make use of its exclusive *pokeb()* function. The actual C++ code used to generate them can be found in Appendix B. The *pokeb()* function allows direct assignment of 8-bit character type data to physical memory. It is important to note that this type of operation is not possible using the Win32 Application Program Interface (API) that is included with Windows 95, which the GUI is designed to use. Microsoft intentionally designed its Win32 API to prevent programmers from tampering with the physical memory. Instead they created a virtual memory management system that gives programmers control over virtual memory blocks which are then allocated to physical memory in a manner best suited to optimize execution efficiency. While this type of memory management system has several benefits associated with it, the one drawback is that it prevents any direct access to hardware. The design of any and all device control requires the use of a Device Driver Kit (DDK) under the Win32 API. Since DDK design can be a complicated venture to the casual engineer, the use of a series of MS-DOS based executables was deemed the most practical alternative for this work. While separately calling these executables from the GUI is sufficient for laboratory testing purposes, it remains a rather roundabout method of control. Thus DDK design is an important facet to explore with regards to any future form of system optimization. Further details on Win32 memory management can be found in Toth's online book *Visual C++ 4 Unleashed* [45].

The GUI was constructed using Microsoft Visual Basic 5.0 and consists of a form module, which defines the layout of the controls and their corresponding event procedures, and an additional function module. This function module contains a library of functions that accomplish all file I/O and necessary calling of external executables. Appendix A includes all the relevant functions and event procedures used in its construction. The actual form file that defines the main control form is not presented in its entirety since it would include hundreds of pages of automatically generated code defining the layout. As a testament to how much layout code actually exists in these form files, the *frmSplash.frm* file, that merely defines the splash screen shown in Figure 4-6 is also listed in Appendix A.

In the bi-stable mode, the GUI interprets the array pattern as an array of 1's and 0's indicating deflected and undeflected states respectively. When saved to disk, the array is stored as an ASCII text file with the *.pat extension, where each row is a separate line and each element is space delimited. This allows the pattern to be easily interpreted for use in other programming environments, such as MATLAB®, which the far-field simulation code is written in. For the 32x32 pattern to be downloaded to memory, the data must be appropriately converted to a series of 8-bit segments for the *pokeb()* function to interpret in the MS-DOS executable file. This process is accomplished by the *createliasonfile()* function in the *modLAMAfunctions.bas* module which is part of the code contained in Appendix A. First, the 32x32 pattern array is transposed [element (*i, j*) -> element (*j, i*)] to compensate for the fact that Visual Basic uses column major ordering for its arrays while the C++ derived executables use row major ordering. Next, each row in this transposed 32x32 array is repeated 32 times yielding a 1024x32 array. The memory allocated for this system actually permits 32 discrete levels of pulse width for each segment of row signal that is designated for each column. For bi-stable operation however we do not need this feature, and will merely set the pulse width to either its maximum value or its minimum value. Next, the rows in this 1024x32 array are sequentially read across in 8-bit word segments, in a serial manner. These 8-bit word segments are then stored as integers in a 1-D character array that is 4096 entries long. Finally, this character array of integers (which range from 0 to 255), is written to a separate ASCII text file. This text file acts as a liason between the GUI and the MS DOS executable file that subsequently reads in the text data and uses the *pokeb()* function to download the data segments to memory. This entire process is outlined in Figure 4-8.



32x32 bi-stable pattern array
Columns
Rows
element (j, i) = element (i, j)
Transposed 32x32 bi-stable pattern array
Rows
Columns
Each row's data is repeated 32 times for given column
1024x32 data array
Rows
8 bits
8 bits
8 bits
8 bits
Column 1 Data Segment
8 bit binary segments converted to integers
1-D 4096 data array of integers (ranging from 0 to 255)
Stored in liason text file
Executable links with text file and downloads pattern to memory

Figure 4-8. Outline of software array manipulations for bi-stable pattern loading

The forward slash test pattern and the two voltage ramp patterns are applied directly by their executable files through direct variation of these 8-bit word segments. For the voltage ramp patterns, the frequency content of the PWM signals is maximized by utilizing these 32 discrete levels inherent within each column data segment.

4.3.2 PC Control Board

The PC Control Board receives as inputs the memory block (4096 word x 8 bit) previously defined by the software interface and converts it into actual PWM streams that are applied to the rows and columns of the micromirror device. The board is constructed from a JDR Microdevices 16-bit prototype board, which inherently includes much of the I/O decode logic necessary for use in an IBM PC-AT based architecture. Details on the structure and capabilities associated with this type of board can be found in its user's manual [3]. The card was previously designed and tested by W. Cowan and D. Conrad. However, it was never integrated into a complete system prior to this work.

The board consists of three main functional components: 1) ISA (Industry Standard Architecture) bus transceivers; 2) memory; and 3) control components. A functional block diagram of the entire card is shown in Figure 4-9. Communication with the board is accomplished through a standard 16-bit device ISA bus cycle; however, it is configured to utilize only the least-significant eight data bus bits (D0-D7). The signal timings and intricacies of ISA and its 32-bit cousin, Extended ISA (EISA), are well documented [42].

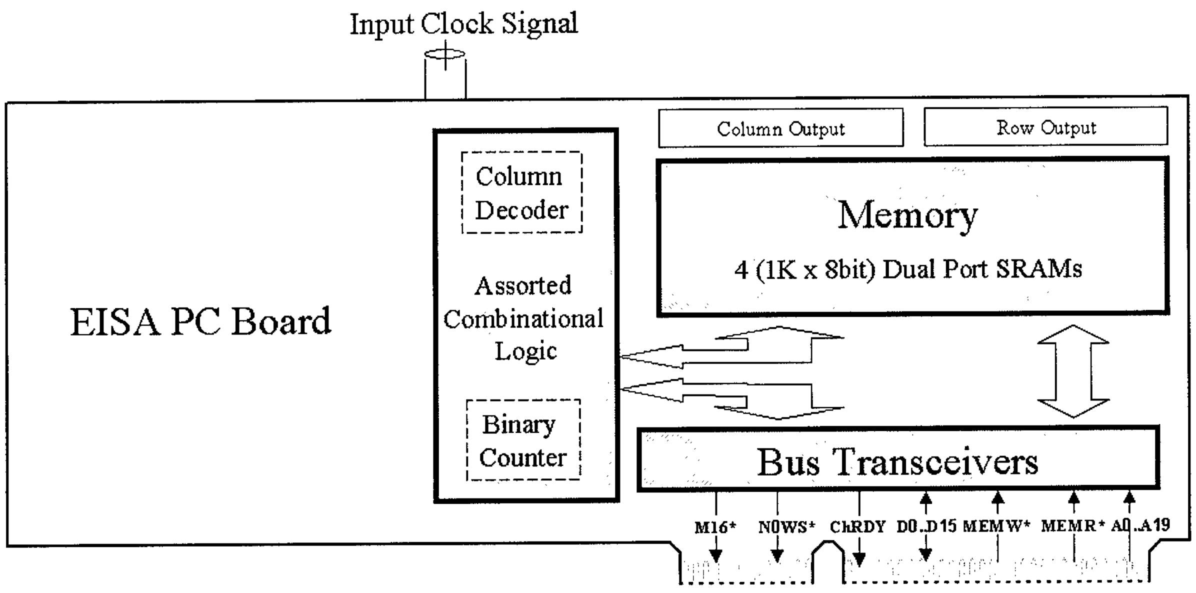


Figure 4-9. High-level schematic of PC Control Board

The ISA bus transceivers continuously update the contents of the memory on the PC board with the contents of the segment of PC system memory that stores the 8-bit word array downloaded by the software. This occurs in a serial fashion at a rate dictated by the bus speed. The primary bus signals utilized in accomplishing the data transfer are listed in Figure 4-10. The actual components and transceiver logic used is detailed in the board's user manual [3].

A0-A19	System Address Bits (used to address memory)
MEMR*	Memory Read
MEMW*	Memory Write
D0..D15	System Data Bits (only D0-D7 are used to transfer memory contents)
ChRDY	Channel Ready (allows slower ISA boards to lengthen memory cycles by inserting wait states)
NOWS*	Zero Wait State (indicates device is capable of operating without wait states)
M16*	Memory Chip Select 16 (indicates device is capable of 16 bit data transfer)

Figure 4-10. Listing of primary ISA bus signals

The memory area of the board is comprised of four (1kB x 8-bit) SRAM (Static Random Access Memory) chips connected in parallel through a conventional MASTER-SLAVE wiring. This configuration creates the necessary (1kB x 32-bit) memory bank needed to store the actual row waveform patterns to be applied to the array. The SRAM chips are dual port which implies the chips have two sets of I/O data lines and address lines. This allows one set of address lines to be wired to a jumper block for "hardwire" testing of specific areas in the memory block. The second set of address lines are connected to the control circuitry. The two I/O data lines are split between input and output from the array. The input lines receive data from the ISA bus transceivers and the output lines are connected to a 34-pin male ribbon cable socket.

The final, and arguably most significant section of the board, is the control circuitry. The foremost component in this section is the 12-bit binary counter chip. This chip, which triggers off of the negative going edge of a separate input clock signal to the board, continuously counts from zero to 1096 in binary across its nine least-significant output pins (the remaining 2 output pins are not utilized). These counter output signals are wired to one of the sets of memory address lines. Thus, when the counter is operating, it effectively cycles through the contents of the array, causing a continuous stream of 32-bit words to be seen across the output lines. These 32-bit words are manifested as a stream of 32 pulse width modulated signals whose voltage varies from 0 to 5 Volts since the SRAM chips are CMOS based. These signals are amplified and applied to the rows of the LAMA device.

The last five most-significant bits of binary output from the counter chip are also wired to a pair of 4 to 16-bit decoder chips. A decoder is a digital circuit device that receives binary input from multiple lines and selects a single output line. These decoder chips (also CMOS) are wired together in a MASTER-SLAVE wiring to act collectively as a 5 to 32-bit decoder. When the counter is operating, the decoder output is an individual voltage pulse 32 input clock cycles wide, with a period of 1024 clock cycles, applied across each of its 32 output lines. The pulse is then inverted, so the inactive state is +5V and the active state is 0V. The 32 output lines are also connected to a separate 34-pin ribbon cable connector. They are of course later amplified and applied to the columns of the LAMA device, acting as the column select signals described in Section 4.2.

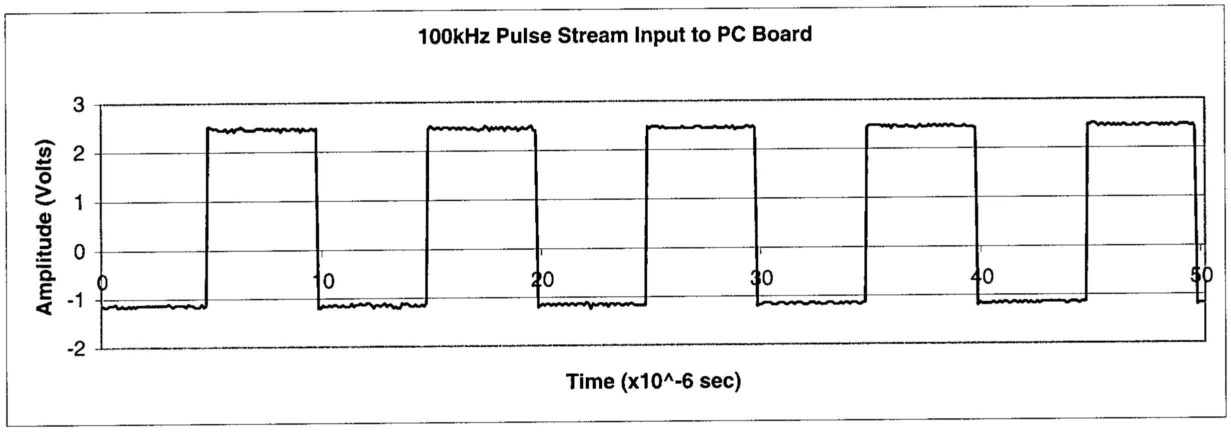
It should become obvious that the counter chip is what keeps the timing of all the signals synchronized with the input clock frequency. Since it is edge triggered, the period of the counter, τ_{counter} , or the time between binary increments, is

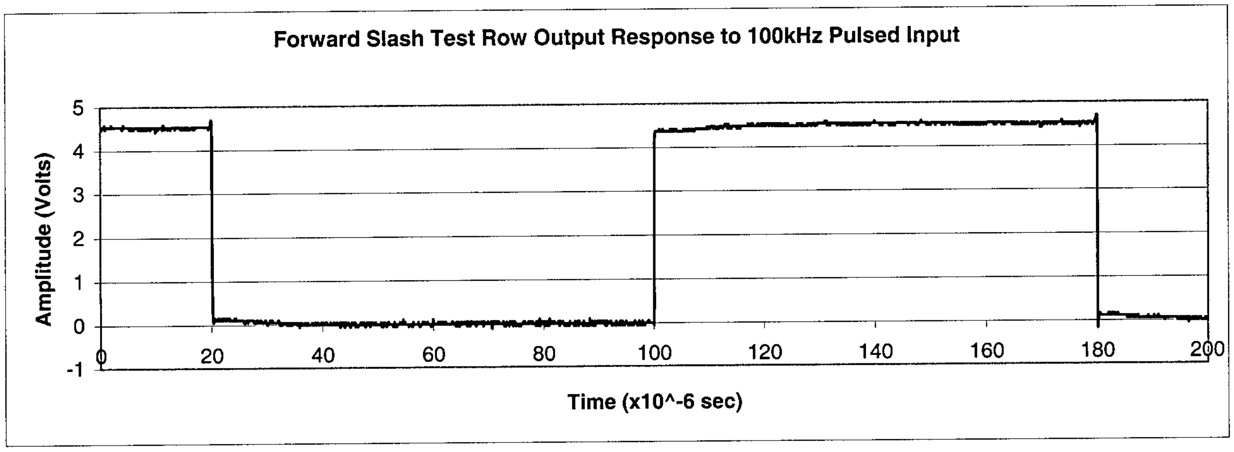
$$\tau_{counter} = \tau_{pulse} + \tau_{separation}\,, \qquad \{4\text{-}9\}$$

where τ_{pulse} is the pulse width, and $\tau_{\text{separation}}$ is the pulse separation. Thus the frequency of the column sweep is equivalent to:

$$f_{col} = \frac{1}{1024 \cdot \tau_{counter}}\,. \qquad \{4\text{-}10\}$$

The remaining two output pins on each of the 34-pin connectors are tied to ground. They are used to keep the amplifier board described in the next section, referenced to the same voltage as the PC board. Figure 4-11 shows actual sample row and column output PWM streams created by the PC board for a given input clock signal.





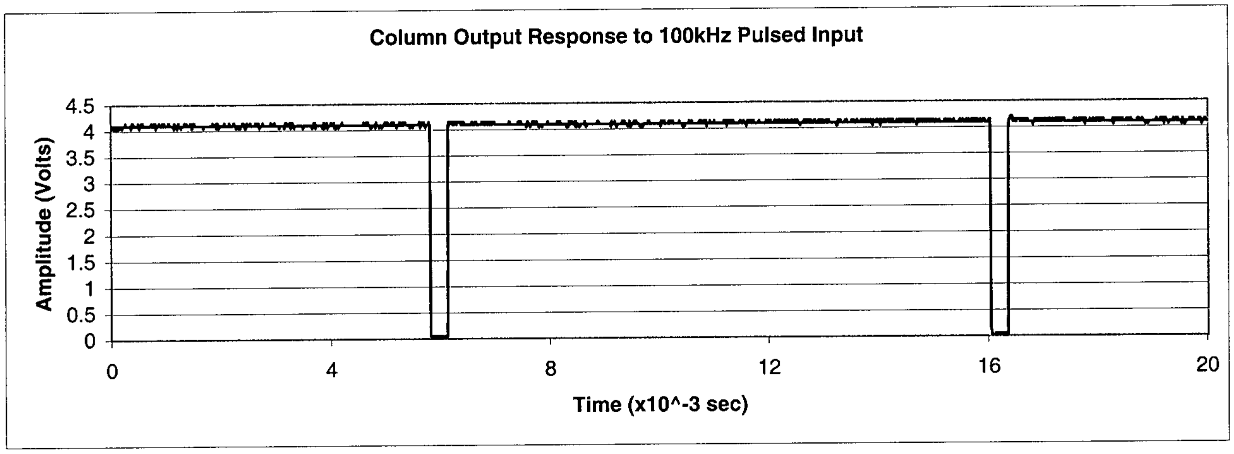


Figure 4-11. Plots of 100kHz clocked input and corresponding row and column output responses for a forward slash test pattern.

4.3.3 Pulse Amplification Board

The two 34-pin output connectors from the control board provide the correct timing and modulation of PWM output. However, the peak pulse voltage from the controller card is only 5V. The pulse amplification board provides the necessary amplification of these signals such that they can be used to achieve significant element deflection in the LAMA device.

The pulse amplification board was designed to maximize flexibility during device testing. Because of this flexibility, the amplifier board could potentially be used to test alternate control schemes or other electrostatically actuated MEMS devices. At the same time, its circuitry was kept simple, with power requirements practical for a laboratory setting. It allows the user to adjust the row signals to virtually any desired level, while still maintaining the ability to operate for both bi-stable modes, as well as above mechanical cutoff. Figure 4-12. shows a photograph of the board with a LAMA device mounted in the middle of it.

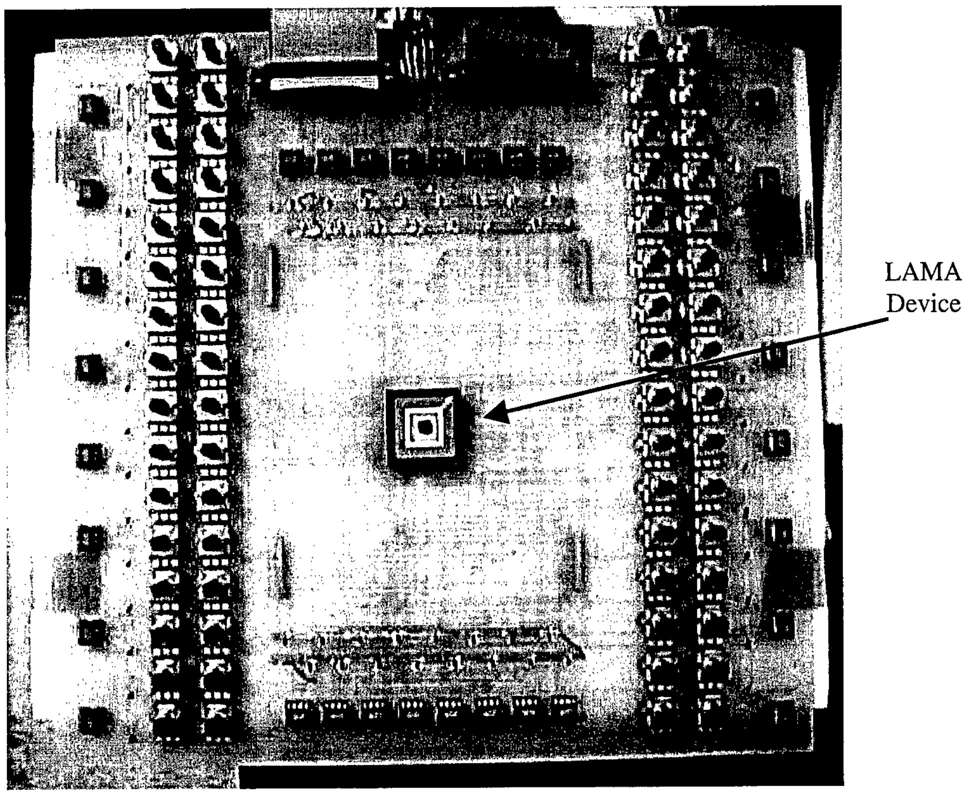


Figure 4-12. Photograph of Pulse Amplification Board with LAMA device

The board schematic and layout was designed using Douglas Electronics CAD/CAM professional design software on a Macintosh computer platform. Images of the final schematic and layout used are included in Appendices C and D respectively. The schematic contains the pin number assignments and signal names to the components and interconnects of the circuit. Upon completion of the schematic, the CAD/CAM software allows for the creation of a netlist file, which is a text file listing all the nodes and signal interconnects within the circuit. The next task is to find, and/or create the correct pad and thru-hole patterns associated with the actual packages of the components chosen for the design. The most important tasks during this step are insuring correct spacings between pads and correct numbering of the pads. The pin numberings on the schematic must also be set to correctly refer to those on the package for each device. The packages must then be graphically positioned by hand, using the layout software, to form the board. Once this is completed, the layout is saved to file and an autorouter program maps and creates the interconnects between devices on the layout based on the schematic netlist. Often in designs with large numbers of interconnects, such as this one, the autorouter must be run a number of times and layout adjustments in component position must be made before it completes close to 100% of the required wiring. Once the autorouter performance is maximized, the remaining interconnects are connected by hand. After visually inspecting/adjusting the layout to maximize the spacing between lines created by the autorouter, the design can be submitted to Douglas Electronics for fabrication. In the case of this design, fabrication took approximately two weeks.

The basic amplification circuit used for each channel in this design is a simple linear pull-down amplifier. A small-signal n-channel enhancement mode MOSFET (metal-oxide-semiconductor field effect transistor), capable of switching voltages up to 60V, was selected for use as a binary switch in the circuit. This implies the MOSFET is either on (closed switch), or off (open switch) based on the voltage applied to its gate. For the rows, a series of two 2.5kΩ variable resistors (1/8 W) were used to act as a variable voltage divider, allowing the user to adjust the inactive state voltage of the output PWM signal. A schematic of this circuit is shown in Figure 4-13. The actual output PWM signals for this circuit will have an active state equivalent to V_{row} and an inactive state determined by the voltage divider equation

$$Vrowout = \left(\frac{Vrow}{R1 + R2} \right) \cdot R2 \qquad \{4\text{-}11\}$$

Since the column signals are fixed, the circuit for these channels uses a single non-variable 2kΩ resistor (1/8 W) as the pull-down resistor (R1 = 2kΩ, R2 = 0Ω). The resistances were chosen to minimize the current drawn from each channel without compromising the frequency response of the circuit when coupled to the output capacitance of the transistors. Since this circuit acts as a digital inverter, to maintain a positive logic system, an inverting high speed MOSFET driver (1.2A) was used to buffer the signal from the PC Board and drive the gate capacitance of the MOSFET device, insuring a clean reproduction of the signal.

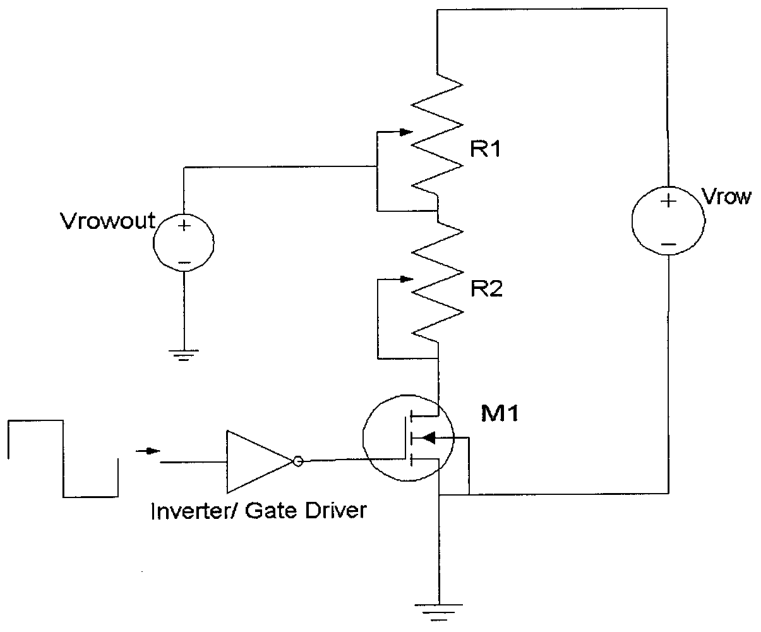


Figure 4-13. Schematic of "pull-down" amplification circuit used for row channels. M1 = small-signal MOSFET, R1 & R2 = variable pull-down resistors

With the exception of the potentiometers, the components have power capacities higher then anything the amp board can produce, to the extent that the amount of voltage the circuit can switch is limited to the current capacity of the power supply. The potentiometers are actually slightly underated, due to packaging and cost considerations; however, the risk was deemed acceptable. It is important to note that the maximum current (and power) is drawn when the entire output signal is set to its inactive state, meaning a completely undeflected array draws the most power during bi-stable operation. Thus the amount of current being drawn when all the row voltages are set low is the first thing the user should check.

The amplification board has five separate contacts for connection to a power supply, once again to maximize flexibility in signal output. These are for the ground (GND), substrate (V_{sub}), active row (V_{row}), active column (V_{col}), and chip voltages (V_{chip}). For proper operation, the LAMA device should be connected to the board when all the signals are disabled. To prevent damaging of the device, a pattern setting all the rows low should be first be downloaded to the PC Board (which is accomplished automatically when LAMACON is launched). Then the voltage to power the chip, substrate voltage (if applicable), column voltage, and finally row voltage should be activated, in that order. This prevents the inadvertent application of random or potentially damaging voltages to the amplification board as well as the LAMA device.

On a final note, the PWM signal output from the amplifier is applied directly to the LAMA device. While the LAMA devices is almost an entirely capacitive load, any resistance it does possess will draw a current from the same supply. This additional current may become especially noticeable if the device is damaged or shorted in any way. Thus, it is also prudent to compare power supply currents with and without the LAMA device, in the case that an unusually large amount of current is being drawn from the supply during operation.

V. Experimental Setup and Procedure

"You got to be careful if you don't know where you are going , because you might not get there."

-Yogi Berra [38]

5.1 Overview

This chapter describes the experimental measures taken to characterize the LAMA devices and test the control system. Section 5.2 describes the experimental setup and Section 5.3 describes the experimental procedure.

