A LINEAR ULTRASONIC MOTOR FOR NANO-TECHNOLOGY

by

HENRY O. CHOI

B.S., California Institute of Technology (1994)

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1996

Author .....
Department of Mechanical Engineering
May 18, 1996

Certified by .....
Kamal Youcef-Toumi
Associate Professor
Thesis Supervisor

Accepted by .....
Ain A. Sonin
Chairman, Departmental Committee on Graduate Students

A Linear Ultrasonic Motor for Nano-Technology

by

Henry O. Choi

Submitted to the Department of Mechanical Engineering
on May 18, 1996, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

In response to rising demand for high performance actuators to serve nano-technology, a linear ultrasonic motor was designed and constructed to verify its potential as a viable nano-actuator. After establishing the analytical model that elucidates the physics of the ultrasonic motor operation, experimental performance tests were conducted in a fully digital controlled real-time control environment. This newly developed system fuses flexible user interface with high speed data acquisition and manipulation ability by incorporating a powerful DSP board into the PC based operating kernel. The experiments demonstrate ultrasonic motor's excellent potential as a nano-actuator that will drive tomorrow's high performance devices. Extremely high speed motion capability was achieved without sacrificing either the sub-nanometer positioning accuracy, high bandwidth, high static stiffness, or high torque to weight ratio. These performance advantages over other nano-actuators stem from effective design that couples superior material property of piezoelectric material with a simple structure, and allows direct friction contact between the motor and the linear guide.

Thesis Supervisor: Kamal Youcef-Toumi
Title: Associate Professor

Acknowledgments

It seems only a moment ago, far less than the blink of an eye, that I was writing the acknowledgements for my senior thesis to my mentors and friends at Caltech. But despite all my previous fears, doubts, efforts, and even hopes, here I am now, writing yet another acknowledgement, for another thesis. During the past two years, I have changed; I am more mature and hopefully wiser, with more experience and learning. Though a youthful fire still burns in my soul, I am now better capable of grasping the intangible, and appreciating the gravity and beauty of life. For this, I am grateful for my graduate school experience at MIT. But what *is* the graduate school experience? For me, it is an aggregate sum of, and the interaction of, all elements that filled my life within the past two years: research project, classes, teachers, family and friends, the Cape Cod sunset, just to name a few.

I regret that I have not the space enough to thank all my benefactors, and not the wit enough to confer befittingly flourishing acknowledgement to those few I am compelled to mention. Grotesquely inadequate though my acknowledgements may be, it would be a great injustice to gloss over their names purely on the account of my poor literary skills.

My foremost acknowledgement goes to my advisor Professor Kamal Youcef-Toumi, who at times seemed like a friend than just an advisor; his guidance and support was indespensible in carrying out this research. I am also grateful for the helpful discussions with my labmates T.J., Shi-Ying, Tetsuo, Jake, Tarzen, and Mitchell; they were like second advisors to me, and did not spare themselves to walk me through my own blunders. This research was magnanimously supported by Daewoo Electronic Inc.

Outside the lab, I enjoyed a troop of friends and mentors in sspark, kdsik, Won, marlboro, mystery, Woosok, and the Hansori undergraduates. They were directly involved in my personal growth and developement. I wish them nothing short of spectacular future and life lived to its fullest.

Lastly, but absolutely not leastly, my family. This year marks the tenth anniversary of the date on which my family set root in this country. The immigrant life has been a hard one for all of us. But we have weathered it, and are now growing into our own bedrock capable of supporting not only ourselves, but each other, and even those around us. So I thank my parents, who sacrificed to no end, and held together our family, my sister Danbi, whose American name Deborah seems to have been predestined, for she is a wise and courageous woman capable of leading others, and finally my little brother Ujin, who suffered mightily in his search for his own identity and is now cultivating a beauty and strength that befits only him.

Chapter 1

Introduction

1.1 Advent of nano-technology

In these days, every year, and frequently every day, the world of technology viewed as a collective makes breakthroughs in electronics, computers, communication, energy, transportation, chemistry, biology, and whatever field that constitutes a modern technological world. These breakthroughs in turn drive what some view as increasingly faster jumps in upward exponential technology curves and force a shift in attitudes and resources in not only technology driven manufacturing economy, but the world and the human race as a whole.