5.2 Experimental Setup

Before testing of the LAMA devices could begin, individual test die from each of the foundry processes were released and packaged, so they could be used for experimentation. Section 5.2.1 briefly describes how this was accomplished. Once the devices were packaged they were characterized using a microscopic laser interferometer. The operation of the interferometer operation is described in Section 5.2.2. Upon the completion of the control systems construction, an appropriate experimental test bed was established to accurately characterize its performance when connected to the LAMA device. Section 5.2.3 details the laboratory setup used to in testing the devices.

5.2.1 Releasing and Packaging

Surface micromachined devices are typically shipped from prototype foundries with their sacrificial oxide layers intact to protect them during shipping, and to allow the designer to experiment with various methods of etching. They are also coated with a layer of photoresist to prevent the formation of oxide on the polysilicon surface during shipping. The removal of these layers is collectively termed the "release" process.

Six SUMMiT test die and three MUMPs™ (run #24) test die were released according to the procedure detailed in Appendix F. Packaging of the die first involved adhering the die to the chip carrier, which was accomplished into the release process before the sacrificial layers are etched. Manipulating the die before etching allows one to contact the die surface without damaging the mirror elements. Wire-bonding of the package pins to the connection pads on the die completed the packaging process. Figure 5-1 shows an SEM of a completely released and packaged MUMPs™ LAMA device.

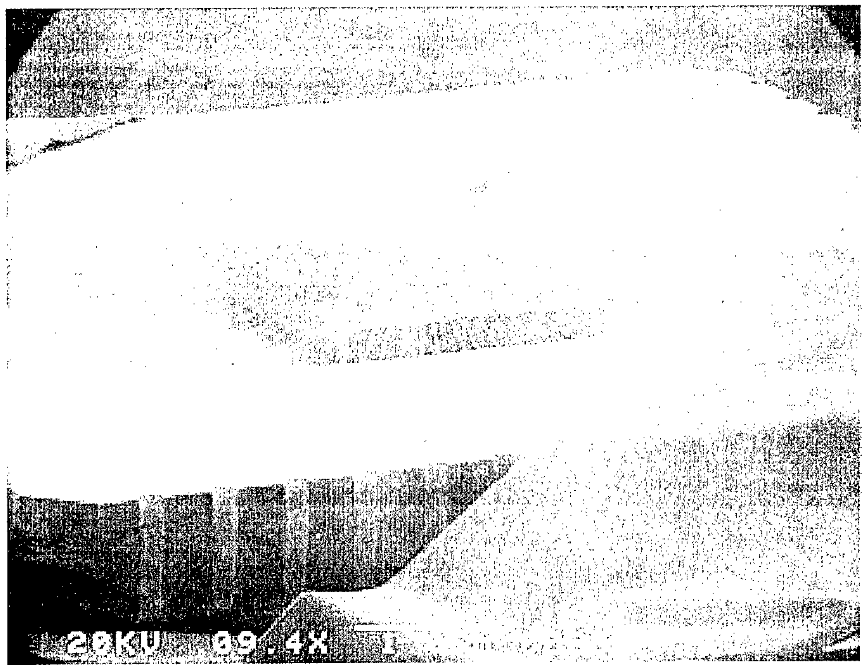


Figure 5-1. SEM Image of MUMPs ™ LAMA device mounted and connected on package

5.2.2 Microscopic Laser Interferometer

The testing and evaluation of the LAMA device was accomplished with the Zygo Maxim 3D microscopic laser interferometer. This instrument is similar in structure to a regular microscope, except it has a built-in HeNe laser ($\lambda = 632.8$ nm) to interferometrically measure phase variations. Specifically, the microFizeau™ x1 and x10 microscope objectives were primarily used to conduct characterization. These microscope objectives are similar to ordinary microscope objectives except that they have a reference surface which is used to from an interference pattern between the field from the reference surface and the field from the test surface [2]. Figure 5-2 shows a ray diagram illustrating the optical paths of these two beams within the microFizeau™ objective.

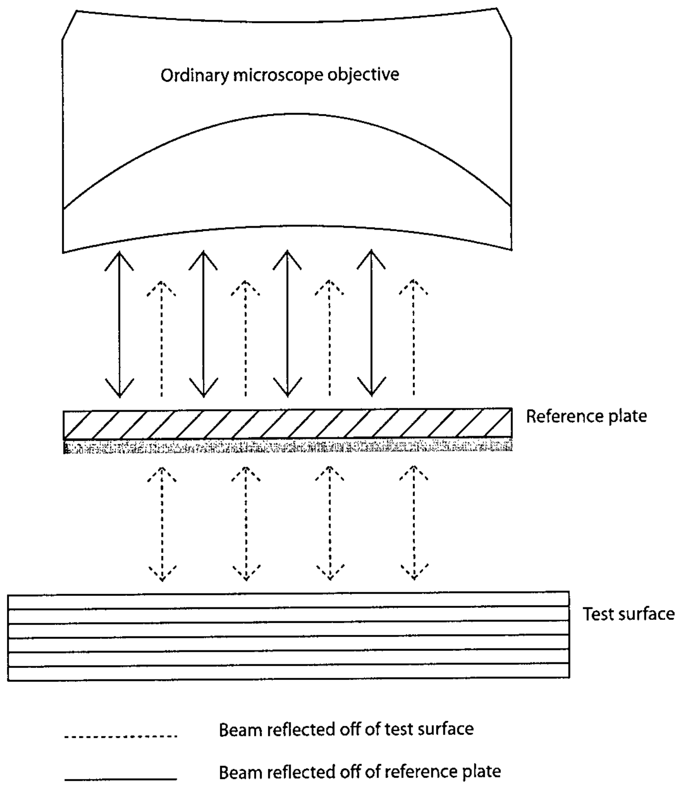

Ordinary microscope objective
Reference plate
Test surface
Beam reflected off of test surface
Beam reflected off of reference plate

Figure 5-2. Ray diagram of microFizeau™ objective, from [2]

The reference surface is a specially coated quarter-wave plate, that can measure test surfaces with reflectivity's as low as 4% (additional optics exist in the interferometer to offset the shift in polarization that occurs upon reflection). Thus, the LAMA devices that were not coated w/ a reflective metal layer were reflective enough to conduct characterization. The fringe spacing is controlled by the tip and tilt of the objective. The range of motion allowed is adequate to create the formation of several fringe lines across a single micromirror element, as well as across the entire array, at low magnification. These fringe lines provide an accurate indicator of curvature and allow the use of the static fringe method described in Section 3.5. Additional information on this interferometer can be found in the Maxim-3D user manual [2].

5.2.3 Control System Setup

The control system components were assembled for testing according to the schematic shown in Figure 5-3. The amplification board received from Douglas Electronics was found to have several layout errors on it. These errors included mislabeled signal lines, incorrectly oriented device pads, and incorrect pad spacings. To correct the misrouted signal lines a separate breadboard (not shown in Figure 5-3) was constructed to provide the correct interface between the PC board and the amplification board. The signal lines are physically interfaced using 34-pin ribbon cables. The GUI is run on a PC with a 133MHz Pentium CPU. The HP3314A Function Generator is used to provide a 3Vp-p, 50% duty cycle square wave to trigger the counter chip on the PC Board. The five power connections for the amplification board are supplied by the HP6624A System DC Power Supply. Signals from the amplifier and PC Board are verified and collected using a Lecroy 7200A Digital Oscilloscope with 7242B dual channel input. The amplification board is mounted onto a Styrofoam base to prevent accidental shorting between nodes. The LAMA device residing in the center of the board, is centered under the microscopic laser interferometer for viewing. Figure 5-4 shows a photograph of the actual lab setup.

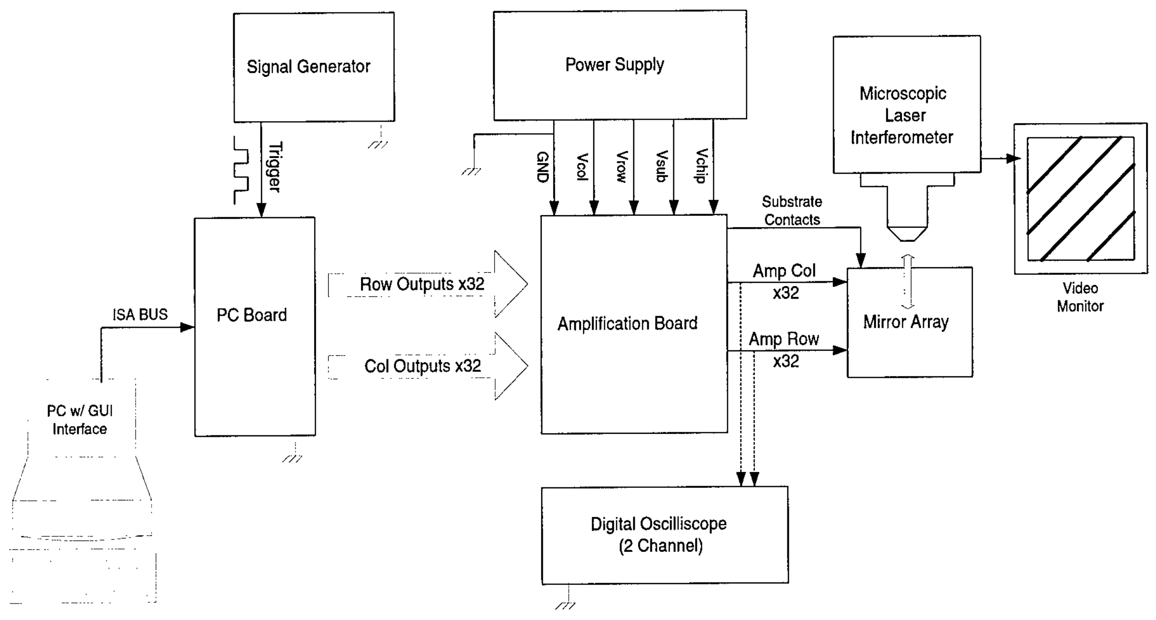


Figure 5-4. Schematic of Control System Experimental Setup

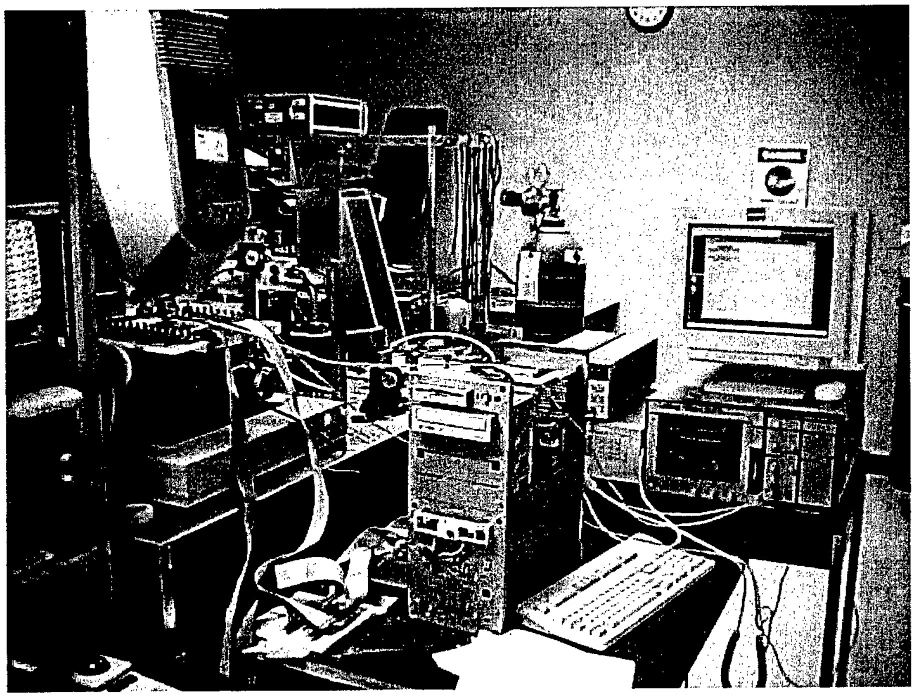


Figure 5-5. Photograph of Control System Experimental Setup

5.3 Experimental Procedure

After packaging the test devices, DC characterization is conducted on each sample via the static fringe method. This is accomplished on a separate test interface that merely wired rows and columns directly to the DC power supply. Estimated spring constants were deduced for various rows, columns, and individual elements across the each array. In addition the snap-down and snap-up voltages were empirically determined by gradually increasing the applied voltage. The resonant and mechanical cutoff frequency of the mirrors was also tested by gradually increasing the frequency of an amplified pulse stream created using a test circuit similar to that in Figure 4-13.

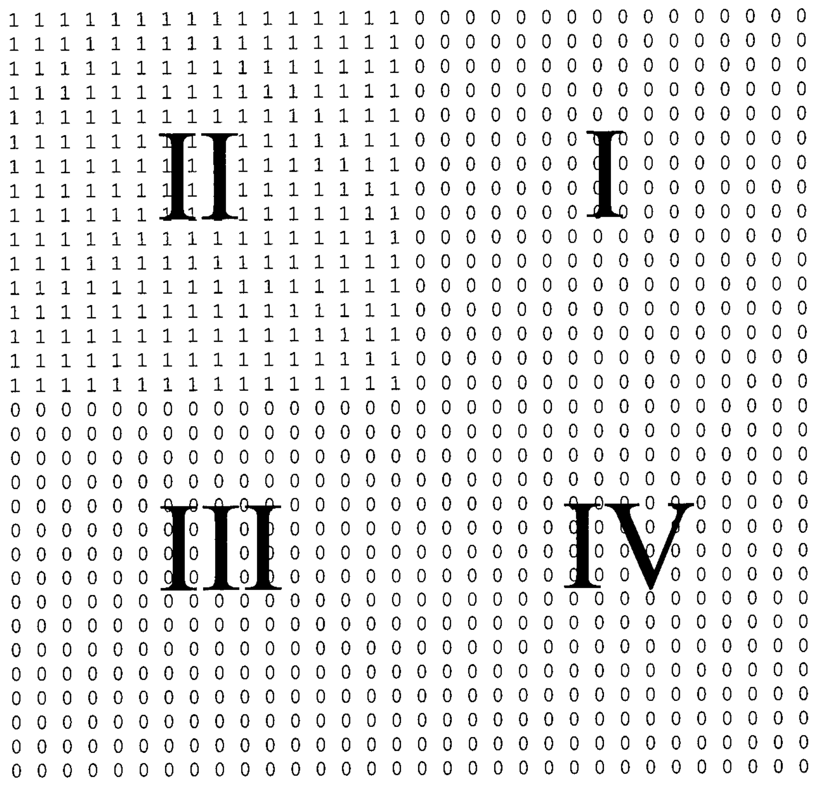
The ultimate focus was to determine the response of the LAMA device under each of the three modes of operation described in Chapter 4 through the use of the control system constructed. Figure 5-6 lists the experimental settings used in testing each of the two bi-stable modes of operation. The multistable ramps were applied for $V_{row} = 20V$ at a frequency of well above the experimentally determined mechanical cutoff (f = 10MHz).

	Conventional	Enhanced
V_{col}	8V	15V
V_{row}	28V	30V
V_{chip}	5V	5V
V_{sub}	8V	0V
R1	1 kΩ	2.5 kΩ
R2	2.5 kΩ	0 kΩ

Figure 5-6. Initial Experimental Settings used for bi-stable testing

The input voltages were chosen for both bi-stable modes to reduce the number of hold states and safely assure the applied differential voltage was greater then the snap-down voltage. The resistance levels were set to minimize the current drawn by each channel during operation as well as to correctly scale the row pulse for the desired mode. These resistances were adjusted by calibrating each potentiometer on the board with a multimeter. Observations made under the microscopic laser interferometer were recorded onto a video cassette for documentation purposes.

If we imagine a set of coordinate axis zeroed at element (16,16) the array can be divided into four quadrants. The primary test pattern used for testing of the bi-stable modes was the "Quadrant II" test pattern, so named because it snaps-down all the elements in the second quadrant; as shown in Figure 5-7. This pattern isolates a large group of elements for snap-down and allows easy comparison between quadrants I and quadrant II and IV which will have different hold states.



```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 5-7. "Quadrant 2" bi-stable test pattern

VI. Test Results and Analysis

"We didn't lose the game; we just ran out of time."

-Vince Lombardi [38]

6.1 Overview

This chapter details the observations and subsequent analysis made throughout the course of experimentation with the LAMA device and its corresponding control system. The application of optical far-field and frequency response models for the LAMA are also presented, based on the device structure and estimated spring constants.

6.2 Micromirror Release Yield

The release process used on the LAMA devices was originally derived from a process previously utilized by AFIT students for MUMPs™ die (see [10]). The first of the three MUMPs™ die released was deliberately overetched for 3.5 minutes to observe the effects of overetching in MUMPs™. The surface on this device was incongruous and pitted as expected. Several elements on it were destroyed. The remaining two die, one with MUMPs™ metal and one without, were released with the recommended 2.5 minute etch time. The sample without metal had 16 faulty mirror elements (98.4% yield) and the one with metal had 22 faulty elements (97.8% yield).

This same process, with a considerably extended etch time, was applied to the SUMMiT arrays. However, the results were inconsistent for the six samples. The SUMMiT die were released in pairs at different times. The first two die were etched for a total time of 50 minutes in HF and the results were reasonable (~98% yield). The second pair of devices when etched for this same time did not appear to completely release. Eight minutes of additional etching was required before uniform release was observed (~98% yield). It was initially believed after releasing this second set of devices, that an accurate etch time had been established for the SUMMiT devices. However, the final pair of devices, when etched for 58 minutes, did not appear to release uniformly. An additional 5 minutes of etching was accomplished on one of these devices in an attempt to complete the release but this resulted in the destruction of mirror elements; the supporting hinges were entirely dissolved (see Figure 6-1). The final two devices were not packaged or included in further experimentation. Although, at the completion of the experiment, it was noticed that the device which wasn't etched for an additional 5 minutes in this final pair actually appeared to have released when reexamined over a month later. This observation suggests that perhaps the problem observed is not in the etching, but rather in the evaporation of the methanol at the end of the release process. It is suspected that additional drying time may be required for the SUMMiT devices. In addition, the number of unreleased elements tends to increase towards the center of the array. This observation is explained partially by noting the geometry of the device. The elements on the edge of the array have both top and side sacrificial oxide area exposed for etching, while the elements on the inside of the array have only the top area exposed. If the etch holes in the SUMMiT devices alone don't allow enough ion exchange during etching, the elements towards the outside should release more rapidly and completely then those on the inside. Since the SUMMiT devices are etched for much longer than the

MUMPs™ devices, the fact that the HF solution is stagnate during etching may be causing an uneven etch rate across the device. This conclusion is strengthened by noting that the first two pairs of test die were physically dipped in and out of the HF solution in an effort to establish a proper etch time, which raises the suspicion that the dipping itself may have enhanced the effect of the etch, resulting in successful release. A potential solution to this problem is to introduce slight agitation while etching in order to insure a constant ion exchange is occuring.

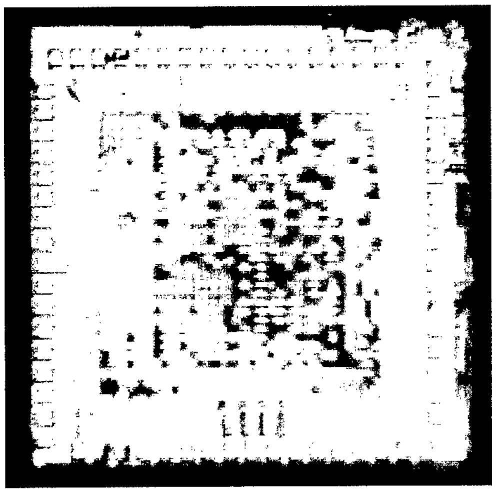


Figure 6-1. Image of poorly released SUMMiT LAMA device.

The details of this release process may at first appear inconsequential to the control of the LAMA device, especially with such high yields, however this is not the case. In fact, this step is potentially the most critical to master if line-addressing is to be realized in micromirror arrays. While 97-98% yield is acceptable for researching individually addressed arrays, it is potentially devastating to a line-addressable array of this design, since the top plates of all the mirror elements are tied together by rows. If an element's top plate breaks off, not only is that element destroyed but control is lost for the remaining elements in that row. In addition, if other defects occur such as a shorting of a top plate to a bottom plate, control of both the column and row is affected. The full ramifications of this will become apparent when the operation of the control system is discussed in Section 6.5.

6.3 Micromirror Characterization

To begin device characterization, static fringe voltages were measured for the rows, columns, and individual elements of the three SUMMiT and two MUMPs™ samples; using the technique described in Section 3.5. The voltage values listed in Figure 6-2 were generated by applying positive voltages to the rows and negative voltages to the columns with the substrate held at ground. Results are consistent between samples, however varied across the micromirror array for the MUMPs™ devices. In the MUMPs™ devices, the static fringe voltage increases for rows/columns farther from the center of the array. This effect is being attributed to uneven etching across the array. The column static fringe voltages were consistently larger then the row voltages for all devices tested. While the overlapping plate area remains unchanged between measurements, the top plates (row plates) are larger and partially overlap the substrate. There is an additional force produced by this overlap and the effective spring constant is substantially less. The individual element measurements are actually dependent on the amount of row to substrate differential voltage, in addition to the differential voltage between plates. Thus for the individual elements the estimated spring constant actually varied slightly depending on how the differential voltage was created across the element. As the ratio of the positive voltage on the row with respect to the negative voltage on the column increased, the static fringe voltage decreased. The value listed in Figure 6.2 is an average value of measurements taken when these voltages are approximately equal in magnitude.

	SUMMiT	MUMPs™ w/o metal
Row	24V	18-21V
Column	26.5V	21-28V
Element	~26V	21-27V

Figure 6-2. DC Static Fringe Voltage measurements taken for SUMMiT and MUMPs ™ LAMA Devices. The SUMMiT static fringe voltage for individual elements was averaged for various applied DC row/column voltages with the substrate grounded.

The static fringe voltage for the MUMPs™ device with metal is not listed because snap-down occurred (at ~14V) before the measurement could be taken. The measured snap down voltages for the remaining devices were all just above the static fringe voltage and are listed in Figure 6-3. The snap-up voltages are not listed because it was found that the snap-up values are strongly dependent on how hard the element is initially snapped down. For SUMMiT elements that are snapped down at voltages just above the listed snap-down voltage, the snap-up voltage is roughly ~9V. The MUMPs™ elements tested under the same conditions didn't snap-up until the voltage was very close to ground. Several elements didn't recover at all though when snapped down even under these conditions. These elements remained permanently snapped down. At voltages well above the snap-down voltage this permanent snap down always occurred. More importantly, these snapped down elements were not at a fixed position. This effect is due to a design flaw in the LAMA device. To prevent stiction from occurring during the release process, dimples were placed under the two corners where the flexure beams join the top mirror plate. When an element is snapped down, these dimples make contact with the substrate, and support the top plate at these two corners. Since the distance between plates is much closer at snap down (~0.95 μm for MUMPs™, and ~0.25 μm for SUMMiT) and any slight instability can cause the top plate to diagonally pivot on these two dimples. The top plate can therefore hold three equilibrium positions at snap-down; two tilt positions and a position where it remains balanced on the two dimples. Each of these three possible states is shown in Figure 6-4. It was also noted that if an element was held at a voltage near the snap down voltage for a significant amount of time (5-15 sec) the element would eventually snap-down towards the substrate. This "long term snap-down" is attributed to capacitive charge build-up occurring between the plates of the element, which is biased in a fixed direction. One way to eliminate this effect is to switch the polarity of voltages applied to the element, thus maintaing the same differential voltage while the built-up charge drains. Countering this effect was not deemed critical to the demonstration attempted in this work, which is why it was not built into the control electronics. It is, however, important in the control of future micromirror arrays in which a fixed pattern needs to be held for a long period of time.

	SUMMiT	MUMPs™ w/o metal
Row	25V	20-24V
Column	27V	24-31V
Element	~26.5V	24-28V

Figure 6-3. DC Snap-down Voltage measurements taken for SUMMiT and MUMPs™ LAMA Devices. The SUMMiT static fringe voltage for individual elements was averaged for various applied DC row/column voltages with the substrate grounded.

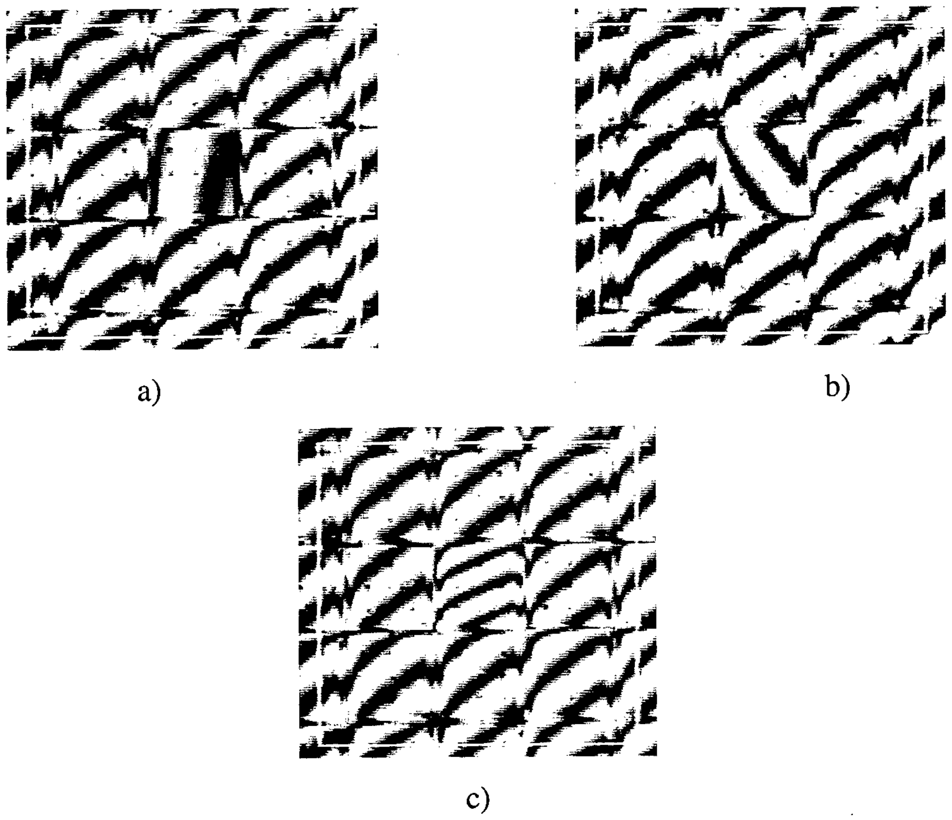


Figure 6-4. Images captured under the microscopic laser interferometer indicating three potential snap-down states of the LAMA element
The rotation of fringe lines indicates additional tilt induced by the element; a) and b) show tilting to the opposite corners, while c) shows some intermediate state.

Operation of the mirror elements above the mechanical cutoff correlated directly with predicted theory. The static fringe rms voltages collected were the same as the static fringe voltages collected under DC conditions. The degree of frequency above the mechanical cutoff did not appear to make a difference in operation.

Surface profiles of the entire array were also viewed under the microscopic laser interferometer. While actual differential heights could not be measured due to lack of an adequate reference plane, the bending of fringe lines is observed across the SUMMiT arrays as shown by Figure 6-5. The fringe bending indicates a slight curvature profile across the entire array. The MUMPs™ arrays also exhibited this property. Curvature was also observed on individual elements for both the SUMMiT and MUMPs™ devices. Figure 6-5 shows a positive curvature profile with ~30 nm center-to-edge difference across an individual SUMMiT element. This figure also shows the difference in actuation observed between rows and columns for a given DC voltage.

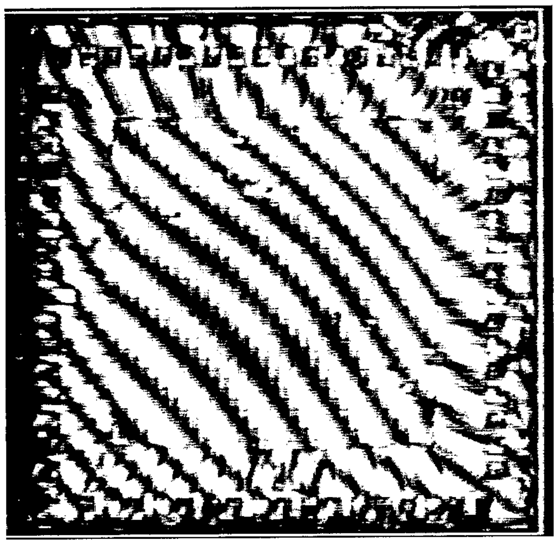


Figure 6-5. Image of SUMMiT LAMA device under microscopic laser interferometer indicating curvature across the array

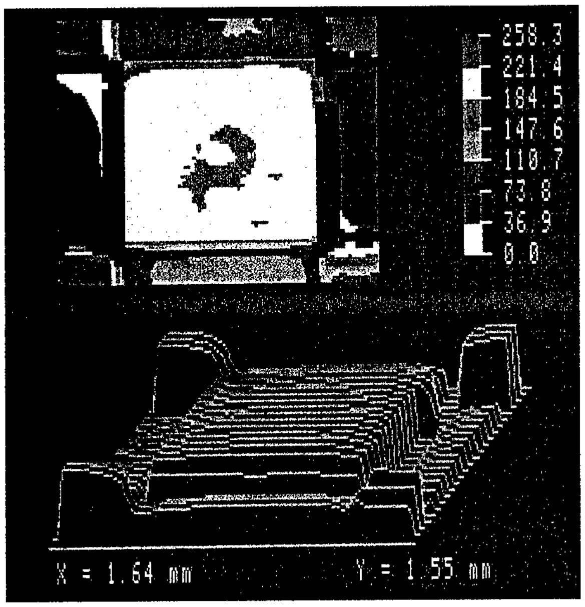


Figure 6-6. Height profile of individual SUMMiT LAMA element with +20V applied to its corresponding row and column
The contour profile shown is not to absolute scale with respect to the substrate, however it does provide an accurate indication of relative position between elements and across an individual element. The scale shown is in units of nm. The difference between column and row deflection for a given voltage is easily seen as well as the curvature profile across the element surface.

6.4 Micromirror Response Modeling

The models discussed in Chapter 3 were implemented and results were generated using an individual SUMMiT element as an example. The intention of this exercise was to demonstrate the extent to which device response can be estimated through the use of simple models. The static DC voltage response model was generated using MATLAB® (see Appendix G). This same code was extended to present the analysis in Section 4.2.3. The estimated spring constant was found to be 22.6 N/m and the resulting projected deflection curve is shown in Figure 6-7.

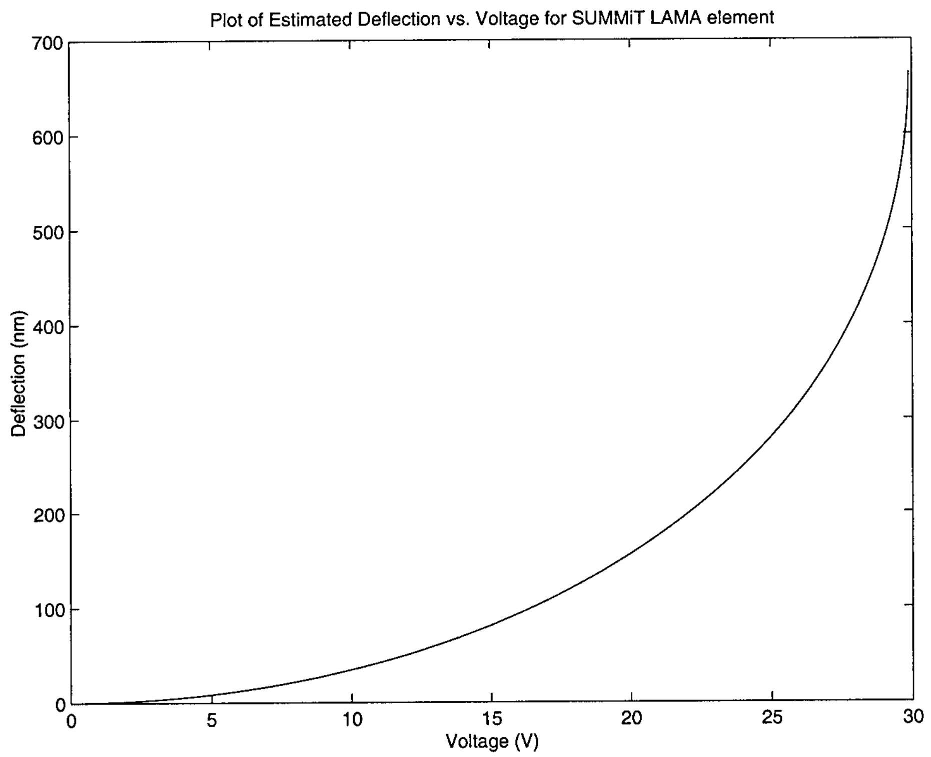


Figure 6-7. Estimated Deflection Response Curve for individual element on SUMMiT device. Curve was generated from a static fringe voltage of 26 Volts.

The corresponding estimated resonant frequency, from Equation {3-8}, is 81kHz. The exact actual resonant frequency and cutoff frequency are difficult to discern under the microscopic laser interferometer, however they are estimated to be roughly 100kHz and 150kHz respectively. This discrepancy between estimations is difficult to explain, since neither estimation is known to be the correct value. Figure 6-8 shows a Bode diagram indicating the mechanical frequency response of a SUMMiT element, based on the mass-spring system presented in Section 3.3.3.

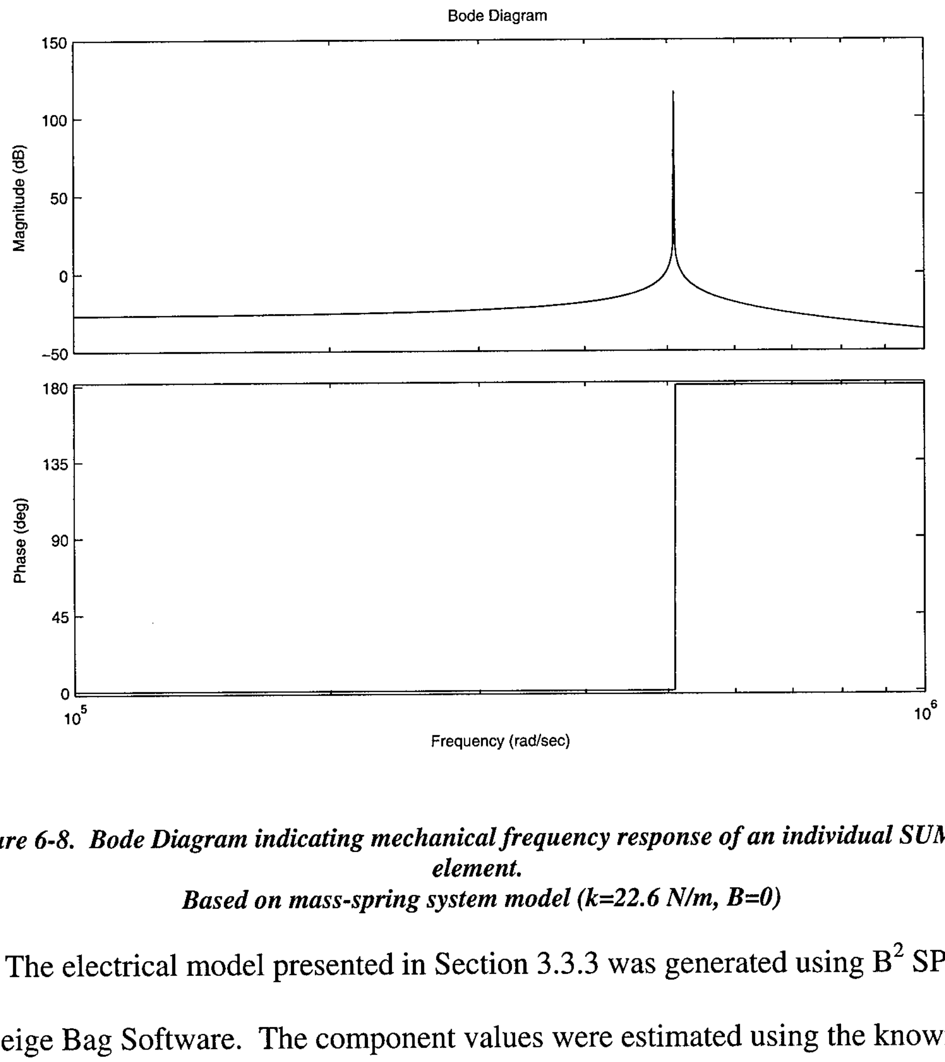


Figure 6-8. Bode Diagram indicating mechanical frequency response of an individual SUMMiT element.
Based on mass-spring system model (k=22.6 N/m, B=0)

The electrical model presented in Section 3.3.3 was generated using B^2 SPICE, from Beige Bag Software. The component values were estimated using the known material parameters and the design dimensions of each circuit element. Figure 6-9 shows that the theoretical response exhibits very little attenuation up until the GHz range.

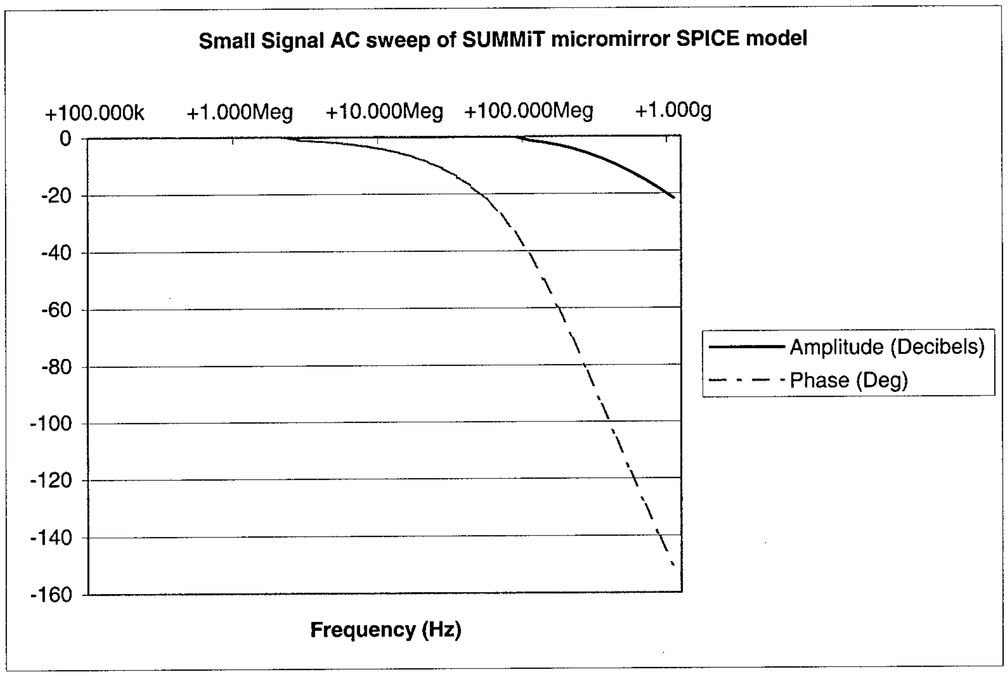


Figure 6-9. Frequency response generated from the individual micromirror (SUMMiT) SPICE model.

An optical model of the reflected far-field was also created in MATLAB® for the LAMA device holding an arbitrary bi-stable pattern (see Appendix H). This code takes full advantage of MATLAB®'s matrix manipulation capabilities, and is subsequently much more efficient compared to previously implemented models [37]. Slight curvature profiles were included in the model to match the observations made during device characterization. Figure 6-10 shows the ideal bi-stable "checkerboard" pattern used to test the optical model (each black/white square = 1 element). Figure 6-11 shows the corresponding theoretical far-field response for a plane-wave reflected off of this pattern.

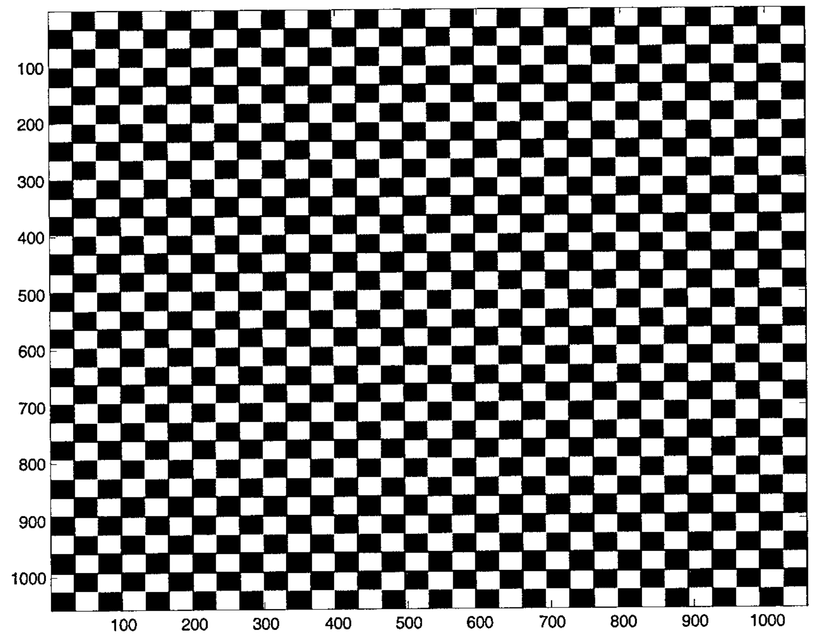
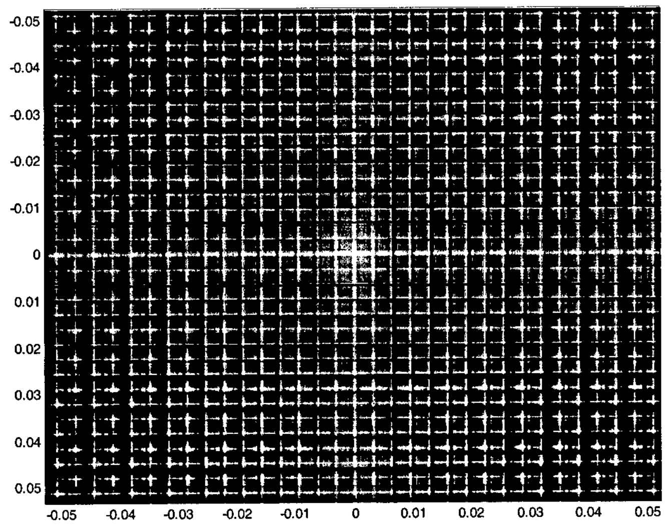
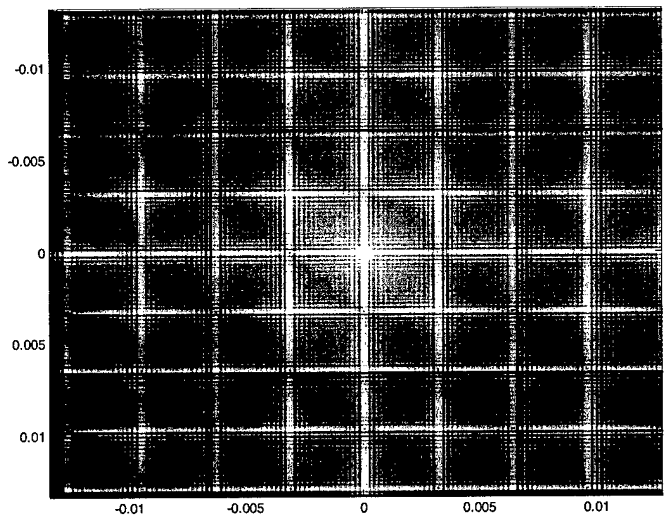


Figure 6-10. Pseudo-image of checkerboard phase map used for optical far-field modeling. The black elements represent the phase accrual associated with snap down, while the white elements are undeflected. The xy axis are scaled in units of pixels.



a)



b)

Figure 6-11. Theoretical far-field intensity distribution for a monochromatic plane wave reflected off of an ideal bi-stable"checkerboard" pattern
The xy axis are scaled for a distance of 1 meter from the array and for a wavelength of 632.8 nm. Image a) shows the far-field pattern b) shows a close-up around the central order. The distribution around the central order indicates the spatial frequency components inherent in the checkerboard aperture profile.

6.5 System Performance

Before bi-stable control of the LAMA device was attempted the control system was first tested separately to insure it functioned properly. Several mistakes were found in the routing of signals, the majority of which trace back to errors made during the PC Board design process. All the known errors were remedied and the entire system was proved functional. This testing was accomplished by observing the arbitrary row and column signals for proper amplitude and phase overlap using a digital oscilliscope. Figure 6-12. shows the overlapping of pulse segments indicating actuation of element (21,4) for the conventional mode of operation.

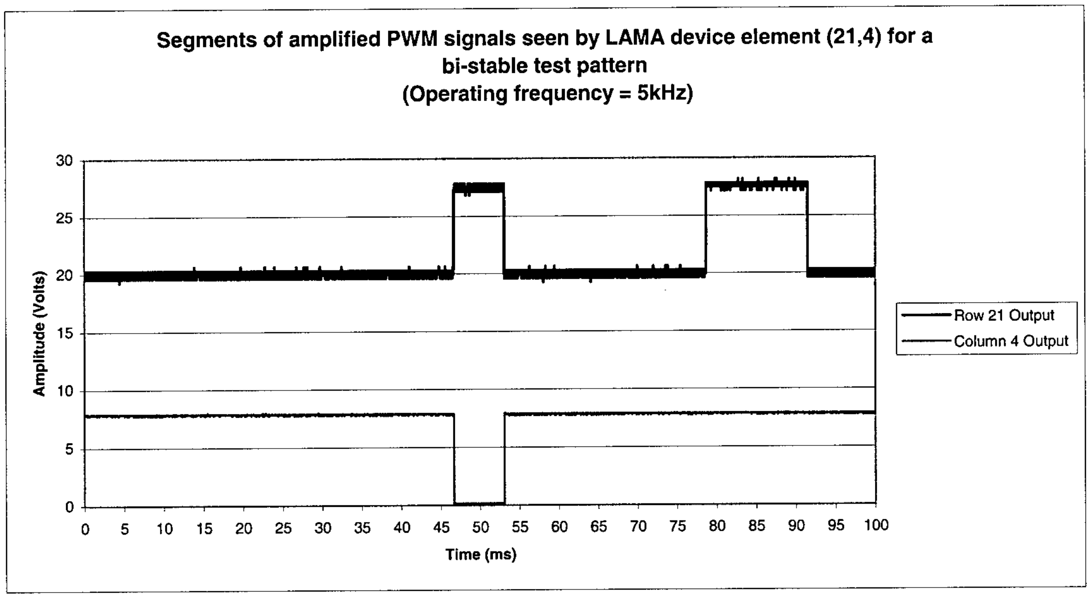


Figure 6-12. Actual row and column signals generated by the control system for operation in the conventional bi-stable mode.
Overlapping of signals indicates that element(21,4) is being directed to the snap-down position.

Testing of the bi-stable modes was predominantly unsuccessful. The first conventional mode did not appear to induce snap-down in any of the elements for the voltages applied. The second method showed a bit more promise. The intended elements did exhibit some motion at the correct time; however they failed to achieve snap-down. The substrate voltage was later changed to that of the columns in the second scenario to better stabilize the array. This helped discern individual deflections in the array, however the elements still did not snap down appropriately. Operation above cutoff was attempted; however, too many defects existed in the arrays to have an even remotely consistent profile across the array. In fact, the problem of imperfect yield is largely responsible for inhibiting the device operation. The number of defects (both shorts and broken elements) is too large to give an accurate appraisal on the validity of the control theory. As a test of how many rows can be successfully actuated at once, simple grating patterns were tested on the array. The application of these pattern revealed that while many rows responded correctly, several of them were not even getting the signal. This effect was observed for all the devices.

The application of the "Quadrant II" bi-stable test pattern, shown in Figure 5-7, using the enhanced bi-stable mode of operation (with Vsub = Vch = 15V and Vrh =30V) showed the most promising results. While a bi-stable pattern was not obtained under these conditions, the greatest actuation of elements was observed in the second quadrant. The timing of the control signals was also verified visually for this pattern with the operating frequency slowed down significantly (1 kHz). When the column scan was seen to reach the 17th column, actuation of the first 16 rows stopped as anticipated. An especially important observation made during the application of this pattern, was that the elements in quadrant I also deflected, to a lesser degree, when the elements in quadrant II deflected. According to the theory in Chapter 4, the elements in quadrant I, should remain undeflected like those in quadrants III and IV. The reason for this discrepancy is that the theory in Chapter 4 assumes equal deflection responses for the rows and columns. Characterization of the deflection response however revealed that this was not the case, due the fact that the plate areas of the top and bottom plates in the LAMA devices were different. This difference in row and column response has a significant effect on bi-stable control.

While the defects in the array represent a huge hurdle in device operation, the conditions applied to the array should have at least produced a few snapped down elements. This leads to the conclusion that the snap-down voltage under operating conditions may be significantly different then that measured under DC conditions. Since the snap-down phenomenon is non-linear and not well understood, it is difficult to determine how or why this is occurring. Higher voltage levels were tried (up to Vrh = 32V, Vch = 17V for the enhanced mode) however, the effect was not changed.

VII. Conclusions and Recommendations

"Each problem that I solved became a rule which served to solve other problems."

-Rene Descartes (1596-1650), "Discours de la Methode" [38]

While bi-stable pattern control was not achieved for this thesis, much was still learned about realizing line-addressing in MEMS micromirror arrays. The characterization accomplished using the microscopic laser interferometer proved to be a very simple method in generating static DC deflection response profiles for the LAMA devices. This information is useful for LAMA device operation above the mechanical cutoff. The inability to demonstrate individual bi-stable element control suggests though that the electromechanical bi-stability properties of the device measured under DC conditions may not be accurate for time-varying signals below the mechanical cutoff. Since similar bi-stable control has been demonstrated in the past on tilt arrays, the evidence suggests that perhaps the piston-actuated structure may be partially responsible for this deviation from the line-addressable theory. Operation above cutoff also remains a viable option and could potentially hold some use. However, the main conclusion to be drawn from this work is that line-addressable control, in all modes of operation, remains highly dependent upon near perfect fabrication and release yield. While the concept still remains a possibility in phase modulating micromirror arrays, better released devices LAMA devices are necessary to accurately determine if bi-stable line-addressing can be realized. Thus optimizing the SUMMiT release process is critical to the success in any future work.

The successful implementation of the control system used to test the LAMA shows that the control of large micromirror arrays, such as the 32x32, is a feasible task if line-addressing is used. Several areas of optimization exist if a more practical control system is ever to be realized. The most immediate of these would be the integration of the PC controller board with the amplifier board. One method worthy of investigation would be the use of high voltage CMOS combinational logic devices that would make the PC Board circuitry robust enough such that an amplifier isn't required. Another potential method would be to eliminate the PC Board and have the software create the PWM signals directly through field-programmable gate arrays (FPGAs) which would then be amplified. A third approach would be to follow Rounsavall's lead and replace the PC Board with a VLSI chip. While these areas will be of eventual interest, the immediate research emphasis should be placed on the optimization of the LAMA device.

Since the control system itself was found to be fully functional for operation, the same system should be used on a tilt array design to first reaffirm the findings originally made by Cunningham et al. In addition a smaller unflawed version of the piston actuated array should be designed and fabricated for future testing. The smaller size will allow more flexibility during testing and less grief in assessing any undiscovered difficulties with the control system.

The redesigned piston actuated LAMA device should include features that will increase the performance and operational yield of the device. The first recommended change is to include dimples at all four corners of the top plate to support the top plate upon snap-down. This will eliminate much of the shorting and permanant snap-down associated with the LAMA device. The second major feature that should be incorporated are having additional sheet contacts under the flexures of the elements. This will allow the area under the flexure beams to be set to the row voltage, which will prevent any spurious electrostatic attraction between the top plate and the substrate. One of the main drawbacks to line-addressable control with the LAMA is that the destruction of a single element's top mirror plate implies the disconnection of all the subsequent elements in that row. The final suggestion for redesign would be to eleiminate this problem by including an additional pair of bus lines for each row, which wuld connect at the flexure vias. Figure 7-1 shows a proposed schematic for a redesigned LAMA element, which includes these modifications.

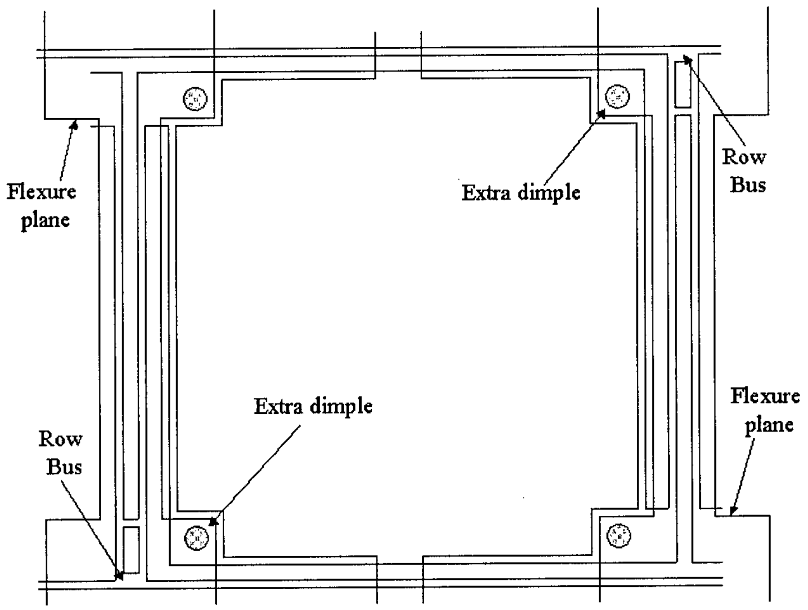


Figure 7-1. Proposed LAMA element redesign

Efforts at optimizing the control system components and redesigning the LAMA device would directly carry over the work presented in this thesis; however, there are also several additional areas that should be explored that would help further line-addressing and micromirror design. One of these is establishing better models for the dynamic response of the micromirror array. This effort includes ascertaining a better understanding of the snap-down instability. Another is devising an alternative mirror device which utilizes additional hold states to achieve some form of multi-stable control. Designs that make a compromise between control lines and individual elements stability should also be given consideration.

APPENDIX A – LAMACON VISUAL BASIC CODE

The ensuing code was written and compiled in Microsoft Visual Basic 5. This code was used to create the Graphical User Interface (GUI) component of the Line-Addressable Micromirror Control (LAMACON) software used in testing the LAMA device. The project file was defined in *ProjectLAMA.vbp* and was used to create a Standard EXE file *ProjectLAMA.exe*. The project consisted of two main form files *frmLAMACON.frm,* which defined the physical control layout, and *frmSplash.frm*, which provided a splash screen with version information. In addition, the project included a separate module file *modLAMAfunctions.bas*, which contained all the subroutines to handle file I/O and calling of the various MS-DOS executable files which handled the physical memory manipulation.

The Microsoft Developers Network (MSDN) help files and Overland's book *Visual Basic 6 in Plain English* [33] were used heavily as references for proper syntax throughout code development.

The following is the complete contents of *frmSplash.frm*:

```
VERSION 5.00
Begin VB.Form frmSplash
   BorderStyle     =   3  'Fixed Dialog
   ClientHeight    =   5970
   ClientLeft      =   255
   ClientTop       =   1410
   ClientWidth     =   7995
   ClipControls    =   0   'False
   ControlBox      =   0   'False
   Icon            =   "frmSplash.frx":0000
   KeyPreview      =   -1  'True
   LinkTopic       =   "Form2"
   MaxButton       =   0   'False
   MinButton       =   0   'False
```

```
ScaleHeight     =   5970
ScaleWidth      =   7995
ShowInTaskbar   =   0   'False
StartUpPosition =   2  'CenterScreen
Begin VB.Frame frameSplash
   Height          =   5715
   Left            =   120
   TabIndex        =   0
   Top             =   120
   Width           =   7785
   Begin VB.Label Label4
      Alignment       =   2  'Center
      Caption         =   "Released 12 November 2000"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   255
      Left            =   3480
      TabIndex        =   8
      Top             =   4320
      Width           =   3495
   End
   Begin VB.Label Label3
      Alignment       =   2  'Center
      Caption         =   $"frmSplash.frx":000C
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   -1  'True
         Strikethrough   =   0   'False
      EndProperty
      Height          =   855
      Left            =   120
      TabIndex        =   7
      Top             =   4800
      Width           =   7215
      WordWrap        =   -1  'True
   End
   Begin VB.Label Label1
      Alignment       =   2  'Center
      Caption         =   "Air Force Institute of Technology"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   255
   Left            =   3480
   TabIndex        =   6
   Top             =   3960
   Width           =   3495
End
Begin VB.Line Line3
   X1              =   3480
   X2              =   7200
   Y1              =   3480
   Y2              =   3480
End
Begin VB.Image Image1
   Height          =   915
   Left            =   3480
   Picture         =   "frmSplash.frx":00CC
   Stretch         =   -1  'True
   Top             =   360
   Width           =   3765
End
Begin VB.Label lblAuthor
   Alignment       =   2  'Center
   Caption         =   "Created by 2d Lt Harris J. Hall USAF"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   12
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   375
   Left            =   3240
   TabIndex        =   5
   Top             =   3600
   Width           =   4215
   WordWrap        =   -1  'True
End
Begin VB.Label lblVersion
   Alignment       =   2  'Center
   Caption         =   "Version 1.1"
   BeginProperty Font
      Name            =   "Impact"
      Size            =   12
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   255
   Left            =   3600
   TabIndex        =   4
   Top             =   3120
   Width           =   3495
End
Begin VB.Line Line2
   X1              =   3480
   X2              =   7200
   Y1              =   3000
```

```
         Y2              =   3000
      End
      Begin VB.Line Line1
         X1              =   2880
         X2              =   2880
         Y1              =   360
         Y2              =   4560
      End
      Begin VB.Label lblTitle
         Alignment       =   2  'Center
         Caption              =     "Line  Addressable  Micromirror  Array  Control
Environment "
         BeginProperty Font
            Name            =   "Impact"
            Size            =   14.25
            Charset         =   0
            Weight          =   400
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   735
         Left            =   3240
         TabIndex        =   3
         Top             =   2160
         Width           =   4335
         WordWrap        =   -1  'True
      End
      Begin VB.Label Label2
         Alignment       =   2  'Center
         Caption         =   "Charlie the Llama"
         BeginProperty Font
            Name            =   "MS Sans Serif"
            Size            =   8.25
            Charset         =   0
            Weight          =   700
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   255
         Left            =   120
         TabIndex        =   2
         Top             =   4560
         Width           =   2295
      End
      Begin VB.Label lblLAMACON
         Alignment       =   2  'Center
         Caption         =   "LAMACON"
         BeginProperty Font
            Name            =   "Times New Roman"
            Size            =   24
            Charset         =   0
            Weight          =   400
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   615
         Left            =   3240
```

```
            TabIndex        =   1
            Top             =   1440
            Width           =   4335
         End
         Begin VB.Image imgCharlie
            Height          =   4155
            Left            =   120
            Picture         =   "frmSplash.frx":1AE0
            Top             =   360
            Width           =   2340
         End
      End
   End
Attribute VB_Name = "frmSplash"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Option Explicit

Private Sub Form_KeyPress(KeyAscii As Integer)
    frmLAMACON.Show
    Unload Me
End Sub

Private Sub frameSplash_Click()
    frmLAMACON.Show
    Unload Me
End Sub

Private Sub lblTitle_Click()
    frmLAMACON.Show
    Unload Me
End Sub
```

The following is primary form manipulation code contained in *frmLAMACON.frm*:

PLEASE NOTE: Only the section of code in *frmLAMACON.frm* that handles actual manipulation of controls and event procedures has been included. The code generated by the VB programming environment that determines control position and asthetics has been left out to conserve space. That section of code looks very similar to that found in *frmSplash.frm* except it's over 1000 times longer. Although the form layout is fundamental to the GUI design, the code defining it is not necessary in understanding the operation of the form.

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'frmLAMACON Load Initialization
'for ProjectLAMA.vbp
'by Harris Hall
'
'- NOTE: ProjectLAMA.exe should reside in C:\LAMA with additional *.exe
'  files it calls
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Private Sub Form_Load()

    'initialize variables
    Dim i As Integer, j As Integer   'increment variables

    'initialize control panel

        'initialize array pattern
        For i = 0 To 31

            cmdRow1(i).Caption = "0"
            cmdRow1(i).Enabled = False
            cmdRow1(i).ToolTipText = "(1," & i + 1 & ")"
            cmdRow1(i).Tag = 0

            cmdRow2(i).Caption = "0"
            cmdRow2(i).Enabled = False
            cmdRow2(i).ToolTipText = "(2," & i + 1 & ")"
            cmdRow2(i).Tag = 0

            cmdRow3(i).Caption = "0"
            cmdRow3(i).Enabled = False
            cmdRow3(i).ToolTipText = "(3," & i + 1 & ")"
            cmdRow3(i).Tag = 0

            cmdRow4(i).Caption = "0"
            cmdRow4(i).Enabled = False
            cmdRow4(i).ToolTipText = "(4," & i + 1 & ")"
            cmdRow4(i).Tag = 0

            cmdRow5(i).Caption = "0"
            cmdRow5(i).Enabled = False
            cmdRow5(i).ToolTipText = "(5," & i + 1 & ")"
            cmdRow5(i).Tag = 0

            cmdRow6(i).Caption = "0"
            cmdRow6(i).Enabled = False
            cmdRow6(i).ToolTipText = "(6," & i + 1 & ")"
            cmdRow6(i).Tag = 0

            cmdRow7(i).Caption = "0"
            cmdRow7(i).Enabled = False
            cmdRow7(i).ToolTipText = "(7," & i + 1 & ")"
            cmdRow7(i).Tag = 0

            cmdRow8(i).Caption = "0"
            cmdRow8(i).Enabled = False
            cmdRow8(i).ToolTipText = "(8," & i + 1 & ")"
```

```
cmdRow8(i).Tag = 0

cmdRow9(i).Caption = "0"
cmdRow9(i).Enabled = False
cmdRow9(i).ToolTipText = "(9," & i + 1 & ")"
cmdRow9(i).Tag = 0

cmdRow10(i).Caption = "0"
cmdRow10(i).Enabled = False
cmdRow10(i).ToolTipText = "(10," & i + 1 & ")"
cmdRow10(i).Tag = 0

cmdRow11(i).Caption = "0"
cmdRow11(i).Enabled = False
cmdRow11(i).ToolTipText = "(11," & i + 1 & ")"
cmdRow11(i).Tag = 0

cmdRow12(i).Caption = "0"
cmdRow12(i).Enabled = False
cmdRow12(i).ToolTipText = "(12," & i + 1 & ")"
cmdRow12(i).Tag = 0

cmdRow13(i).Caption = "0"
cmdRow13(i).Enabled = False
cmdRow13(i).ToolTipText = "(13," & i + 1 & ")"
cmdRow13(i).Tag = 0

cmdRow14(i).Caption = "0"
cmdRow14(i).Enabled = False
cmdRow14(i).ToolTipText = "(14," & i + 1 & ")"
cmdRow14(i).Tag = 0

cmdRow15(i).Caption = "0"
cmdRow15(i).Enabled = False
cmdRow15(i).ToolTipText = "(15," & i + 1 & ")"
cmdRow15(i).Tag = 0

cmdRow16(i).Caption = "0"
cmdRow16(i).Enabled = False
cmdRow16(i).ToolTipText = "(16," & i + 1 & ")"
cmdRow16(i).Tag = 0

cmdRow17(i).Caption = "0"
cmdRow17(i).Enabled = False
cmdRow17(i).ToolTipText = "(17," & i + 1 & ")"
cmdRow17(i).Tag = 0

cmdRow18(i).Caption = "0"
cmdRow18(i).Enabled = False
cmdRow18(i).ToolTipText = "(18," & i + 1 & ")"
cmdRow18(i).Tag = 0

cmdRow19(i).Caption = "0"
cmdRow19(i).Enabled = False
cmdRow19(i).ToolTipText = "(19," & i + 1 & ")"
cmdRow19(i).Tag = 0

cmdRow20(i).Caption = "0"
cmdRow20(i).Enabled = False
cmdRow20(i).ToolTipText = "(20," & i + 1 & ")"
```

```
cmdRow20(i).Tag = 0

cmdRow21(i).Caption = "0"
cmdRow21(i).Enabled = False
cmdRow21(i).ToolTipText = "(21," & i + 1 & ")"
cmdRow21(i).Tag = 0

cmdRow22(i).Caption = "0"
cmdRow22(i).Enabled = False
cmdRow22(i).ToolTipText = "(22," & i + 1 & ")"
cmdRow22(i).Tag = 0

cmdRow23(i).Caption = "0"
cmdRow23(i).Enabled = False
cmdRow23(i).ToolTipText = "(23," & i + 1 & ")"
cmdRow23(i).Tag = 0

cmdRow24(i).Caption = "0"
cmdRow24(i).Enabled = False
cmdRow24(i).ToolTipText = "(24," & i + 1 & ")"
cmdRow24(i).Tag = 0

cmdRow25(i).Caption = "0"
cmdRow25(i).Enabled = False
cmdRow25(i).ToolTipText = "(25," & i + 1 & ")"
cmdRow25(i).Tag = 0

cmdRow26(i).Caption = "0"
cmdRow26(i).Enabled = False
cmdRow26(i).ToolTipText = "(26," & i + 1 & ")"
cmdRow26(i).Tag = 0

cmdRow27(i).Caption = "0"
cmdRow27(i).Enabled = False
cmdRow27(i).ToolTipText = "(27," & i + 1 & ")"
cmdRow27(i).Tag = 0

cmdRow28(i).Caption = "0"
cmdRow28(i).Enabled = False
cmdRow28(i).ToolTipText = "(28," & i + 1 & ")"
cmdRow28(i).Tag = 0

cmdRow29(i).Caption = "0"
cmdRow29(i).Enabled = False
cmdRow29(i).ToolTipText = "(29," & i + 1 & ")"
cmdRow29(i).Tag = 0

cmdRow30(i).Caption = "0"
cmdRow30(i).Enabled = False
cmdRow30(i).ToolTipText = "(30," & i + 1 & ")"
cmdRow30(i).Tag = 0

cmdRow31(i).Caption = "0"
cmdRow31(i).Enabled = False
cmdRow31(i).ToolTipText = "(31," & i + 1 & ")"
cmdRow31(i).Tag = 0

cmdRow32(i).Caption = "0"
cmdRow32(i).Enabled = False
cmdRow32(i).ToolTipText = "(32," & i + 1 & ")"
```

```
            cmdRow32(i).Tag = 0

        Next i

        'Default to Set Rows Low
        'initialize storage array to zero ("Up" Position across array)
        optMode(0).Value = True
        lblProgress.Caption = "Welcome to LAMACON!"

End Sub
Private Sub cmdExit_Click()
    End
End Sub

Private Sub mnuLoad_Click()

    Dim openFileName As String

    'Only reads bi-stable pattern files saved with *.pat extension
    CommonDialog1.Filter = "Binary Pattern Files (*.pat)|*.pat"
    CommonDialog1.ShowOpen

    openFileName = CommonDialog1.FileName 'obtains path and name info

    readPatternFile openFileName 'reads in pattern file

    updateConArray 'updates control array

    lblProgress.Caption = "File loaded:" & openFileName

End Sub

Private Sub mnuSavePattern_Click()

    Dim saveFileName As String

    'Bi-stable Pattern Files are saved with *.pat extension
    CommonDialog1.Filter = "Binary Pattern Files (*.pat)|*.pat"
    CommonDialog1.ShowSave

    saveFileName = CommonDialog1.FileName 'obtains path and name info

    createPatternFile saveFileName

    lblProgress.Caption = "File saved as: " & saveFileName

End Sub

Private Sub mnuViewSplash_Click()
    frmSplash.Show
End Sub

Private Sub optMode_Click(Index As Integer)

    Dim i As Integer

    If optMode(0).Value = True Then
        disableConArray
        lowSetArray  'sets mirrorarray to 0 and calls lowSet.exe
        updateConArray
        cmdUpdate.Enabled = False
        lblProgress.Caption = "Writing 00h to Memory (CF000..CFFFF)"

    ElseIf optMode(1).Value = True Then
        disableConArray
        highSetArray  'sets mirrorarray to 1 and calls highSet.exe
        updateConArray
        cmdUpdate.Enabled = False
        lblProgress.Caption = "Writing FFh to Memory (CF000..CFFFF)"

    ElseIf optMode(2).Value = True Then
        disableConArray
        multiConArray
        testSlash      'calls testSlash.exe
        cmdUpdate.Enabled = False
        lblProgress.Caption = "Writing Forward Test Slash Signal Pattern to
Memory (CF000..CFFFF)"

    ElseIf optMode(3).Value = True Then
        disableConArray
        multiConArray
        upTiltArray          'calls upTilt.exe
        cmdUpdate.Enabled = False
        lblProgress.Caption = "Upwards Voltage Ramp Loaded - Operate ABOVE
cutoff frequency"

    ElseIf optMode(4).Value = True Then
        disableConArray
        multiConArray
        downTiltArray        'calls downTilt.exe
        cmdUpdate.Enabled = False
        lblProgress.Caption = "Downwards Voltage Ramp Loaded - Operate ABOVE
cutoff frequency"

    ElseIf optMode(5).Value = True Then
        enableConArray
        lowSetArray     'sets micromirror array to 0 and calls lowSet.exe
        updateConArray
        cmdUpdate.Enabled = True
        lblProgress.Caption = "Bistable Control Enabled - Operate BELOW cutoff
frequency"

    End If

End Sub

Private Sub cmdUpdate_Click()

    createLiasonFile 'converts and creates liason text file
    memSetArray  ' calls program to interpret liason text file
                 '   and download pattern to physical memory

End Sub

Private Sub cmdRow1_Click(Index As Integer)

   If cmdRow1(Index).Tag = 0 Then
       cmdRow1(Index).Tag = 1
       cmdRow1(Index).Caption = 1
       mirrorArray(0, Index) = 1
   Else
       cmdRow1(Index).Tag = 0
       cmdRow1(Index).Caption = 0
       mirrorArray(0, Index) = 0
   End If

End Sub
Private Sub cmdRow2_Click(Index As Integer)

   If cmdRow2(Index).Tag = 0 Then
       cmdRow2(Index).Tag = 1
       cmdRow2(Index).Caption = 1
       mirrorArray(1, Index) = 1
   Else
       cmdRow2(Index).Tag = 0
       cmdRow2(Index).Caption = 0
       mirrorArray(1, Index) = 0
   End If

End Sub
Private Sub cmdRow3_Click(Index As Integer)

   If cmdRow3(Index).Tag = 0 Then
       cmdRow3(Index).Tag = 1
       cmdRow3(Index).Caption = 1
       mirrorArray(2, Index) = 1
   Else
       cmdRow3(Index).Tag = 0
       cmdRow3(Index).Caption = 0
       mirrorArray(2, Index) = 0
   End If

End Sub
Private Sub cmdRow4_Click(Index As Integer)

   If cmdRow4(Index).Tag = 0 Then
       cmdRow4(Index).Tag = 1
       cmdRow4(Index).Caption = 1
       mirrorArray(3, Index) = 1
   Else
       cmdRow4(Index).Tag = 0
       cmdRow4(Index).Caption = 0
       mirrorArray(3, Index) = 0
   End If

End Sub
Private Sub cmdRow5_Click(Index As Integer)

   If cmdRow5(Index).Tag = 0 Then
       cmdRow5(Index).Tag = 1
       cmdRow5(Index).Caption = 1
       mirrorArray(4, Index) = 1
   Else
       cmdRow5(Index).Tag = 0
       cmdRow5(Index).Caption = 0
       mirrorArray(4, Index) = 0
   End If

End Sub
Private Sub cmdRow6_Click(Index As Integer)
```

```
   If cmdRow6(Index).Tag = 0 Then
       cmdRow6(Index).Tag = 1
       cmdRow6(Index).Caption = 1
       mirrorArray(5, Index) = 1
   Else
       cmdRow6(Index).Tag = 0
       cmdRow6(Index).Caption = 0
       mirrorArray(5, Index) = 0
   End If

End Sub
Private Sub cmdRow7_Click(Index As Integer)

   If cmdRow7(Index).Tag = 0 Then
       cmdRow7(Index).Tag = 1
       cmdRow7(Index).Caption = 1
       mirrorArray(6, Index) = 1
   Else
       cmdRow7(Index).Tag = 0
       cmdRow7(Index).Caption = 0
       mirrorArray(6, Index) = 0
   End If

End Sub
Private Sub cmdRow8_Click(Index As Integer)

   If cmdRow8(Index).Tag = 0 Then
       cmdRow8(Index).Tag = 1
       cmdRow8(Index).Caption = 1
       mirrorArray(7, Index) = 1
   Else
       cmdRow8(Index).Tag = 0
       cmdRow8(Index).Caption = 0
       mirrorArray(7, Index) = 0
   End If

End Sub
Private Sub cmdRow9_Click(Index As Integer)

   If cmdRow9(Index).Tag = 0 Then
       cmdRow9(Index).Tag = 1
       cmdRow9(Index).Caption = 1
       mirrorArray(8, Index) = 1
   Else
       cmdRow9(Index).Tag = 0
       cmdRow9(Index).Caption = 0
       mirrorArray(8, Index) = 0
   End If

End Sub
Private Sub cmdRow10_Click(Index As Integer)

   If cmdRow10(Index).Tag = 0 Then
       cmdRow10(Index).Tag = 1
       cmdRow10(Index).Caption = 1
       mirrorArray(9, Index) = 1
   Else
       cmdRow10(Index).Tag = 0
       cmdRow10(Index).Caption = 0
```

```
        mirrorArray(9, Index) = 0
    End If

End Sub
Private Sub cmdRow11_Click(Index As Integer)

    If cmdRow11(Index).Tag = 0 Then
        cmdRow11(Index).Tag = 1
        cmdRow11(Index).Caption = 1
        mirrorArray(10, Index) = 1
    Else
        cmdRow11(Index).Tag = 0
        cmdRow11(Index).Caption = 0
        mirrorArray(10, Index) = 0
    End If

End Sub
Private Sub cmdRow12_Click(Index As Integer)

    If cmdRow12(Index).Tag = 0 Then
        cmdRow12(Index).Tag = 1
        cmdRow12(Index).Caption = 1
        mirrorArray(11, Index) = 1
    Else
        cmdRow12(Index).Tag = 0
        cmdRow12(Index).Caption = 0
        mirrorArray(11, Index) = 0
    End If

End Sub
Private Sub cmdRow13_Click(Index As Integer)

    If cmdRow13(Index).Tag = 0 Then
        cmdRow13(Index).Tag = 1
        cmdRow13(Index).Caption = 1
        mirrorArray(12, Index) = 1
    Else
        cmdRow13(Index).Tag = 0
        cmdRow13(Index).Caption = 0
        mirrorArray(12, Index) = 0
    End If

End Sub
Private Sub cmdRow14_Click(Index As Integer)

    If cmdRow14(Index).Tag = 0 Then
        cmdRow14(Index).Tag = 1
        cmdRow14(Index).Caption = 1
        mirrorArray(13, Index) = 1
    Else
        cmdRow14(Index).Tag = 0
        cmdRow14(Index).Caption = 0
        mirrorArray(13, Index) = 0
    End If

End Sub

Private Sub cmdRow15_Click(Index As Integer)

    If cmdRow15(Index).Tag = 0 Then
```

```
        cmdRow15(Index).Tag = 1
        cmdRow15(Index).Caption = 1
        mirrorArray(14, Index) = 1
    Else
        cmdRow15(Index).Tag = 0
        cmdRow15(Index).Caption = 0
        mirrorArray(14, Index) = 0
    End If

End Sub
Private Sub cmdRow16_Click(Index As Integer)

    If cmdRow16(Index).Tag = 0 Then
        cmdRow16(Index).Tag = 1
        cmdRow16(Index).Caption = 1
        mirrorArray(15, Index) = 1
    Else
        cmdRow16(Index).Tag = 0
        cmdRow16(Index).Caption = 0
        mirrorArray(15, Index) = 0
    End If

End Sub
Private Sub cmdRow17_Click(Index As Integer)

    If cmdRow17(Index).Tag = 0 Then
        cmdRow17(Index).Tag = 1
        cmdRow17(Index).Caption = 1
        mirrorArray(16, Index) = 1
    Else
        cmdRow17(Index).Tag = 0
        cmdRow17(Index).Caption = 0
        mirrorArray(16, Index) = 0
    End If

End Sub
Private Sub cmdRow18_Click(Index As Integer)

    If cmdRow18(Index).Tag = 0 Then
        cmdRow18(Index).Tag = 1
        cmdRow18(Index).Caption = 1
        mirrorArray(17, Index) = 1
    Else
        cmdRow18(Index).Tag = 0
        cmdRow18(Index).Caption = 0
        mirrorArray(17, Index) = 0
    End If

End Sub
Private Sub cmdRow19_Click(Index As Integer)

    If cmdRow19(Index).Tag = 0 Then
        cmdRow19(Index).Tag = 1
        cmdRow19(Index).Caption = 1
        mirrorArray(18, Index) = 1
    Else
        cmdRow19(Index).Tag = 0
        cmdRow19(Index).Caption = 0
        mirrorArray(18, Index) = 0
    End If
End Sub
Private Sub cmdRow20_Click(Index As Integer)

   If cmdRow20(Index).Tag = 0 Then
       cmdRow20(Index).Tag = 1
       cmdRow20(Index).Caption = 1
       mirrorArray(19, Index) = 1
   Else
       cmdRow20(Index).Tag = 0
       cmdRow20(Index).Caption = 0
       mirrorArray(19, Index) = 0
   End If

End Sub
Private Sub cmdRow21_Click(Index As Integer)

   If cmdRow21(Index).Tag = 0 Then
       cmdRow21(Index).Tag = 1
       cmdRow21(Index).Caption = 1
       mirrorArray(20, Index) = 1
   Else
       cmdRow21(Index).Tag = 0
       cmdRow21(Index).Caption = 0
       mirrorArray(20, Index) = 0
   End If

End Sub
Private Sub cmdRow22_Click(Index As Integer)

   If cmdRow22(Index).Tag = 0 Then
       cmdRow22(Index).Tag = 1
       cmdRow22(Index).Caption = 1
       mirrorArray(21, Index) = 1
   Else
       cmdRow22(Index).Tag = 0
       cmdRow22(Index).Caption = 0
       mirrorArray(21, Index) = 0
   End If

End Sub
Private Sub cmdRow23_Click(Index As Integer)

   If cmdRow23(Index).Tag = 0 Then
       cmdRow23(Index).Tag = 1
       cmdRow23(Index).Caption = 1
       mirrorArray(22, Index) = 1
   Else
       cmdRow23(Index).Tag = 0
       cmdRow23(Index).Caption = 0
       mirrorArray(22, Index) = 0
   End If

End Sub
Private Sub cmdRow24_Click(Index As Integer)

   If cmdRow24(Index).Tag = 0 Then
       cmdRow24(Index).Tag = 1
       cmdRow24(Index).Caption = 1
       mirrorArray(23, Index) = 1
```

```
    Else
        cmdRow24(Index).Tag = 0
        cmdRow24(Index).Caption = 0
        mirrorArray(23, Index) = 0
    End If

End Sub
Private Sub cmdRow25_Click(Index As Integer)

    If cmdRow25(Index).Tag = 0 Then
        cmdRow25(Index).Tag = 1
        cmdRow25(Index).Caption = 1
        mirrorArray(24, Index) = 1
    Else
        cmdRow25(Index).Tag = 0
        cmdRow25(Index).Caption = 0
        mirrorArray(24, Index) = 0
    End If

End Sub
Private Sub cmdRow26_Click(Index As Integer)

    If cmdRow26(Index).Tag = 0 Then
        cmdRow26(Index).Tag = 1
        cmdRow26(Index).Caption = 1
        mirrorArray(25, Index) = 1
    Else
        cmdRow26(Index).Tag = 0
        cmdRow26(Index).Caption = 0
        mirrorArray(25, Index) = 0
    End If

End Sub
Private Sub cmdRow27_Click(Index As Integer)

    If cmdRow27(Index).Tag = 0 Then
        cmdRow27(Index).Tag = 1
        cmdRow27(Index).Caption = 1
        mirrorArray(26, Index) = 1
    Else
        cmdRow27(Index).Tag = 0
        cmdRow27(Index).Caption = 0
        mirrorArray(26, Index) = 0
    End If

End Sub
Private Sub cmdRow28_Click(Index As Integer)

    If cmdRow28(Index).Tag = 0 Then
        cmdRow28(Index).Tag = 1
        cmdRow28(Index).Caption = 1
        mirrorArray(27, Index) = 1
    Else
        cmdRow28(Index).Tag = 0
        cmdRow28(Index).Caption = 0
        mirrorArray(27, Index) = 0
    End If

End Sub
Private Sub cmdRow29_Click(Index As Integer)
```

```
   If cmdRow29(Index).Tag = 0 Then
       cmdRow29(Index).Tag = 1
       cmdRow29(Index).Caption = 1
       mirrorArray(28, Index) = 1
   Else
       cmdRow29(Index).Tag = 0
       cmdRow29(Index).Caption = 0
       mirrorArray(28, Index) = 0
   End If

End Sub
Private Sub cmdRow30_Click(Index As Integer)

   If cmdRow30(Index).Tag = 0 Then
       cmdRow30(Index).Tag = 1
       cmdRow30(Index).Caption = 1
       mirrorArray(29, Index) = 1
   Else
       cmdRow30(Index).Tag = 0
       cmdRow30(Index).Caption = 0
       mirrorArray(29, Index) = 0
   End If

End Sub
Private Sub cmdRow31_Click(Index As Integer)

   If cmdRow31(Index).Tag = 0 Then
       cmdRow31(Index).Tag = 1
       cmdRow31(Index).Caption = 1
       mirrorArray(30, Index) = 1
   Else
       cmdRow31(Index).Tag = 0
       cmdRow31(Index).Caption = 0
       mirrorArray(30, Index) = 0
   End If

End Sub
Private Sub cmdRow32_Click(Index As Integer)

   If cmdRow32(Index).Tag = 0 Then
       cmdRow32(Index).Tag = 1
       cmdRow32(Index).Caption = 1
       mirrorArray(31, Index) = 1
   Else
       cmdRow32(Index).Tag = 0
       cmdRow32(Index).Caption = 0
       mirrorArray(31, Index) = 0
   End If

End Sub


Public Sub enableConArray()

    For i = 0 To 31
        cmdRow1(i).Enabled = True
        cmdRow2(i).Enabled = True
        cmdRow3(i).Enabled = True
        cmdRow4(i).Enabled = True
        cmdRow5(i).Enabled = True
        cmdRow6(i).Enabled = True
        cmdRow7(i).Enabled = True
        cmdRow8(i).Enabled = True
        cmdRow9(i).Enabled = True
        cmdRow10(i).Enabled = True
        cmdRow11(i).Enabled = True
        cmdRow12(i).Enabled = True
        cmdRow13(i).Enabled = True
        cmdRow14(i).Enabled = True
        cmdRow15(i).Enabled = True
        cmdRow16(i).Enabled = True
        cmdRow17(i).Enabled = True
        cmdRow18(i).Enabled = True
        cmdRow19(i).Enabled = True
        cmdRow20(i).Enabled = True
        cmdRow21(i).Enabled = True
        cmdRow22(i).Enabled = True
        cmdRow23(i).Enabled = True
        cmdRow24(i).Enabled = True
        cmdRow25(i).Enabled = True
        cmdRow26(i).Enabled = True
        cmdRow27(i).Enabled = True
        cmdRow28(i).Enabled = True
        cmdRow29(i).Enabled = True
        cmdRow30(i).Enabled = True
        cmdRow31(i).Enabled = True
        cmdRow32(i).Enabled = True
    Next i

End Sub

Public Sub disableConArray()
    Dim i As Integer

    For i = 0 To 31
            cmdRow1(i).Enabled = False
            cmdRow2(i).Enabled = False
            cmdRow3(i).Enabled = False
            cmdRow4(i).Enabled = False
            cmdRow5(i).Enabled = False
            cmdRow6(i).Enabled = False
            cmdRow7(i).Enabled = False
            cmdRow8(i).Enabled = False
            cmdRow9(i).Enabled = False
            cmdRow10(i).Enabled = False
            cmdRow11(i).Enabled = False
            cmdRow12(i).Enabled = False
            cmdRow13(i).Enabled = False
            cmdRow14(i).Enabled = False
            cmdRow15(i).Enabled = False
            cmdRow16(i).Enabled = False
            cmdRow17(i).Enabled = False
            cmdRow18(i).Enabled = False
            cmdRow19(i).Enabled = False
            cmdRow20(i).Enabled = False
            cmdRow21(i).Enabled = False
            cmdRow22(i).Enabled = False
            cmdRow23(i).Enabled = False
            cmdRow24(i).Enabled = False
            cmdRow25(i).Enabled = False
            cmdRow26(i).Enabled = False
            cmdRow27(i).Enabled = False
            cmdRow28(i).Enabled = False
            cmdRow29(i).Enabled = False
            cmdRow30(i).Enabled = False
            cmdRow31(i).Enabled = False
            cmdRow32(i).Enabled = False
        Next i

    End Sub

    Public Sub updateConArray()

        For j = 0 To 31
            cmdRow1(j).Caption = mirrorArray(0, j)
            cmdRow1(j).Tag = mirrorArray(0, j)

            cmdRow2(j).Caption = mirrorArray(1, j)
            cmdRow2(j).Tag = mirrorArray(1, j)

            cmdRow3(j).Caption = mirrorArray(2, j)
            cmdRow3(j).Tag = mirrorArray(2, j)

            cmdRow4(j).Caption = mirrorArray(3, j)
            cmdRow4(j).Tag = mirrorArray(3, j)

            cmdRow5(j).Caption = mirrorArray(4, j)
            cmdRow5(j).Tag = mirrorArray(4, j)

            cmdRow6(j).Caption = mirrorArray(5, j)
            cmdRow6(j).Tag = mirrorArray(5, j)

            cmdRow7(j).Caption = mirrorArray(6, j)
            cmdRow7(j).Tag = mirrorArray(6, j)

            cmdRow8(j).Caption = mirrorArray(7, j)
            cmdRow8(j).Tag = mirrorArray(7, j)

            cmdRow9(j).Caption = mirrorArray(8, j)
            cmdRow9(j).Tag = mirrorArray(8, j)

            cmdRow10(j).Caption = mirrorArray(9, j)
            cmdRow10(j).Tag = mirrorArray(9, j)

            cmdRow11(j).Caption = mirrorArray(10, j)
            cmdRow11(j).Tag = mirrorArray(10, j)

            cmdRow12(j).Caption = mirrorArray(11, j)
            cmdRow12(j).Tag = mirrorArray(11, j)

            cmdRow13(j).Caption = mirrorArray(12, j)
            cmdRow13(j).Tag = mirrorArray(12, j)

            cmdRow14(j).Caption = mirrorArray(13, j)
            cmdRow14(j).Tag = mirrorArray(13, j)

            cmdRow15(j).Caption = mirrorArray(14, j)
            cmdRow15(j).Tag = mirrorArray(14, j)
            cmdRow16(j).Caption = mirrorArray(15, j)
            cmdRow16(j).Tag = mirrorArray(15, j)

            cmdRow17(j).Caption = mirrorArray(16, j)
            cmdRow17(j).Tag = mirrorArray(16, j)

            cmdRow18(j).Caption = mirrorArray(17, j)
            cmdRow18(j).Tag = mirrorArray(17, j)

            cmdRow19(j).Caption = mirrorArray(18, j)
            cmdRow19(j).Tag = mirrorArray(18, j)

            cmdRow20(j).Caption = mirrorArray(19, j)
            cmdRow20(j).Tag = mirrorArray(19, j)

            cmdRow21(j).Caption = mirrorArray(20, j)
            cmdRow21(j).Tag = mirrorArray(20, j)

            cmdRow22(j).Caption = mirrorArray(21, j)
            cmdRow22(j).Tag = mirrorArray(21, j)

            cmdRow23(j).Caption = mirrorArray(22, j)
            cmdRow23(j).Tag = mirrorArray(22, j)

            cmdRow24(j).Caption = mirrorArray(23, j)
            cmdRow24(j).Tag = mirrorArray(23, j)

            cmdRow25(j).Caption = mirrorArray(24, j)
            cmdRow25(j).Tag = mirrorArray(24, j)

            cmdRow26(j).Caption = mirrorArray(25, j)
            cmdRow26(j).Tag = mirrorArray(25, j)

            cmdRow27(j).Caption = mirrorArray(26, j)
            cmdRow27(j).Tag = mirrorArray(26, j)

            cmdRow28(j).Caption = mirrorArray(27, j)
            cmdRow28(j).Tag = mirrorArray(27, j)

            cmdRow29(j).Caption = mirrorArray(28, j)
            cmdRow29(j).Tag = mirrorArray(28, j)

            cmdRow30(j).Caption = mirrorArray(29, j)
            cmdRow30(j).Tag = mirrorArray(29, j)

            cmdRow31(j).Caption = mirrorArray(30, j)
            cmdRow31(j).Tag = mirrorArray(30, j)

            cmdRow32(j).Caption = mirrorArray(31, j)
            cmdRow32(j).Tag = mirrorArray(31, j)

        Next j
End Sub
''''''''''''''''''''''''''''''''''''''''''''''''''''
'multiConArray()
'- lets user know when mirror is in non-bistable mode
''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Sub multiConArray()

        For j = 0 To 31
```

```
            cmdRow1(j).Caption = "X"
            cmdRow2(j).Caption = "X"
            cmdRow3(j).Caption = "X"
            cmdRow4(j).Caption = "X"
            cmdRow5(j).Caption = "X"
            cmdRow6(j).Caption = "X"
            cmdRow7(j).Caption = "X"
            cmdRow8(j).Caption = "X"
            cmdRow9(j).Caption = "X"
            cmdRow10(j).Caption = "X"
            cmdRow11(j).Caption = "X"
            cmdRow12(j).Caption = "X"
            cmdRow13(j).Caption = "X"
            cmdRow14(j).Caption = "X"
            cmdRow15(j).Caption = "X"
            cmdRow16(j).Caption = "X"
            cmdRow17(j).Caption = "X"
            cmdRow18(j).Caption = "X"
            cmdRow19(j).Caption = "X"
            cmdRow20(j).Caption = "X"
            cmdRow21(j).Caption = "X"
            cmdRow22(j).Caption = "X"
            cmdRow23(j).Caption = "X"
            cmdRow24(j).Caption = "X"
            cmdRow25(j).Caption = "X"
            cmdRow26(j).Caption = "X"
            cmdRow27(j).Caption = "X"
            cmdRow28(j).Caption = "X"
            cmdRow29(j).Caption = "X"
            cmdRow30(j).Caption = "X"
            cmdRow31(j).Caption = "X"
            cmdRow32(j).Caption = "X"
        Next j

End Sub
```