In light of such overwhelming power of technology, we are obsessed with measuring our own technological prowess as well as another's. As convenient yardstick, we compare our monument machines, the so called state of the art machines that measure the machines that make other machines, and so forth, forming the technology pyramid shown in Fig 1-1. At the top are the monument machines, the best that we are capable of at the moment, from which the technology that will build lower class machines trickle down. Although the benefit of building one of a kind and hence costly monument machines is not felt for most people, the pyramid effect renders the monument machines far more value than their immediate stated worth. Two things are important in this technology pyramid: while only intense focused research and development efforts can bring out a monument machine, maintaining a well balanced technology pyramid ultimately broadens and fosters a healthy manufacturing economy, which can support an even more advanced monument machine and close the technology advancement loop. Only a few with vision, sufficient resource, and then luck can push the technology frontier; towing the rest of the technology pyramid is far trickier.

The three fundamental units in our physical world are mass, length, and time; any other units can be broken down to a mathematical expression of these three units. For example, according to Newton's First Law, force can be expressed as mass times acceleration, which in turn is distance divided by time squared. A monument machine pushes the frontier in one or more of these three fundamental units. This thesis is concerned with monument machines in length and their direct descendents; to be more specific, with a method that will push actuation accuracy down to sub-nanometer, and an actuator to prove the claim. For the sake of convenience, such high performance actuators shall be called nano-actuators throughout this thesis. Accordingly, the technology that encompasses the nano-actuators and nano-sensors is termed nano-technology.

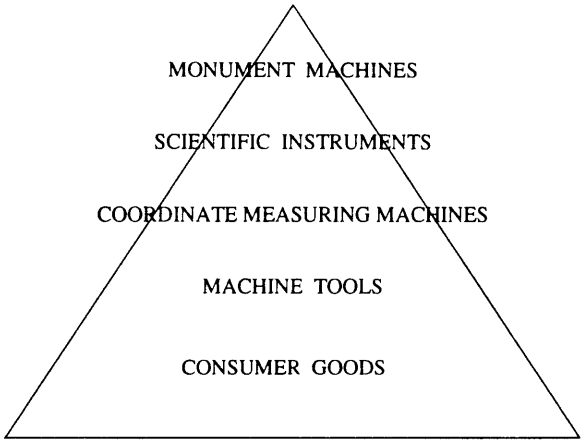


Figure 1-1: Technology pyramid

method	strain	hysteresis	aging	response time
thermal expansion	$10^{-5} \sim 10^{-3}$	low	low	s
magnetostriction	$10^{-5} \sim 10^{-3}$	large	low	$ns \sim \mu s$
piezoelectricity	$10^{-4} \sim 10^{-2}$	large	large	$\mu s \sim ms$
electrostriction	$10^{-9} \sim 10^{-3}$	low	low	μs

Table 1.1: Solid displacement actuation methods

The need for such fantastic accuracy ultimately arises out of the need to make the next generation of consumer products. The next generation of Pentium microprocessor, the next internet structure, the next turbofan engine, the next TV, and even the next razor; every part of this technology driven society demands even more fantastic performance from one another. Actuators and sensors are no exception. Already, most machines tools and even some high end consumer goods, which are at the bottom of the technology pyramid, are measured in microns and sub-microns; not surprisingly, monument machines are measured in nano-meters or angstroms–down to the scale of atoms. The large optics diamond turning machine (LODTM) developed at the Lawrence Livermore National Laboratory (LLNL) is one such machine [29]. LODTM's accuracy is measured in parts per billion, putting it near the top of the technology pyramid. Clearly, a physical metrology frame based precision machine such as LODTM require high performance sensors and actuators that are measured in nano-meters and angstroms, that is, nano-actuators.