The following is the complete contents of *modLAMAfunctions.bas*:

```
Attribute VB_Name = "modLAMAfunctions"
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'modLAMAfunctions.bas
'for ProjectLAMA.vbp
'by Harris Hall
'
'- NOTE: ProjectLAMA.exe should reside in C:\LAMA with additional *.exe files
it calls
'- contains all text file manipulation functions
'- contains mirrorArray() manipulation within createliasonfile() function
'- contains all executable file call functions (*.exe calls)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'Declare Global Variables

Public mirrorArray(31, 31) As Byte 'stores 2D Bi-Stable Array Pattern

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'createPatternFile()
'- creates a text file that contains a map of the 32x32 binary array
'- can only be used by GUI
'- file extension is *.pat
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Public Sub createPatternFile(patternFileName As String)

    'Declare variables
    Dim i, j As Integer
    Dim pfobject, pat

    'Uses FileSystemObject Object
    Set pfobject = CreateObject("Scripting.FileSystemObject")

    If Len(patternFileName) > 1 Then 'checks for Cancel button case

        'NOTE: Overwriting of existing file enabled, ASCII file created
        Set pat = pfobject.CreateTextFile(patternFileName, True)

        'Space delimiting allows pattern to be easily read by MATLAB
        For i = 0 To 31

            For j = 0 To 31

                pat.Write mirrorArray(i, j)
                pat.Write (" ") 'Inserts space between numbers


            Next j

            pat.WriteBlankLines 1

        Next i

        pat.Close

    Else

        MsgBox "Invalid Filename", vbExclamation, "Error in User Input"

    End If

End Sub
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'readPatternFile()
'- reads a text file that contains a map of the 32x32 binary array
'- can only be used by GUI
'- file extension is *.pat
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Public Sub readPatternFile(patternFileName As String)

    'Declare variables
    Dim i, j As Integer
    Dim pfobject, f, ts
    Dim verify As Boolean

    'Uses FileSystemObject Object
    Set pfobject = CreateObject("Scripting.FileSystemObject")
    verify = pfobject.FileExists(patternFileName)
    If verify = True Then 'checks for valid path name
        Set f = pfobject.GetFile(patternFileName)
        Set ts = f.OpenAsTextStream(1, 0)
        'NOTE: ForReading only, ASCII file created

        For i = 0 To 31

            For j = 0 To 31

                mirrorArray(i, j) = ts.Read(1)
                ts.Skip (1)       'Skips over space

            Next j

            ts.SkipLine

        Next i

        ts.Close

    Else

        MsgBox "Invalid Filename", vbExclamation, "Error in User Input"

    End If

End Sub
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'createLiasonFile()
' 1) transfers 32x32 mirrorArray() to inverted 32x32 invMirrorArray()
' 2) transfers 32x32 invMirrorArray() to 1024x32 binary array
' 3) transfers 1024x32 binary array to 1-D char array 4096 long
'       (each element represents 8-bit word) in form of text liasonfile
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Public Sub createLiasonFile()

    'Declare local variables
    Dim a As Integer
    Dim invMirrorArray(31, 31) As Byte 'used to store inverted 32x32 array
    Dim rectArray(1023, 31) As Byte 'used to store 1024x32 binary array
    Dim wordValue As Integer   'used to store 8-bit words
    Dim i, j As Integer
    Dim fso, lfile

    wordValue = 0 'initialize contents of wordValue

    ' Step 1 - Inverts 32x32 mirrorarray() and stores as invMirrorArray

    For i = 0 To 31

        For j = 0 To 31

            invMirrorArray(j, i) = mirrorArray(i, j)

        Next j

    Next i
    i = 0  'reinitialize variables
    j = 0

    ' Step 2 - Transfer 32x32 inverted array to 1024x32 Binary array

    For i = 0 To 31

        For j = 0 To 31

            If invMirrorArray(i, j) = 0 Then

                 'loop assigns 32 rows in 1024x32 for every 1 row in 32x32
                 For a = (i * 32) To ((i * 32) + 31)

                        rectArray(a, j) = 0

                 Next a

            ElseIf invMirrorArray(i, j) = 1 Then

                 'loop assigns 32 rows in 1024x32 for every 1 row in 32x32
                 For a = (i * 32) To ((i * 32) + 31)

                        rectArray(a, j) = 1

                 Next a

            Else
                 'for all other values display error messagebox
                 'message box displays faulty coordinate and value
                 MsgBox "Non-Binary Value Read" & vbCrLf & "(" & i & "," & j &
") =" _
                        & invMirrorArray(i, j), vbCritical, "Error in array
conversion"

                 Exit Sub

            End If  'end conditional structure

        Next j    'end for loop for j

    Next i  'end for loop for i

    i = 0  'reinitialize counters
    j = 0

    ' Step 3 - Transfer 1024x32 array to 1-D integer array in liason text file
        '- Segments contents of rectArray into 8-bit binary segments
        '- Completes binary to base 10 conversion (max value 255)
        '- Downloads integers to liason file (lfile.mem)
        '- Ex. 00001111 binary (0x0F hex) -> 15 decimal
        '- pokeb C++ code will read in as char type

    'operating filename and path set below
    'if file doesn't already exist it is automatically created

    'opens lfile.mem (need to have short MS-DOS filename extension)
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set lfile = fso.CreateTextFile("c:\LAMA\lfile.mem", True)
    For i = 0 To 1023

        For j = 1 To 32

            If (j Mod 8 = 0) Then

                ' add final 8th bits contribution to wordValue
                wordValue = wordValue + rectArray(i, j - 1)

                'write line to liason file
                lfile.WriteLine wordValue

                wordValue = 0  ' clear wordString for new word

            Else
                'converts 8-bit binary data to base 10 value
                wordValue = wordValue + _
                                (rectArray(i, j - 1) * (2 ^ (8 - (j Mod 8))))

            End If

          Next j

    Next i

    lfile.Close   'Closes liasonfile.mem

End Sub
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'lowSetArray()
'- initialize storage array to zero ("Up" Position across array)
'- calls lowSet.exe (quicker than performing conversion
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Public Sub lowSetArray()

    'Declare variables
    Dim i, j As Integer
    Dim RetVal As Variant

    For i = 0 To 31
        For j = 0 To 31
            mirrorArray(i, j) = 0
        Next j
    Next i

    ' Specifying 1 as the second argument opens the application
    '   in normal mode and gives it the focus.

    RetVal = Shell("C:\LAMA\lowSet.EXE", 1)   ' Runs Low Output

End Sub
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'highSetArray()
'- initialize storage array to 1 ("Snap Down" Position across array)
'- calls highSet.exe (quicker then performing conversion)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
Public Sub highSetArray()

    'Declare variables
    Dim i, j As Integer
    Dim RetVal As Variant

    For i = 0 To 31
        For j = 0 To 31
            mirrorArray(i, j) = 1
        Next j
    Next i

    RetVal = Shell("C:\LAMA\highSet.EXE", 1)    ' Runs High Output

End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'testSlash()
'- calls testSlash.exe
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Public Sub testSlash()

    'Declare variables
    Dim RetVal As Variant

    ' Specifying 1 as the second argument in Shell opens the application
    '   in normal mode and gives it the focus.

    RetVal = Shell("C:\LAMA\testSlash.EXE", 1)    ' Runs Forward Test Slash
Output

End Sub
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'memSetArray()
'- calls memSet.exe
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Sub memSetArray()

    'Declare variables
    Dim RetVal As Variant

    ' Specifying 1 as the second argument in Shell opens the application
    '   in normal mode and gives it the focus.

    RetVal = Shell("C:\LAMA\memSet.EXE", 1)    ' Runs Binary pattern interpreter

End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'upTiltArray()
'- calls upTilt.exe
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Sub upTiltArray()

    'Declare variables
    Dim RetVal As Variant

    ' Specifying 1 as the second argument in Shell opens the application
    '   in normal mode and gives it the focus.

    RetVal = Shell("C:\LAMA\upTilt.EXE", 1)    ' Runs Upwards Tilt pattern

End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'downTiltArray()
'- calls downTilt.exe
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Sub downTiltArray()

    'Declare variables
    Dim RetVal As Variant

    ' Specifying 1 as the second argument in Shell opens the application
    '   in normal mode and gives it the focus.

    RetVal = Shell("C:\LAMA\downTilt.EXE", 1)    ' Runs Downwards Tilt pattern

End Sub
```

APPENDIX B – LAMACON C++ CODE

The included code was written and compiled in Borland C++ version 3.1 for MS-DOS. This code was used to create the various MS-DOS executable files that handled the physical memory manipulation for the LAMACON software used in testing the LAMA device. The Borland help files and Deitel & Deitel's book *C++ How to Program* were used as a reference for syntax [16].

There were six separate programs created: *lowSet.exe*, *highSet.exe*, *testSlash.exe*, *memSet.exe*, *downTilt.exe*, and *upTilt.exe*. The contents of their corresponding *.cpp C++ code files are listed below.

```
/*
// lowSet.CPP
// part of:
//     LAMACON - Line-Addressable Micromirror Array CONtrol Software
//     Version 1.1 - by Harris Hall
// SUMMARY - Sets all elements in array to low voltage
//Executable form is lowSet.exe called by LAMACON GUI (ProjectLAMA.exe)
//Can also be called seperately
// NOTE: This code is intended to manipulate physical memory directly.
// It can only be compiled using Windows 3.1 API (16-bit), specifically
// Borland C++ v 3.1 is intended.
// WIN32 API does not permit manipulation of physical memory without
// the aid of a device driver kit (DDK).
*/

#include <dos.h>    // Borland C++ specific header file - defines pokeb function
#include <iostream.h>

int main(void)
{
	cout << "LAMACON Version 1.1 \n";
	cout << "Line Addressable Micromirror Array CONtrol Software \n";
	cout << "\n";
	cout << "Writing 00h to Memory (CF000..CFFFF)\n" ;

	// memory block is 1-D 8-bit char array 4096 elements long
	// 0xCF00 is starting memory block
	for (int i=0;i<4096;i++)
	{
		pokeb(0xCF00, i, 0x00); //fills memory block with zeros
	}
```

```
        return 0; //exit program
}

/*
// highSet.CPP
// part of:
//LAMACON - Line-Addressable Micromirror Array CONtrol Software
//Version 1.1 - by Harris Hall
//SUMMARY - Sets all elements in array to high voltage
//Executable form is highSet.exe called by LAMACON GUI (ProjectLAMA.exe)
// Can also be called seperately
// NOTE: This code is intended to manipulate physical memory directly.
// It can only be compiled using Windows 3.1 API (16-bit), specifically
// Borland C++ v 3.1 is intended.
// WIN32 API does not permit manipulation of physical memory without
// the aid of a device driver kit (DDK).
*/

#include <dos.h>   // Borland C++ specific header file - defines pokeb function
#include <iostream.h>

int main(void)
{
        cout << "LAMACON Version 1.1 \n";
        cout << "Line Addressable Micromirror Array CONtrol Software \n";
        cout << "\n";
        cout << "Writing FFh to Memory (CF000..CFFFF)\n" ;

        // memory block is 1-D 8-bit char array 4096 elements long
        // 0xCF00 is starting memory block
        for (int i=0;i<4096;i++)
        {
                pokeb(0xCF00, i, 0xFF); //fills memory block with ones
        }
        return 0; //exit program
}

/*
// testSlash.CPP
// part of:  LAMACON - Line-Addressable Micromirror Array CONtrol Software
// Version 1.1 - by Harris Hall
// SUMMARY - Set Forward Slash Test Pattern to memory
// Executable form is highSet.exe called by LAMACON GUI (ProjectLAMA.exe)
// Can also be called seperately
// NOTE: This code is intended to manipulate physical memory directly.
// It can only be compiled using Windows 3.1 API (16-bit), specifically
// Borland C++ v 3.1 is intended.
// WIN32 API does not permit manipulation of physical memory without
// the aid of a device driver kit (DDK).
*/

#include  <dos.h>        //  Borland  C++  specific  header  file  -  defines  pokeb
function
#include <iostream.h>

int main(void)
{
        static char fslash [64] = {0xFF,0x00,0xFF,0x00, 0xFE,0x01,0xFE,0x01,
                                   0xFC,0x03,0xFC,0x03, 0xF8,0x07,0xF8,0x07,
                                   0xF0,0x0F,0xF0,0x0F, 0xE0,0x1F,0xE0,0x1F,
                              0xC0,0x3F,0xC0,0x3F, 0x80,0x7F,0x80,0x7F,
                              0x00,0xFF,0x00,0xFF, 0x01,0xFE,0x01,0xFE,
                              0x03,0xFC,0x03,0xFC, 0x07,0xF8,0x07,0xF8,
                              0x0F,0xF0,0x0F,0xF0, 0x1F,0xE0,0x1F,0xE0,
                              0x3F,0xC0,0x3F,0xC0, 0x7F,0x80,0x7F,0x80};
        cout << "LAMACON Version 1.1 \n";
        cout << "Line Addressable Micromirror Array CONtrol Software \n";
        cout << "\n";
        cout << "Writing a Fwd-Slash Diagonal Pattern to Memory (CF000..CFFFF)
\n";

        for (int i=0;i<64;i++) { //# of times to repeat fslash pattern
                for (int j=0;j<64;j++) { //# of bytes in fslash pattern
                        pokeb(0xCF00, (64*i)+j, fslash[j]);
                        }
                }
        return 0;
}

/*
// memSet.CPP
// part of:
//LAMACON - Line-Addressable Micromirror Array CONtrol Software
//Version 1.1 - by Harris Hall
// SUMMARY - Reads in lfile.mem and outputs contents to physical memory block
// lfile.mem needs to be in C:\LAMA directory
// Executable form is memSet.exe called by LAMACON GUI (ProjectLAMA.exe)
//  Can  also  be  called  seperately  however  requires  previously  created
liasonfile.mem
// NOTE: This code is intended to manipulate physical memory directly.
// It can only be compiled using 16-bit DOS API, specifically
// Borland C++ v 3.1 for MS-DOS is intended.
// WIN32 API does not permit manipulation of physical memory without
// the aid of a device driver kit (DDK).
*/

#include <dos.h>   // Borland C++ specific header file - defines pokeb function
#include  <stdlib.h>   //  includes  exit  and  atoi()  function  -  string  to  int
converter
#include <fstream.h> // includes iostream.h and ifstream() functions
#include <string.h> // allows string manipulation functions
#include <iomanip.h> // allows setw() function

int main(void)
{

        int inumber;     // stores 8-bit int number
        char segment[4];  // stores 8-bit string segment

        int i = 0; //initialize counter variable

        ifstream inPatternFile; // creates ifstream object

        cout << "LAMACON Version 1.1 \n";
        cout << "Line Addressable Micromirror Array CONtrol Software \n";
        cout << "\n";
        cout << "Writing Bi-Stable Signal Pattern to Memory \n";

        //ifstream constructor opens the file
        inPatternFile.open( "lfile.mem", ios::in ); //opens liasonfile.mem for
input

        if ( !inPatternFile )
        {
                cerr << "File could not be opened" << endl;
                exit ( 1 );
        }

        // memory block is 1-D 8-bit char array 4096 elements long
        // 0xCF00 is starting memory block
        // sets maximum number of digits to 4
        while (inPatternFile >> setw( 4 ) >> segment)
        {

                  inumber = atoi(segment);  // converts string to integer
                  cout << i << " " << inumber << endl;
                  pokeb(0xCF00, i , inumber);  // downloads to memory
                  i = i+1;    //increments to next address

        }

        inPatternFile.close(); //closes ifstream object

        return 0; //exit program
}

/*
// downTilt.CPP
// part of:  LAMACON - Line-Addressable Micromirror Array CONtrol Software
//          Version 1.1 - by Harris Hall
// SUMMARY - Sets PWM Vertical Downward Voltage Ramp to memory
//          For operation ABOVE cutoff
//          Vrms voltage levels to be interpreted
//          Executable form is downTilt.exe called by LAMACON GUI
(ProjectLAMA.exe)
//          Can also be called seperately
// NOTE: This code is intended to manipulate physical memory directly.
//      It can only be compiled using Windows 3.1 API (16-bit), specifically
//      Borland C++ v 3.1 is intended.
//      WIN32 API does not permit manipulation of physical memory without
//      the aid of a device driver kit (DDK).
*/

#include <dos.h>       // Borland C++ specific header file - defines pokeb
function
#include <iostream.h>

int main(void)
{
        // Define PWM signals pattern
        static char segmentblock0  [4] = {0x7F, 0xFF, 0xFF, 0xFF};
        static char segmentblock1  [4] = {0x3F, 0xFF, 0xFF, 0xFF};
        static char segmentblock2  [4] = {0x1F, 0xFF, 0xFF, 0xFF};
        static char segmentblock3  [4] = {0x0F, 0xFF, 0xFF, 0xFF};
        static char segmentblock4  [4] = {0x07, 0xFF, 0xFF, 0xFF};
        static char segmentblock5  [4] = {0x03, 0xFF, 0xFF, 0xFF};
        static char segmentblock6  [4] = {0x01, 0xFF, 0xFF, 0xFF};
        static char segmentblock7  [4] = {0x00, 0xFF, 0xFF, 0xFF};
        static char segmentblock8  [4] = {0x00, 0x7F, 0xFF, 0xFF};
static char segmentblock9  [4] = {0x00, 0x3F, 0xFF, 0xFF};
static char segmentblock10 [4] = {0x00, 0x1F, 0xFF, 0xFF};
static char segmentblock11 [4] = {0x00, 0x0F, 0xFF, 0xFF};
static char segmentblock12 [4] = {0x00, 0x07, 0xFF, 0xFF};
static char segmentblock13 [4] = {0x00, 0x03, 0xFF, 0xFF};
static char segmentblock14 [4] = {0x00, 0x01, 0xFF, 0xFF};
static char segmentblock15 [4] = {0x00, 0x00, 0xFF, 0xFF};
static char segmentblock16 [4] = {0x00, 0x00, 0x7F, 0xFF};
static char segmentblock17 [4] = {0x00, 0x00, 0x3F, 0xFF};
static char segmentblock18 [4] = {0x00, 0x00, 0x1F, 0xFF};
static char segmentblock19 [4] = {0x00, 0x00, 0x0F, 0xFF};
static char segmentblock20 [4] = {0x00, 0x00, 0x07, 0xFF};
static char segmentblock21 [4] = {0x00, 0x00, 0x03, 0xFF};
static char segmentblock22 [4] = {0x00, 0x00, 0x01, 0xFF};
static char segmentblock23 [4] = {0x00, 0x00, 0x00, 0xFF};
static char segmentblock24 [4] = {0x00, 0x00, 0x00, 0x7F};
static char segmentblock25 [4] = {0x00, 0x00, 0x00, 0x3F};
static char segmentblock26 [4] = {0x00, 0x00, 0x00, 0x1F};
static char segmentblock27 [4] = {0x00, 0x00, 0x00, 0x0F};
static char segmentblock28 [4] = {0x00, 0x00, 0x00, 0x07};
static char segmentblock29 [4] = {0x00, 0x00, 0x00, 0x03};
static char segmentblock30 [4] = {0x00, 0x00, 0x00, 0x01};
static char segmentblock31 [4] = {0x00, 0x00, 0x00, 0x00};

int counter = 0; //initialize global counter variable
int i,j = 0; // initialize loop counter variable

cout << "LAMACON Version 1.1 \n";
cout << "Line Addressable Micromirror Array CONtrol Software \n";
cout << "\n";
cout  <<  "Writing  Upward  Voltage  Ramp  /  upTilt  Pattern  to  (CF00..CFFF)
\n";
cout << "For operation ABOVE cutoff frequency!  \n";

// Repeats diagonal quanta pattern
// Signals constructed to maximize central frequency component
for (i=0;i<32;i++)
{ //# of times to repeat each segment block

      //segmentblock0 download to memory
      for (j=0;j<4;j++)
      { //# of 8-bit parts in each 32-bit segment block
            pokeb(0xCF00, counter, segmentblock0[j]);
            counter++; // increment global counter
      }

      //segmentblock1 download to memory
      for (j=0;j<4;j++)
      { //# of 8-bit parts in each 32-bit segment block
            pokeb(0xCF00, counter, segmentblock1[j]);
            counter++; // increment global counter
      }


      //segmentblock1 download to memory
      for (j=0;j<4;j++)
      { //# of 8-bit parts in each 32-bit segment block
            pokeb(0xCF00, counter, segmentblock1[j]);
            counter++; // increment global counter
      }
```

```
//segmentblock2 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
       pokeb(0xCF00, counter, segmentblock2[j]);
       counter++; // increment global counter
}

//segmentblock3 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
       pokeb(0xCF00, counter, segmentblock3[j]);
       counter++; // increment global counter
}

//segmentblock4 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
       pokeb(0xCF00, counter, segmentblock4[j]);
       counter++; // increment global counter
}

//segmentblock5 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
       pokeb(0xCF00, counter, segmentblock5[j]);
       counter++; // increment global counter
}

//segmentblock6 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
       pokeb(0xCF00, counter, segmentblock6[j]);
       counter++; // increment global counter
}

//segmentblock7 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
       pokeb(0xCF00, counter, segmentblock7[j]);
       counter++; // increment global counter
}

//segmentblock8 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
       pokeb(0xCF00, counter, segmentblock8[j]);
       counter++; // increment global counter
}

//segmentblock9 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
       pokeb(0xCF00, counter, segmentblock9[j]);
       counter++; // increment global counter
}

//segmentblock10 download to memory
for (j=0;j<4;j++)
```

```
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock10[j]);
      counter++; // increment global counter
}

//segmentblock11 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock11[j]);
      counter++; // increment global counter
}

//segmentblock12 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock12[j]);
      counter++; // increment global counter
}

//segmentblock13 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock13[j]);
      counter++; // increment global counter
}

//segmentblock14 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock14[j]);
      counter++; // increment global counter
}

//segmentblock15 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock15[j]);
      counter++; // increment global counter
}

//segmentblock16 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock16[j]);
      counter++; // increment global counter
}

//segmentblock17 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock17[j]);
      counter++; // increment global counter
}

//segmentblock18 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock18[j]);
      counter++; // increment global counter
}
```

```
//segmentblock19 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock19[j]);
      counter++; // increment global counter
}

//segmentblock20 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock20[j]);
      counter++; // increment global counter
}

//segmentblock21 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock21[j]);
      counter++; // increment global counter
}

//segmentblock22 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock22[j]);
      counter++; // increment global counter
}

//segmentblock23 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock23[j]);
      counter++; // increment global counter
}

//segmentblock24 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock24[j]);
      counter++; // increment global counter
}

//segmentblock25 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock25[j]);
      counter++; // increment global counter
}

//segmentblock26 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock26[j]);
      counter++; // increment global counter
}

//segmentblock27 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
```

```
                pokeb(0xCF00, counter, segmentblock27[j]);
                counter++; // increment global counter
        }

        //segmentblock28 download to memory
        for (j=0;j<4;j++)
        { //# of 8-bit parts in each 32-bit segment block
                pokeb(0xCF00, counter, segmentblock28[j]);
                counter++; // increment global counter
        }

        //segmentblock29 download to memory
        for (j=0;j<4;j++)
        { //# of 8-bit parts in each 32-bit segment block
                pokeb(0xCF00, counter, segmentblock29[j]);
                counter++; // increment global counter
        }

        //segmentblock30 download to memory
        for (j=0;j<4;j++)
        { //# of 8-bit parts in each 32-bit segment block
                pokeb(0xCF00, counter, segmentblock30[j]);
                counter++; // increment global counter
        }

        //segmentblock31 download to memory
        for (j=0;j<4;j++)
        { //# of 8-bit parts in each 32-bit segment block
                pokeb(0xCF00, counter, segmentblock31[j]);
                counter++; // increment global counter
        }

        //segmentblock32 download to memory
        for (j=0;j<4;j++)
        { //# of 8-bit parts in each 32-bit segment block
                pokeb(0xCF00, counter, segmentblock32[j]);
                counter++; // increment global counter
        }

    }

    cout << counter << endl;

    return 0; // exit program

}// end of downTilt.cpp

/*
// upTilt.CPP
// part of:  LAMACON - Line-Addressable Micromirror Array CONtrol Software
//          Version 1.1 - by Harris Hall
// SUMMARY - Sets PWM Vertical Upward Voltage Ramp to memory
//          For operation ABOVE cutoff
//          Vrms voltage levels to be interpreted
//          Executable   form   is   upTilt.exe   called   by   LAMACON   GUI
(ProjectLAMA.exe)
//          Can also be called seperately
// NOTE: This code is intended to manipulate physical memory directly.
//      It can only be compiled using Windows 3.1 API (16-bit), specifically
//      Borland C++ v 3.1 is intended.
//      WIN32 API does not permit manipulation of physical memory without
//      the aid of a device driver kit (DDK).
*/

#include <dos.h>        // Borland C++ specific header file - defines pokeb
function
#include <iostream.h>

int main(void)
{
        // Define 32 PWM signal quanta - These will construct pulses
        static char segmentblock0  [4] = {0xFF, 0xFF, 0xFF,  0xFE};
        static char segmentblock1  [4] = {0xFF, 0xFF, 0xFF,  0xFC};
        static char segmentblock2  [4] = {0xFF, 0xFF, 0xFF,  0xF8};
        static char segmentblock3  [4] = {0xFF, 0xFF, 0xFF,  0xF0};
        static char segmentblock4  [4] = {0xFF, 0xFF, 0xFF,  0xE0};
        static char segmentblock5  [4] = {0xFF, 0xFF, 0xFF,  0xC0};
        static char segmentblock6  [4] = {0xFF, 0xFF, 0xFF,  0x80};
        static char segmentblock7  [4] = {0xFF, 0xFF, 0xFF,  0x00};
        static char segmentblock8  [4] = {0xFF, 0xFF, 0xFE,  0x00};
        static char segmentblock9  [4] = {0xFF, 0xFF, 0xFC,  0x00};
        static char segmentblock10 [4] = {0xFF, 0xFF, 0xF8,  0x00};
        static char segmentblock11 [4] = {0xFF, 0xFF, 0xF0,  0x00};
        static char segmentblock12 [4] = {0xFF, 0xFF, 0xE0,  0x00};
        static char segmentblock13 [4] = {0xFF, 0xFF, 0xC0,  0x00};
        static char segmentblock14 [4] = {0xFF, 0xFF, 0x80,  0x00};
        static char segmentblock15 [4] = {0xFF, 0xFF, 0x00,  0x00};
        static char segmentblock16 [4] = {0xFF, 0xFE, 0x00,  0x00};
        static char segmentblock17 [4] = {0xFF, 0xFC, 0x00,  0x00};
        static char segmentblock18 [4] = {0xFF, 0xF8, 0x00,  0x00};
        static char segmentblock19 [4] = {0xFF, 0xF0, 0x00,  0x00};
        static char segmentblock20 [4] = {0xFF, 0xE0, 0x00,  0x00};
        static char segmentblock21 [4] = {0xFF, 0xC0, 0x00,  0x00};
        static char segmentblock22 [4] = {0xFF, 0x80, 0x00,  0x00};
        static char segmentblock23 [4] = {0xFF, 0x00, 0x00,  0x00};
        static char segmentblock24 [4] = {0xFE, 0x00, 0x00,  0x00};
        static char segmentblock25 [4] = {0xFC, 0x00, 0x00,  0x00};
        static char segmentblock26 [4] = {0xF8, 0x00, 0x00,  0x00};
        static char segmentblock27 [4] = {0xF0, 0x00, 0x00,  0x00};
        static char segmentblock28 [4] = {0xE0, 0x00, 0x00,  0x00};
        static char segmentblock29 [4] = {0xC0, 0x00, 0x00,  0x00};
        static char segmentblock30 [4] = {0x80, 0x00, 0x00,  0x00};
        static char segmentblock31 [4] = {0x00, 0x00, 0x00,  0x00};

        int counter = 0; //initialize global counter variable
        int i,j = 0; // initialize loops couter variables

        cout << "LAMACON Version 1.1 \n";
        cout << "Line Addressable Micromirror Array CONtrol Software \n";
        cout << "\n";
        cout << "Writing Upward Voltage Ramp / upTilt Pattern to (CF00..CFFF)
\n";
        cout << "For operation ABOVE cutoff frequency!  \n";

        // Repeats diagonal quanta pattern 32 times
        // Pulses constructed to keep central frequency component as
        //  large as possible
        for (i=0;i<32;i++)
        { //# of times to repeat each segment block
```

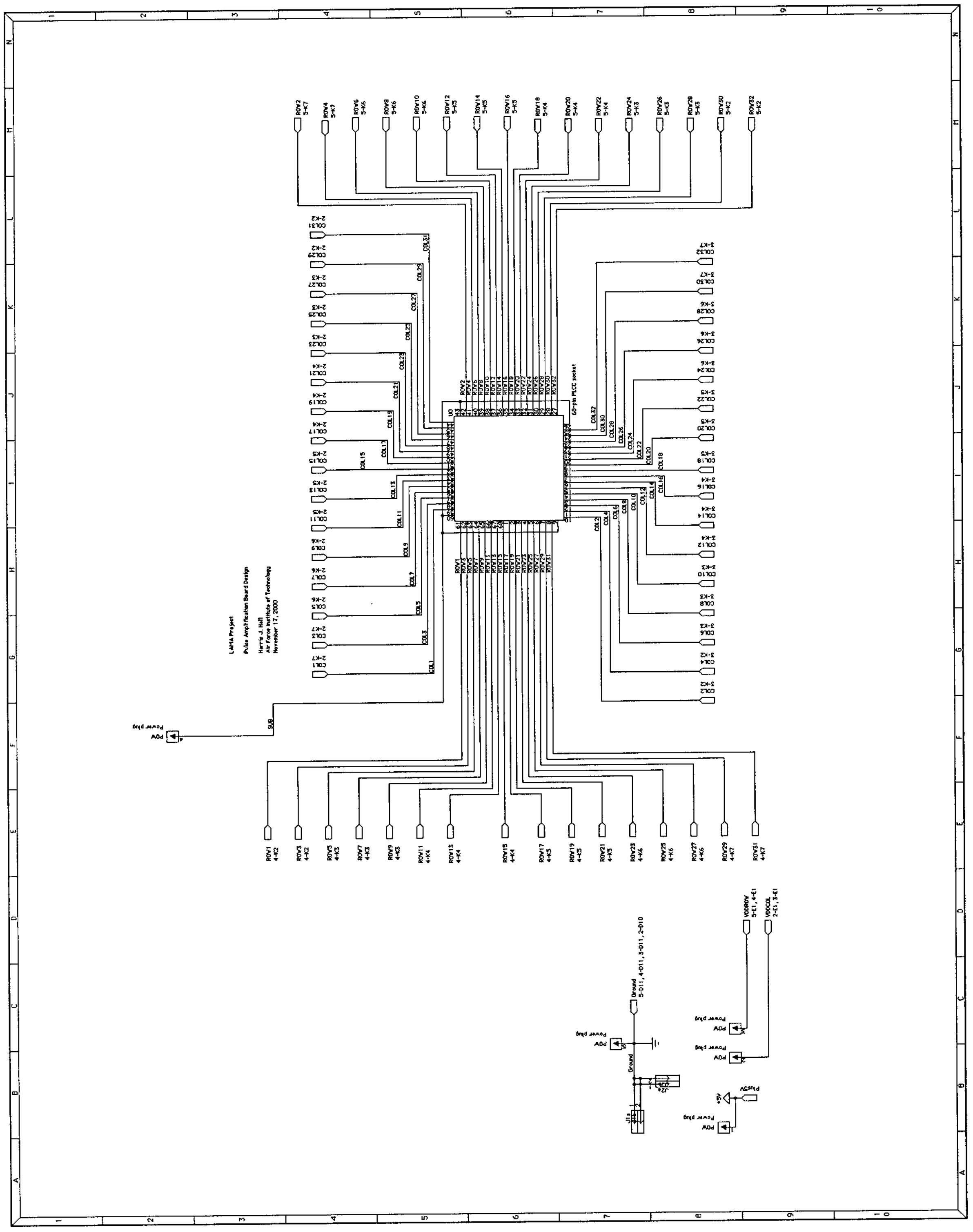
```
//segmentblock0 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock0[j]);
      counter++; // increment global counter
}

//segmentblock1 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock1[j]);
      counter++; // increment global counter
}

//segmentblock2 dowload to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock2[j]);
      counter++; // increment global counter
}

//segmentblock3 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock3[j]);
      counter++; // increment global counter
}

//segmentblock4 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock4[j]);
      counter++; // increment global counter
}

//segmentblock5 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock5[j]);
      counter++; // increment global counter
}

//segmentblock6 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock6[j]);
      counter++; // increment global counter
}

//segmentblock7 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock7[j]);
      counter++; // increment global counter
}

//segmentblock8 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock8[j]);
```

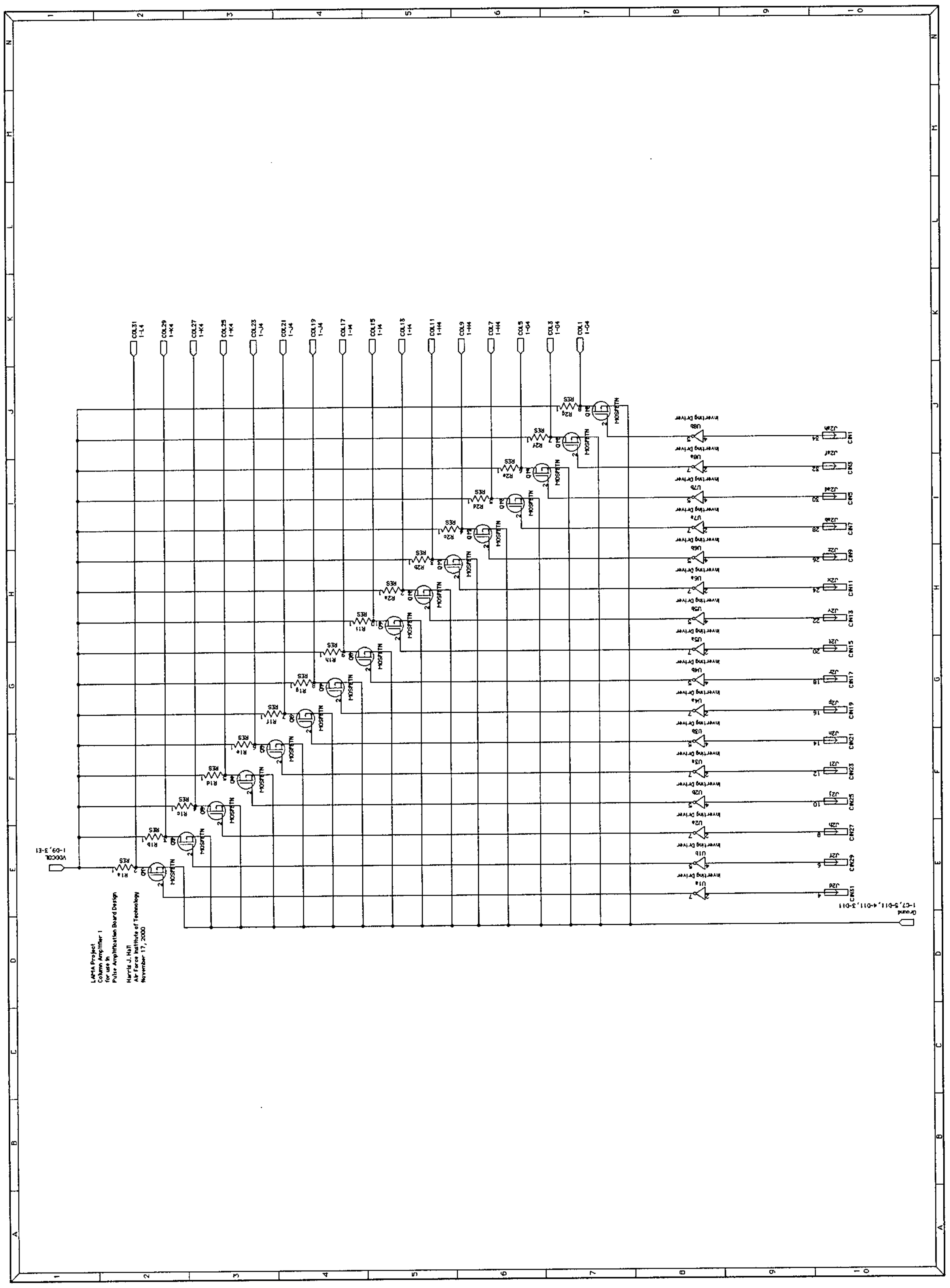
```
        counter++; // increment global counter
}

//segmentblock9 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
        pokeb(0xCF00, counter, segmentblock9[j]);
        counter++; // increment global counter
}

//segmentblock10 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
        pokeb(0xCF00, counter, segmentblock10[j]);
        counter++; // increment global counter
}

//segmentblock11 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
        pokeb(0xCF00, counter, segmentblock11[j]);
        counter++; // increment global counter
}

//segmentblock12 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
        pokeb(0xCF00, counter, segmentblock12[j]);
        counter++; // increment global counter
}

//segmentblock13 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
        pokeb(0xCF00, counter, segmentblock13[j]);
        counter++; // increment global counter
}

//segmentblock14 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
        pokeb(0xCF00, counter, segmentblock14[j]);
        counter++; // increment global counter
}

//segmentblock15 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
        pokeb(0xCF00, counter, segmentblock15[j]);
        counter++; // increment global counter
}

//segmentblock16 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
        pokeb(0xCF00, counter, segmentblock16[j]);
        counter++; // increment global counter
}

//segmentblock17 download to memory
```

```
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock17[j]);
      counter++; // increment global counter
}

//segmentblock18 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock18[j]);
      counter++; // increment global counter
}

//segmentblock19 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock19[j]);
      counter++; // increment global counter
}

//segmentblock20 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock20[j]);
      counter++; // increment global counter
}

//segmentblock21 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock21[j]);
      counter++; // increment global counter
}

//segmentblock22 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock22[j]);
      counter++; // increment global counter
}

//segmentblock23 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock23[j]);
      counter++; // increment global counter
}

//segmentblock24 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock24[j]);
      counter++; // increment global counter
}

//segmentblock25 download to memory
for (j=0;j<4;j++)
{ //# of 8-bit parts in each 32-bit segment block
      pokeb(0xCF00, counter, segmentblock25[j]);
      counter++; // increment global counter
```

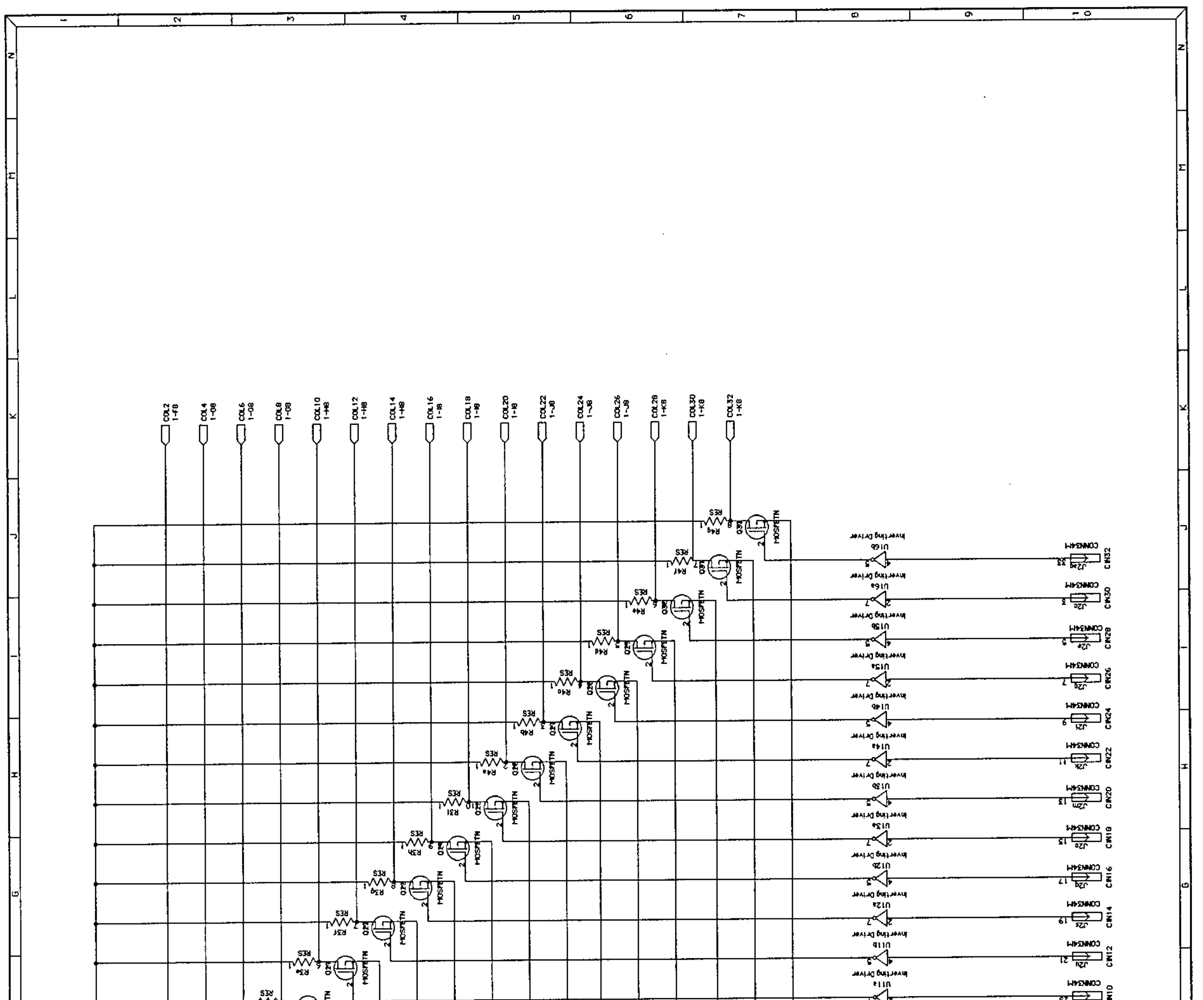
```
            }

            //segmentblock26 download to memory
            for (j=0;j<4;j++)
            { //# of 8-bit parts in each 32-bit segment block
                  pokeb(0xCF00, counter, segmentblock26[j]);
                  counter++; // increment global counter
            }

            //segmentblock27 download to memory
            for (j=0;j<4;j++)
            { //# of 8-bit parts in each 32-bit segment block
                  pokeb(0xCF00, counter, segmentblock27[j]);
                  counter++; // increment global counter
            }

            //segmentblock28 download to memory
            for (j=0;j<4;j++)
            { //# of 8-bit parts in each 32-bit segment block
                  pokeb(0xCF00, counter, segmentblock28[j]);
                  counter++; // increment global counter
            }

            //segmentblock29 download to memory
            for (j=0;j<4;j++)
            { //# of 8-bit parts in each 32-bit segment block
                  pokeb(0xCF00, counter, segmentblock29[j]);
                  counter++; // increment global counter
            }

            //segmentblock30 download to memory
            for (j=0;j<4;j++)
            { //# of 8-bit parts in each 32-bit segment block
                  pokeb(0xCF00, counter, segmentblock30[j]);
                  counter++; // increment global counter
            }

            //segmentblock31 download to memory
            for (j=0;j<4;j++)
            { //# of 8-bit parts in each 32-bit segment block
                  pokeb(0xCF00, counter, segmentblock31[j]);
                  counter++; // increment global counter
            }

            //segmentblock32 download to memory
            for (j=0;j<4;j++)
            { //# of 8-bit parts in each 32-bit segment block
                  pokeb(0xCF00, counter, segmentblock32[j]);
                  counter++; // increment global counter
            }

      }

      cout << counter <<endl;

      return 0; // exit program

} //end of upTilt.cpp
```

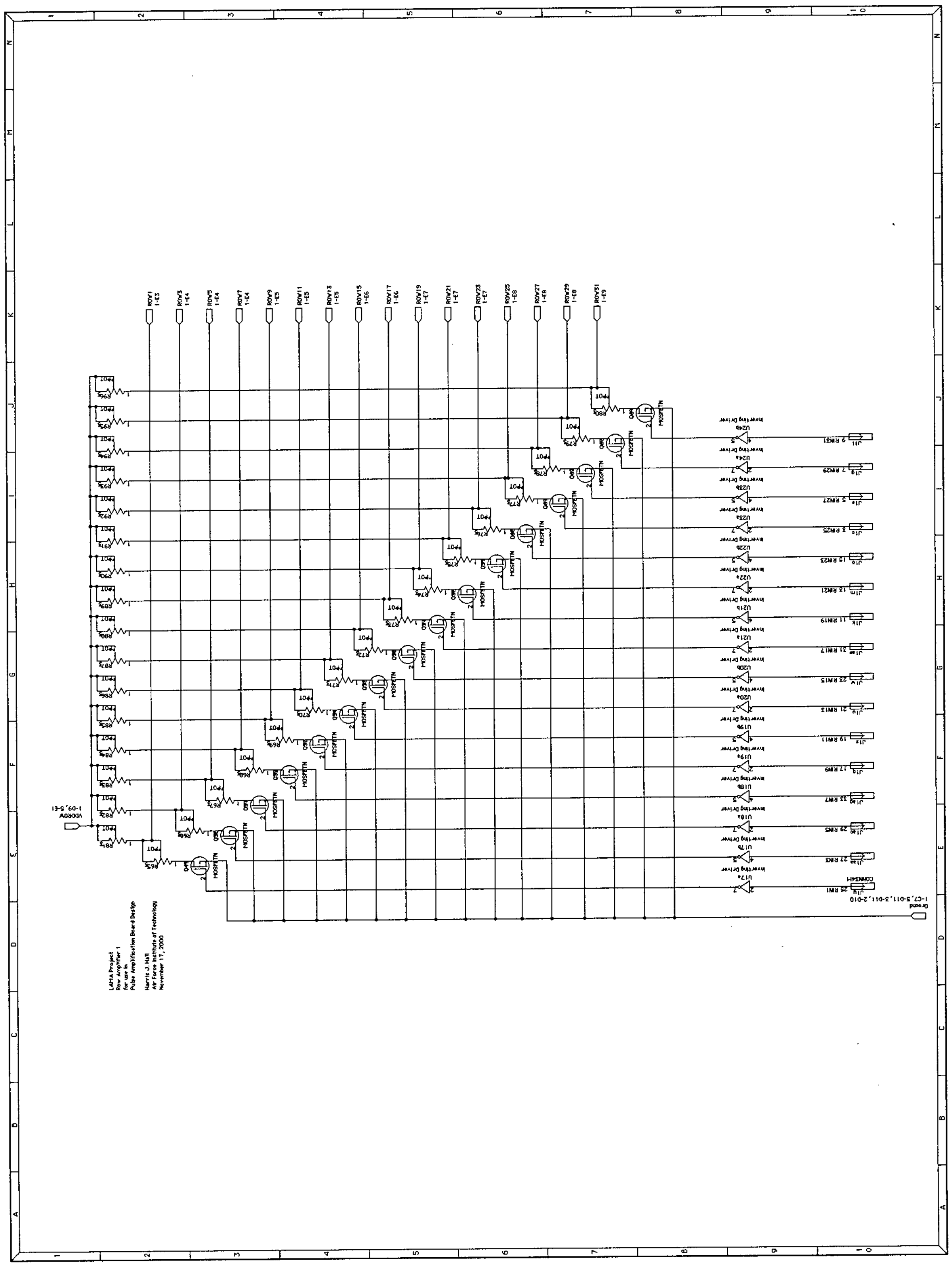
APPENDIX C – AMPLIFICATION BOARD SCHEMATIC

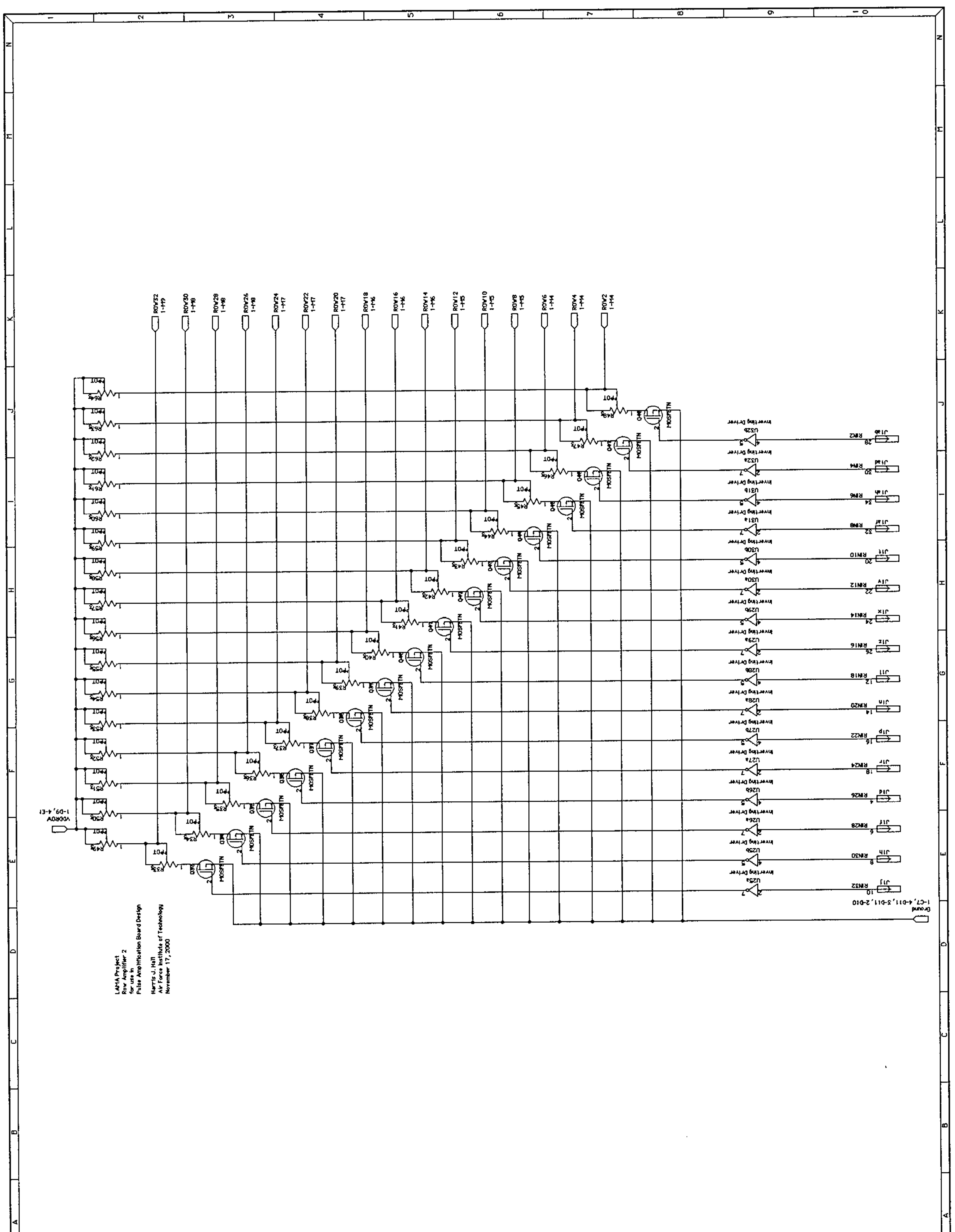




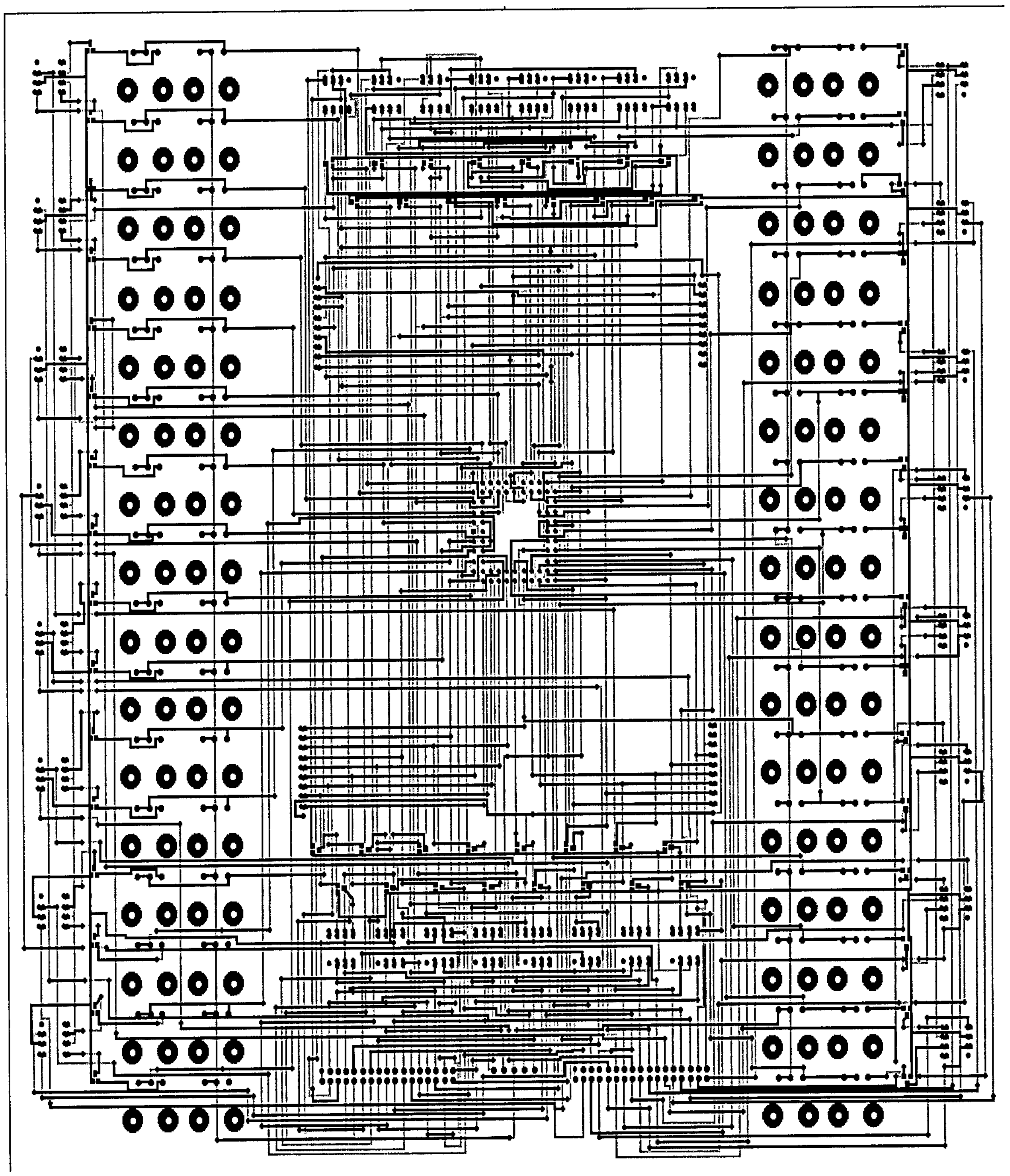
LAMA Project
Column Amplifier 1
for use in
Pulse Amplification Board Design
Harris J. Hall
Air Force Institute of Technology
November 17, 2000
VDDCOL
Ground
MOSFETN
RES
Inverting Driver


LAMA Project
Column Amplifier 2
For use in
Pulse Amplification Board Design
Harris J Hall
Air Force Institute of Technology
November 17, 2000
VDDCOL
Ground
MOSFETN
RES
Inverting Driver
CONN34M

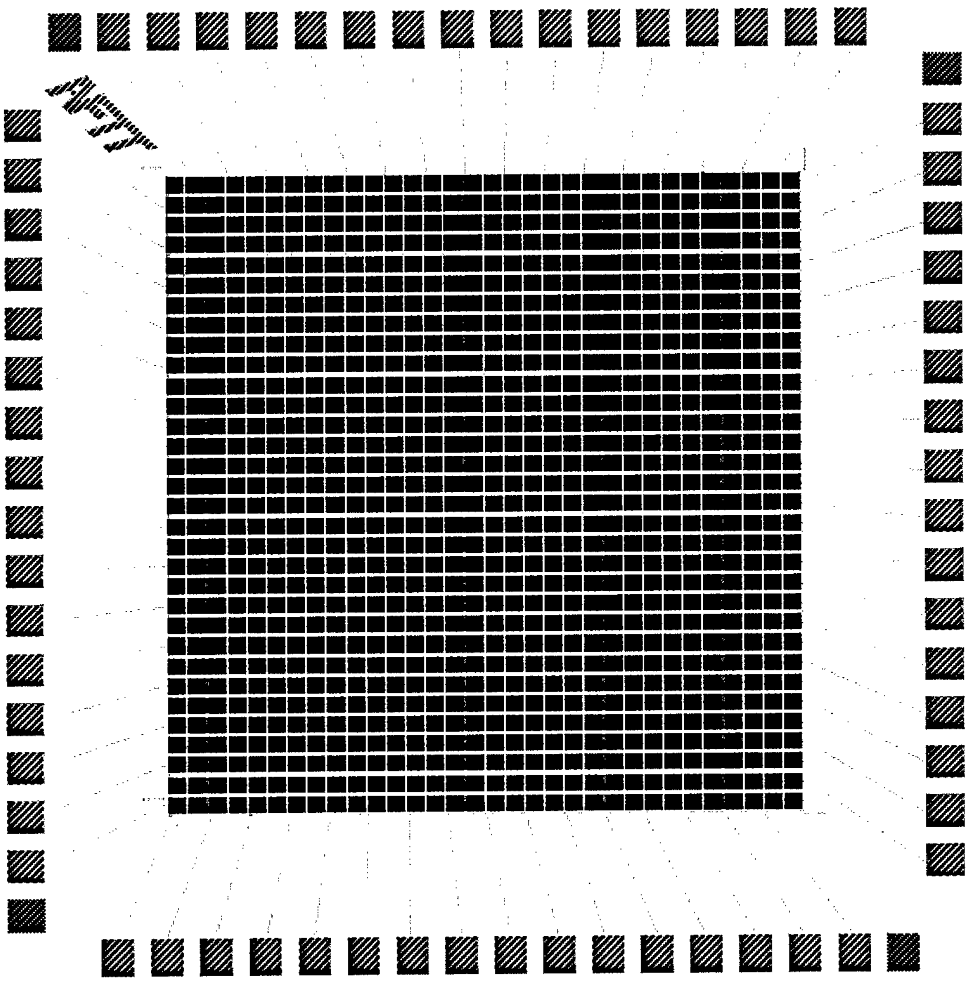

LAPHA Project
Row Amplifier 1
for use in
Pulse Amplification Board Design
Harris J. Hall
Air Force Institute of Technology
November 17, 2000
VDDROW
Ground
CONN34H
MOSFETN
Inverting Driver
ROW1 1-E3
ROW3 1-E4
ROW5 1-E4
ROW7 1-E4
ROW9 1-E5
ROW11 1-E5
ROW13 1-E5
ROW15 1-E6
ROW17 1-E6
ROW19 1-E7
ROW21 1-E7
ROW23 1-E7
ROW25 1-E8
ROW27 1-E8
ROW29 1-E8
ROW31 1-E9


LAMA Project
Row Amplifier 2
for use in
Pulse Amplification Board Design
Harris J. Hall
Air Force Institute of Technology
November 17, 2000
VDDROW
Ground
MOSFETN
POT
Inverting Driver

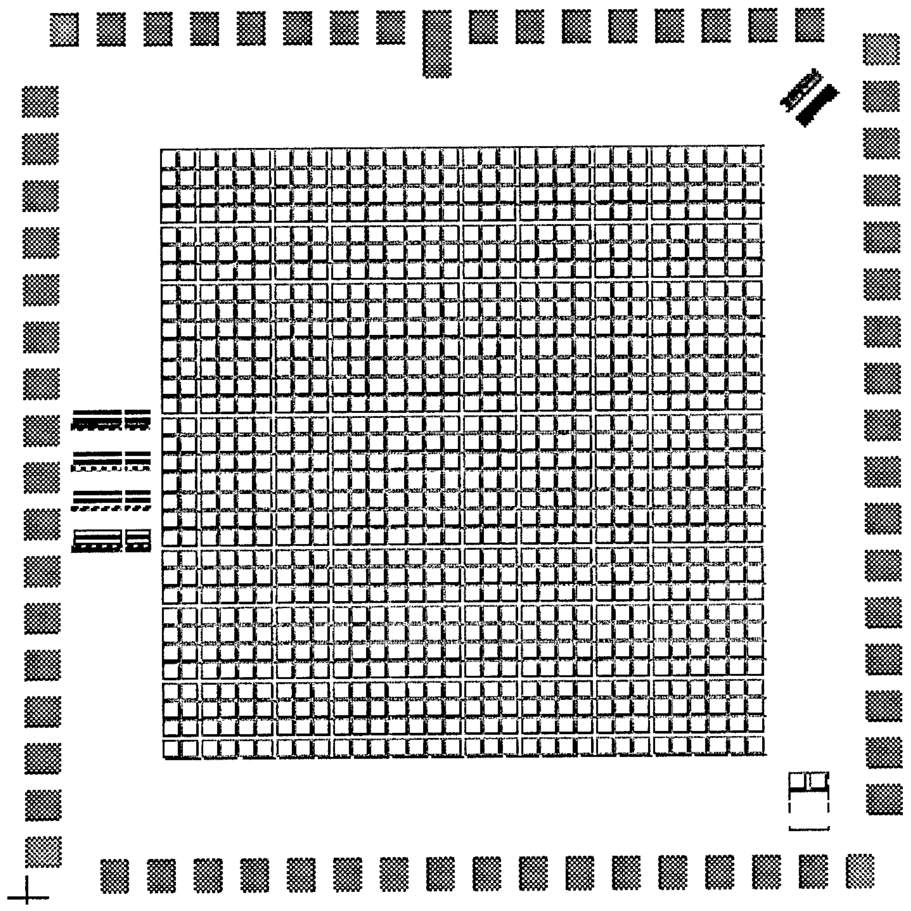
APPENDIX D – AMPLIFICATION BOARD LAYOUT



APPENDIX E – MICROMIRROR ARRAY DESIGN LAYOUT



Cronos Integrated Microsystem's Multi-User MEMS Process (MUMPs™) Line-addressable micromirror array design. Same layout was used for fabrications with and without metal layer. The fabrication run used was MUMPs™_24.



Sandia's Ultra-Planar Multi-level MEMS Technology (SUMMiT) Line-addressable micromirror array design. Beam structures shown to left of array are intended for residual stress testing of the releasable polysilicon layers.

APPENDIX F – RELEASE AND PACKAGING PROCEDURE

The following process was used to release the MUMPs™ and SUMMiT Line-Addressable Micromirror Array Devices. Emphasis is placed on maintaining the cleanest and safest possible environment to protect the handler and maximize device yield. The steps in this process are an extension of those listed in Cowan's dissertation [10].

Laboratory Safety Measures

- Appropriate safety gear is to be worn at all times when handling chemicals including laboratory coat/apron, latex gloves, and protective eyeware. All chemicals should be handled under a fume hood.
- Hydrofluoric acid (HF) is used in the process and is extremely hazardous. In case of skin contact apply calcium phosphate cream and seek immediate emergency treatment.
- Only trained personnel should handle spills in work area.
- A laboratory technician should be available at all times in case of emergency.

Chemical Bath Preparation

1. Five plastic beakers are required. Size of beakers will vary with number of die to be released (~ 1 liter every two die). Rinse them thoroughly with deionized water. Dry off excess with disposable laboratory wipe and blow dry with nitrogen gas.
2. Appropriately label beakers with China marker – *Acetone #1 and #2, Methanol #1 and #2, and HF (hydrofluoric acid).*
3. Fill beakers roughly halfway with fresh chemicals. Pour fresh chemicals from source container into #2 beakers, and then transfer into #1 beakers (if applicable). This is done to ensure that the #2 baths are the purest. Record the before and after weight of the source containers as well as the date and their serial number in laboratory log.
4. All handling of die to and from bathes should be accomplished with separate plastic tweezers.

Photoresist Removal

5. Bath die in Acetone #1 for 5-7 minutes. (bulk photoresist removal)

6. Bath die in Acetone #2 for 5-7 minutes. (finish photoresist removal)
7. Bath die in Methanol #1 for 5-10 minutes. (rinse off Acetone)

Mounting of Package

8. Place die in a clean petri dish and dry on hotplate at 55 °C.
9. Add post-foundry mirror metallization if desired
10. Place clean dry chip carriers on 150 °C hotplate and allow to warm.
11. Apply small dab/chip of hot melt adhesive (either MasterBond™ or CrystalBond™ 509) to chip carrier (glue flows ~ 121 °C)
12. Gently position die on chip carrier using a cotton tip applicator. Make sure die is oriented correctly.
13. Remove any excess glue and or debris with short soak (2-3 min) in Acetone. Use *Acetone #2* if it is not needed for further photoresist removal.
14. Soak die in *Methanol #1* for 5-10 min.

Releasing of die

15. Place packaged die in *HF*. Typical etch times for MUMPs™ device ~2.5 minutes , SUMMiT device ~ 45-55 minutes. It is believe that a slight agitation / stirring of the HF may help the etching process. Be extra careful when handling the HF.
16. Soak packaged die in *Methanol #1* for > 5 min (Methanol rinses remove any water underneath the mirror elements, this ensure that released elements won't stick due to charged ion buildup upon evaporation during drying)
17. Soak packaged die in *Methanol #2* for > 15 min. If necessary, device can be temporarily stored in methanol.
18. Dry packaged die on a ~ 55 °C hot plate. A good release is characterized by rapid evaporation of methanol without boiling.
19. Visually examine device with naked eye and under microscope. Broken or unreleased elements will exhibit some specular reflection. For SUMMiT devices, when viewing under microscope if intensity across elements appears to fluctuate while focusing or certain elements focus at different distances it is a good indication that elements are not uniformly deflected. **Do not blow air or nitrogen on devices!!**
20. If device is not completely released repeat releasing procedure. Be careful not to overetch (this will be obvious under microscope for MUMPs™ devices).

Connect to package

21. Warm-up wirebonding maching roughly 1hout before bonding

22. Wirebond electrical connections from package to device. This will be a difficult task for SUMMiT devices since connection pads are polysilicon on device. In addition, over time, a thin oxide layer will form on polysilicon making it increasingly difficult to bond. For bonding of multiple SUMMiT devices, work on one at a time and store others in Methanol #2.
23. If difficulty in bonding of SUMMiT devices persists, wave packaged die over the mouth of HF bottle.
24. In case of capillary failure on wirebonding machine contact laboratory technician.
25. Store packaged die in clean, dry sealed container. Place container in dry box for long-term storage.

<u>Cleanup Laboratory</u>

26. Empty both Methanol and Acetone dips into hazardous waste jar. Empty waste HF into separate plastic waste container. Do not store HF in glassware.
27. Rinse containers in DI water.
28. Disconnect hot plates (and turn off radio!) before leaving.

APPENDIX G – MATLAB® MECHANICAL RESPONSE CODE

The following is the contents of the "DeflectionResponse.m" MATLAB® function file, which produces a deflection response curve from the static fringe voltage:

```
function [Deflection, Voltage] = DeflectionResponse(Vstatic)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DeflectionResponse( Vstatic(in Volts) )
%       Computes the estimated deflection response based on the static fringe
%       voltage recorded.
%       Written by Harris J. Hall
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

help DeflectionResponse;  % print header

%Declare Variable Parameter Settings

A = (77e-6)*(88e-6);  %computes area of overlapping plates (meters)
epsilon = 8.851e-12;  %coefficient of permittivity in vacuum
((Coulombs^2)/(N*m^2))
dstatic = 316.4e-9;   %static fringe deflection (m) [for HeNe laser]
t = 2e-6;             %seperation between plates at equilibrioum (m)

%Step 1 - Calculate estimated total spring constant

k = (A*epsilon*(Vstatic)^2)/(2*dstatic*(t-dstatic)^2)  % (Newtons/meter)

%Step 2 - Generate Deflection Response Curve

Deflection = (0:1e-9:664e-9);
%Creates a deflection vector in increments of 1 nm
%Models only accurate for deflection < t/3 after which snap down occurs

Voltage = ((2*k/(A*epsilon))^0.5)*(Deflection.^0.5).*(t-Deflection);
```

The following is the contents of "DifferentialLineControl.m", the MATLAB® function file that generates a deflection map for the LAMA device based on the line inputs (requires roughly 160 sec of CPU time with dual 550MHz Pentium processors):

```
function [DeflectionMap, VoltageMap] = DifferentialLineControl(Deflection,
Voltage, Vrows, Vcols)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DifferentialLineControl( Deflection[x}, Voltage[x], Vrows tran[M], Vcols[N])
%       Computes differential voltage map and translates into deflection
pattern
%       based upon lookup tables.
%       Written by Harris J. Hall
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

help DifferentialLineControl;  % print header
tstart = cputime; %Marks start of CPU clock timer
```

```
%Step 1 - Generate Differential Voltage Map
VoltageMap = zeros(32);

A = ones(32,1)*Vrows;

B = (ones(32,1)*Vcols)'; %transpose for cols

VoltageMap = abs(A-B);  %takes absolute value of differential voltage

%Step 2 - Create Deflection Map by searching through lookup table
DeflectionMap = zeros(32);

for i = 1:32
    for j = 1:32
        for x = 1:665

            diff = abs(Voltage(x) - VoltageMap(i,j)) ;

            if diff == min(abs(Voltage-VoltageMap(i,j)));
                 DeflectionMap(i,j) = Deflection(x);
            end

        end
    end
end
```

The following is a sample MATLAB® script file used to generate corresponding response curves and pattern plots:

```
%Execution script to create estimated Deflection Profile based on
%static fringe voltage

tstart = cputime; %Marks start of CPU clock timer

Vstatic = 26

[Deflection, Voltage] = DeflectionResponse(Vstatic);

figure(1)
plot(Voltage,(Deflection./1e-9))
xlabel('Voltage (V)')
ylabel('Deflection (nm)')
title('Plot of Estimated Deflection vs. Voltage for LAMA element')

%Execution script to create 32x32 Deflection Map for arbitrary DC line voltages

Vrowamp = 26;
Vcolamp = 0;

Vrows = (1/32:1/32:1)*Vrowamp; %defines row voltages

Vcols = (1/32:1/32:1)*Vcolamp; %defines col voltages

[DeflectionMap, VoltageMap] = DifferentialLineControl(Deflection, Voltage,
Vrows, Vcols);

figure(2)
colormap(gray)
```

```
brighten(0.7)
bar3(VoltageMap)
title('3D bar chart indicating row voltage ramp distribution across LAMA')
xlabel('Rows')
ylabel('Columns')
zlabel('Voltage (Volts)')

figure(3)
colormap(gray)
brighten(0.7)
bar3(DeflectionMap./1e-9)
title('3D bar chart indicating deflection response to row voltage ramp
distribution across LAMA')
xlabel('Rows')
ylabel('Columns')
zlabel('Deflection (nm)')

CPU_Time_Elapsed = cputime - tstart  %displays elapsed CPU time used to run
algorithm

clear ans; %clears answer and tstart variables on workspace
clear tstart;
```

The following is the contents of “mechresp.m”, the script file that provides a simple mechanical frequency response model of an individual micromirror element based on the mass-spring system presented in Section 3.3.3.

```
%mechresp.m
%Second-order mass-spring model for mechanical response of micromirror element
%uses MATLAB Control System toolbox
%by Harris J. Hall

%Define parameters
rho = 2330; %density of polysilicon (kg/m^3)
vol = 3.74e-14; %volume of top plate (m^3)
B = 0;  %Dampening constant (N-s/m)
k = 22.6;  %Estimated spring constant (N/m)

%Calculate mass
M = rho*vol  %Mass of top plate

%Calculate modal parameters
omegaRes = (k/M)^0.5 ; %angular resonant frequency (rad/sec)
K = 1/k; %DC Gain
Q = 2*pi*(M/B); %Quality factor
fres = (1/ (2*pi) ) *omegaRes; %resonant frequency (Hz)

disp('Resonant Frequency (Hz):')
disp(fres)

disp('Angular Resonant Frequency (rad/sec):')
disp(omegaRes)

%Define transfer function
g = tf([(1/M)],[1 (B/M) (k/M)]);

%Generate Bode plot of frequency response (rad/sec)
figure(1)
bode(g)
```

```
%Generate step response Amp vs. time
figure(2)
step(g)
```

APPENDIX H – MATLAB® FAR-FIELD SIMULATION CODE

The following is the contents of the "FarFieldSim.m" MATLAB® function file:

```
function [apImage, FF, u] = FarFieldSim(patternfilename,wavelength,z)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FarFieldSim( patternfilename, wavelength(nm), z(meters) )
%       Computes the estimated far-field pattern of an arbitrary 32x32 square
%       micromirror array binary phase pattern defined in patternfilename.
%       Written by Harris J. Hall (Special Thanks to Maj Roger Claypoole)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

help FarFieldSim;  % print header
tstart = cputime; %Marks start of CPU clock timer

%Declare Variable Parameter Settings
N_FFT=2048; %Number of pixels in final far-field pattern (determines extent of
zero padding)
T=3e-6;     %T is the spatial period represented by each pixel (in meters)
            %   in this instance we have 1 pixel = 3 microns

N = 32;  % number of phase pixels defining mirror element
M = 1;   % number of pixels defiing gap between adjacent elements (dead space)
downGap = 1050; %defines difference in height (in nm) between up position and
                %snap-down
elementDip = 100; %peak dip in nm for element curvature
arrayDip = 100; %peak dip in nm for macro array curvature

%Preliminary argument check
if nargin ~= 3  %checks if z argument is present when function is called
       z = 1; %default value is z=1 meter
end  %end of if conditional

if nargin >= 2  %checks if wavelength argument is present when function is
called
    elementPhase = (downGap / wavelength) * 2;  %phase accrued for snap-down
(in radians)
                                                %  x2 for reflection
else    %default wavelength is 632.8nm HeNe
    elementPhase = 1.6592;
    wavelength = 632.8;
end  %end of if conditional

elementDip = (elementDip)/wavelength;
%peak amount phase accrued at center of element
%used to scale element curvature profile (radians)

arrayDip =  (arrayDip)/wavelength;
%peak amount phase accrued at center of array
%used to scale macro array curvature profile (radians)

clear downGap; % removes gap variable from memory
```

```
%Step 1 - Read in 32x32 pattern array from *.pat file
%         The data is placed in a variable that has the same name
%           as the filename without the extension.

if nargin == 0

    fileName = 'blank.pat'; %default to empty array if no filename is entered

else

    fileName = num2str(patternfilename); %converts patternfilename to string

end %end of if conditional

smallArray = load(fileName); %reads space delimited 32x32 matrix from text file

%Step 2 - Creates 32x32 exponential array
phase = exp(j*smallArray*elementPhase);

%Step 3 - Create a 33x33 intermediate array that will act as amplitude filter
disp('Creating initial phase array...')
b = zeros(N+M,N+M);
b(1:N,1:N) = 1;

%Step 4 -  Build our big phase map with dead space (1056x1056 pixels)
apArray = kron(phase,b);
%kroneker delta multiplication generates 1056x1056 array

%Step 5 - Create pseudo-image of aperature
apImage = kron(smallArray + 0.5,b);

%Step 6 - Create and apply element curvature profile
disp('Applying element curvature profile...')
elementProfile = zeros(N,N); %32x32 array

p=1:N;
L = sin((p-1)*(pi)/31); %apply scaled Lambertian distribution
elementProfile = transpose(L)*L*elementDip;
%this operation is equivalent to, but faster then a looped element by element
assignment
%elementProfile(p,q) = sin((p-1)*(pi)/31)*sin((q-1)*(pi)/31)*elementDip;
clear p,L; %reset p and L for future allocation

elementProfile = exp(j*elementProfile); %32x32 Array

%generate 1056x1056 curvature array and combine for total phase map
apArray = apArray + kron(elementProfile,b);
clear elementProfile;

%Step 7 - Create and apply macro curvature profile
disp('Applying macro curvature profile...')
arrayProfile = zeros(1056,1056);

p=1:1056;
L = sin((p-1)*(pi)/1055);  %apply scaled Lambertian distribution
arrayProfile = transpose(L)*L*arrayDip;
%this operation is equivalent to, but faster then a looped element by element
assignment
%arrayProfile(p,q) = sin((p-1)*(pi)/1055)*sin((q-1)*(pi)/1055)*arrayDip;
```

```
%combine phase map with array curvature for total curvature
apArray = apArray + arrayProfile;
clear arrayProfile;

%Step 8 - Take FFT and center resulting pattern in freq spectrum
FF=fftshift(fft2(apArray,N_FFT,N_FFT));
FF=abs(FF).^2; % converts to intensity

%Step 9 - Scales far-field pattern according to wavelength and z-distance
fx=[-1/(2*T):1/(N_FFT*T):1/(2*T)-1/(N_FFT*T)];
u=(wavelength*10^-9)*z*fx;

CPU_Time_Elapsed = cputime - tstart
%displays elapsed CPU time used to run algorithm
```

The following is a sample MATLAB® script file used to display images:

```
%Execution script to display aperature and far-field patterns

[apImage,FF,u] = FarFieldSim('blank.pat');

figure(1)
imagesc(apImage);

figure(2)
imagesc(u,u,log(1+FF));

figure(3)
p=1:32;
L = sin((p-1)*(pi)/32); %apply scaled Lambertian distribution
A = transpose(L)*L*100;
imagesc(p,p,A);

figure(4)
mesh(p,p,A);
axis([0 32 0 32 0 100])

clear ans; %clears answer variable on workspace
```

REFERENCES

"From the moment I picked your book up until I laid it down I was convulsed with laughter. Some day I intend to read it."

-Groucho Marx (1895-1977) [38]

1. "ME235 Help Files Web Page, "Pulse Width Modulation",": http://mechsys2.me.berkeley.edu/ME235/LabFiles/pwm.html.

2. *Maxim 3D Operation and Maintenance Manual.* Middlefield, CT: Zygo Corporation, 1988.

3. *User's Manual 16 Bit Prototype Board JDR-PR10*: JDR Microdevices, 1988.

4. Butler, J. T., V. M. Bright, and W. D. Cowan, "Average power control and positioning of polysilicon thermal actuators," *Sensors and Actuators A*, vol. 72, pp. 88-97, 1999.

5. Carlson, A. B., *Circuits : Engineering Concepts and Analysis of Linear Electric Circuits*, Preliminary ed. New York: John Wiley & Sons, Inc., 1996.

6. Chan, E. K. and R. W. Dutton, "Electrostatic Micromechanical Actuator with Extended Range of Travel," *Journal of Microelectromechanical Systems*, vol. 9, pp. 321-328, 2000.

7. Chu, P. B. and K. S. J. Pister, "Analysis of Closed loop Control of Parallel-Plate Electrostatic Microgrippers," *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 820-825, 1994.

8. Collins, D. R., et al., "Deformable mirror device spatial light modulators and their applicability to optical neural networks," *Applied Optics*, vol. 28, pp. 4900-4907, 1989.

9. Comtois, J. H. and V. M. Bright, "Surface Micromachined Polysilicon Thermal Actuator Arrays and Applications," in *Solid-State Sensor and Actuator Workshop*, pp. 174-177, 1996.

10. Cowan, W. D., "Foundry Microfabrication of Deformable Mirrors for Adaptive Optics," Ph.D. Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1998.

11. Cowan, William D., Major, USAF, Adjunct Professor AFIT Wright-Patterson AFB, OH, private communication. June 14, 2000.

12. Cowan, W. D. and J. T. Butler, "Microsensors and Microactuators Course Notes,". WPAFB, OH: Air Force Institute of Technology, 2000.

13. Cowan, W. D., M. K. Lee, B. M. Welsh, V. M. Bright, and M. C. Roggemann, "Surface Micromachined Segmented Mirrors for Adaptive Optics," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 5, pp. 90-101, 1999.

14. Cunningham, B. T., J. J. Bernstein, D. Seltzer, and D. Hom, "Micromirror Pixel Addressing using Electromechanical Bistability," in *Solid-State Sensor and Actuator Workshop*, 1998.

15. D'Azzo, J. J. and C. H. Houpis, *Linear Control System Analysis and Design: Conventional and Modern*, 4th ed. New York: McGraw-Hill, Inc., 1995.

16. Deitel, H. M. and P. J. Deitel, *C++ How to Program*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1998.

17. Francais, O., "Analysis of an Electrostatic Microactuator with the help of Matlab/simulink: transient and frequency characteristics," in *Technical Proceedings on the MSM 2000 International Conference of Modeling and Simulation of Microsystems*, 2000.

18. Goodman, J. W., *Introduction to Fourier Optics*, 2nd ed. New York: The Mc-Graw-Hill Companies, Inc., 1996.

19. Hick, S. R., "Demonstrating Optical Aberration Correction with a MEMs Micromirror Device," MS Thesis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1996.

20. Hornbeck, L. J., "Deformable-mirror spatial light modulators," *Spatial Light Modulators and Applications III, Proc. SPIE*, vol. 1150, pp. 86-102, 1990.

21. Hornbeck, L. J., "From cathode rays to digital micromirrors: A history of electronic projection display technology," TI Technical Journal, Texas Instruments, Dallas, Texas, 1998.

22. Huja, M. and M. Husak, "Micromirror Array Design and Simulation," in *Technical Proceedings on the MSM 2000 International Conference of Modeling and Simulation of Microsystems*, 2000.

23. Hulberd, B. and D. Sandler, "Segmented mirrors for atmospheric compensation," *Optical Engineering*, vol. 29, pp. 1186-1190, 1990.

24. Jaecklin, V. P., C. Linder, N. F. de Rooij, J.-M. Moret, and R. Vuilleumier, "Line-addressable torsional micromirrors for light modulator arrays," *Sensors and Actuators A*, vol. 41-42, pp. 324-329, 1994.

25. Johnson, R. C., "Micromirror arrays perform photolithography step," *EE Times*, Oct 12, 1999.

26. Koester, D. A., R. Mahadevan, A. Shishkoff, and K. W. Markus, *MUMPs Design Handbook.* Research Triangle Park, NC: Cronos Integrated Microsystems, 1999.

27. Kolesar, E. S., P. B. Allen, J. W. Wilken, and J. T. Howard, "Implementation of micromirror arrays as optical binary switches and amplitude modulators," *Thin Solid Films*, vol. 332, pp. 1-9, 1998.

28. Kovacs, G. T. A., *Micromachined Transducers Sourcebook.* New York: WCB / McGraw-Hill, 1998.

29. Lin, T.-H., "Implementation and Characterization od a Flexure-Beam Micromechanical Spatial Light Modulator," *Optical Engineering*, vol. 33, pp. 3643-3648, 1994.

30. Liu, C. and Y. Bar-Cohen, "Scaling Laws of Microactuators and Potential Applications of Electroactive Polymers in MEMs," in *Proceedings of SPIE 6th Annual International Symposium on Smart Structure and Materials*, 1999.

31. Madou, M., *Fundamentals of Microfabrication.* Cleveland, OH: CRC Press, 1997.

32. Michalicek, M. A., "Design, Fabrication, Modeling, and testing of Surface-Micromachined Micromirror Devices," MS Thesis, Department of Electrical and Computer Engineering, Air Force Institute Of Technology, WPAFB, OH, 1995.

33. Overland, B., *Visual Basic 6 in Plain English.* Indianapolis, IN: IDG Books Worldwide Inc., 561, 1999.

34. Pelesko, J. A. and A. A. Triolo, "Nonlocal Problems in MEMS Device Control," in *Technical Proceedings on the MSM 2000 International Conference of Modeling and Simulation of Microsystems*, 2000.

35. Peterson, K. E., "Micromechanical light modulator array fabricated on silicon," *Applied Physics Letters*, vol. 31, pp. 521, 1977.

36. Poularikas, A. D. and S. Seely, *Signals and Systems*, 2nd ed. Boston: PWS-KENT Publishing Company, 1991.

37. Roberts, P. C., "Modeling and Simulation of Optical Characteristics of Microelectromechanical Mirror Arrays," MS Thesis, Department of Engineering Physics, Air Force Institute of Technology, WPAFB, OH, 1996.

38. Robins, G., "Good Quotations by Famous People,". Charlottesville, VA: http:/www.cs.virginia.edu/~robins/quotes.html, 2000.

39. Roden, M. S., "Section 6.5 - Pulse Width (Duration) Modulation," in *Analog and Digital Communication Systems.* Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1979, pp. 262-266.

40. Rounsavall, P. C., "Modulation of Electrostatic Microelectromechanical Mirrors Using a CMOS Controller," MS Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1999.

41. Seeger, J. J. and S. B. Crary, "Analysis and Simulation of MOS Capacitor Feedback for Stabilizing Electrostatically Actuated Mechanical Devices," in *Second international Conference on the Simulation and Design of Microsystems and Microstructures - MICROSIM97*, pp. 199-208, 1997.

42. Solari, E., *ISA & EISA Theory and Operation*. Poway, CA: Annabooks, 495, 1999.

43. Sze, S. M., *Physics of Semiconductor Devices*. New York: John Wiley & Sons, 1994.

44. Tay E.H., F., R. Kumaran, B. L. Chua, and L. VJ, "Electrostatic Spring Effect on the Dynamic Performance of Microresonators," in *Technical Proceedings of the MSM 2000 International Conference on Modeling and Simulation of Microsystems*, 2000.

45. Toth, V., "Chapter 13 - Memory Management," in *Visual C++ 4 Unleashed*. Indianapolis, IN: Sams Publishing, 1996.

46. Zhulin, V. I., S. J. Owen, and D. F. Ostergaard, "Finite Elemnt Based Electrostatic Structural Coupled Analysis with Automated Mesh Morphing," in *Technical Proceedings of the MSM 2000 International Conference on Modeling and Simulation of Microsystems*, 2000.

VITA

"Don't be so humble – you are not that great."
-Golda Meir (1898-1978) to a visiting diplomat

Harris Joseph Hall was born in Suffern, New York to Richard and Leslee Hall. Upon graduating from Suffern High School in 1995, he accepted an Air Force ROTC scholarship at Detachment 550, Rensselaer Polytechnic Institute (RPI) . In May 1999, he graduated from RPI Magna Cum Laude with a Bachelor of Science degree in Electrical Engineering and was commissioned into the United States Air Force as a Second Lieutenant.

Harris's first assignment was to attend the Air Force Institute of Technology (AFIT) at Wright-Patterson AFB, Ohio, to study in the Electro-Optics program. He graduated from AFIT March 2001 with a Masters of Science degree in Electrical Engineering. His follow-on assignment is to Air Force Research Laboratory's (AFRL) Starfire Optical Range at Kirtland AFB, New Mexico where he anticipates working in the field of adaptive optics.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 05-03-2001	2. REPORT TYPE Master's Thesis	3. DATES COVERED Jan 2000-Feb 2001

4. TITLE AND SUBTITLE CONTROL AND CHARACTERIZATION OF LINE-ADDRESSABLE MICROMIRROR ARRAYS	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Hall, Harris, J., Second Lieutenant, USAF	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 P Street, Building 640 WPAFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GEO/ENG/01M-01
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Attn: William Cowan, Lt Col, USAF Hardened Materials Branch Bldg 651 Rm 169 WPAFB OH 45433-7765 DSN: 785-3808 x3148	10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/MLPJ
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research involved the design and implementation of a complete line-addressable control system for a 32x32 electrostatic piston-actuated micromirror array device. Line addressing reduces the number of control lines from N^2 to 2N making it possible to design larger arrays and arrays with smaller element sizes. The system utilizes the electromechanical bi-stability of individual elements to hold arbitrary bi-stable phase patterns. The control system applies pulse width modulated (PWM) signals to the rows and columns of the micromirror array. Three modes of operation were conceived and built into the system. The first was the traditional signal scheme which requires the array to be reset before a new pattern can be applied. The second is an original scheme that allows dynamic switching between bi-stable patterns. The third and final mode applies an effective voltage ramp across the device by operating above mechanical cutoff. Device characterization and control system testing were conducted on predesigned and prefabricated samples from two different foundry processes. Testing results showed that the control system was successfully integrated. However, bi-stable control of individual mirror elements was not successfully demonstrated on samples due to flaws in the device design. A more robust device design which corrects these flaws and increases operational yield is proposed.

15. SUBJECT TERMS

MEMS, MUMPS, SUMMiT, micromirrors, micro-optics, micro-actuators, PWM, electrostatic, electromechanical, bi-stability, line-addressing, micromachining

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Magee, Eric P., Major, USAF (AFIT/ENG)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	185	(937)255-3636