1.2 Linear ultrasonic motor as a nano-actuator

Currently, several types of nano-actuation methods are used for high precision control systems depending on the system requirements. Table 1.1 compares some of such solid displacement methods [38]. Among these, piezoelectricity is the most commonly used method in high precision control systems because of its wide bandwidth, relatively long range, high energy density, and low heat generation, despite the nonlinear effects such as hysteresis,

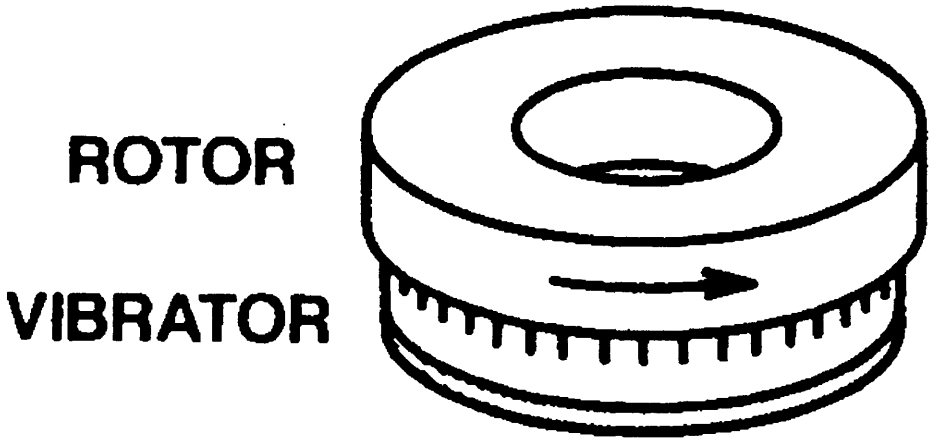


Figure 1-2: Traveling wave ring type ultrasonic motor

creep and aging.

Piezoelectric actuators have found widespread applications in high precision positioning devices including VCR auto tracking head, high precision machine tool, Scanning Probe Microscope (SPM) [4], and ultrasonic motors [26]. Even though piezoelectric material exhibits relatively large displacement among the solid displacement material in Table 1.1, the achievable range is still severely limited. Stacking many piezoelectric material one on top of another to make a piezo stack is one method of amplifying the displacement, and using the bimorph effect to make a piezo tube effectively increases the mechanical advantage. In another instance, semi-active suspension design, piezo stacks are used in conjunction with a "fluidic amplifier" to increase the range of motion. Using such concepts, many piezoelectric actuator packages have been invented, and inch worm motor and ultrasonic motor are among them. Since both are based on similar principle and share common features, only the ultrasonic motor will be discussed here; conceptually, the inch worm motor may be considered yet another version of ultrasonic motor.

Before revealing the principles of the ultrasonic motor operation, a brief explanation of the ultrasonic motor nomenclature is due. Firstly, by definition, the term ultrasonic implies that the motor is operating beyond audible frequency. Also, to draw parallel to the well characterized DC motors, the terms rotor and stator will be used frequently. Through the steps expounded in Chapters 2 and 3 and as shown in Fig 1-2, the stator imparts driving torque (or force in the case of linear ultrasonic motors) on the rotor, which in turn drive the load. Ultrasonic motors operate by generating either a standing or traveling wave on the stator such as the one shown in Fig 1-2 to establish elliptical motion at the friction contact area between the stator and the rotor, and converting this vibration into either linear or rotary motion of the rotor. Until recently, research and development efforts were exclusively confined to Japan, and the target applications ranged anywhere from camera auto-focus lense [14] to micro robots [16]. Popular configurations reported in the literature are the traveling wave on a ring or a disk type [15, 34, 19, 36, 32] with the configuration shown in Fig 1-2, and hybrid transducer type [24, 27, 35] shown in Fig 1-3. Hybrid type ultrasonic motors combine torsional dispalcement piezo stacks and longitudinal displacement piezo stacks in a spark plug like compact shape to produce rotary motion. These references are concerned with rotary motors for consumer electronic appliances. Of course, linear ultrasonic motors have been reported, and one [37] is shown in Fig 1-4. This motor utilizes

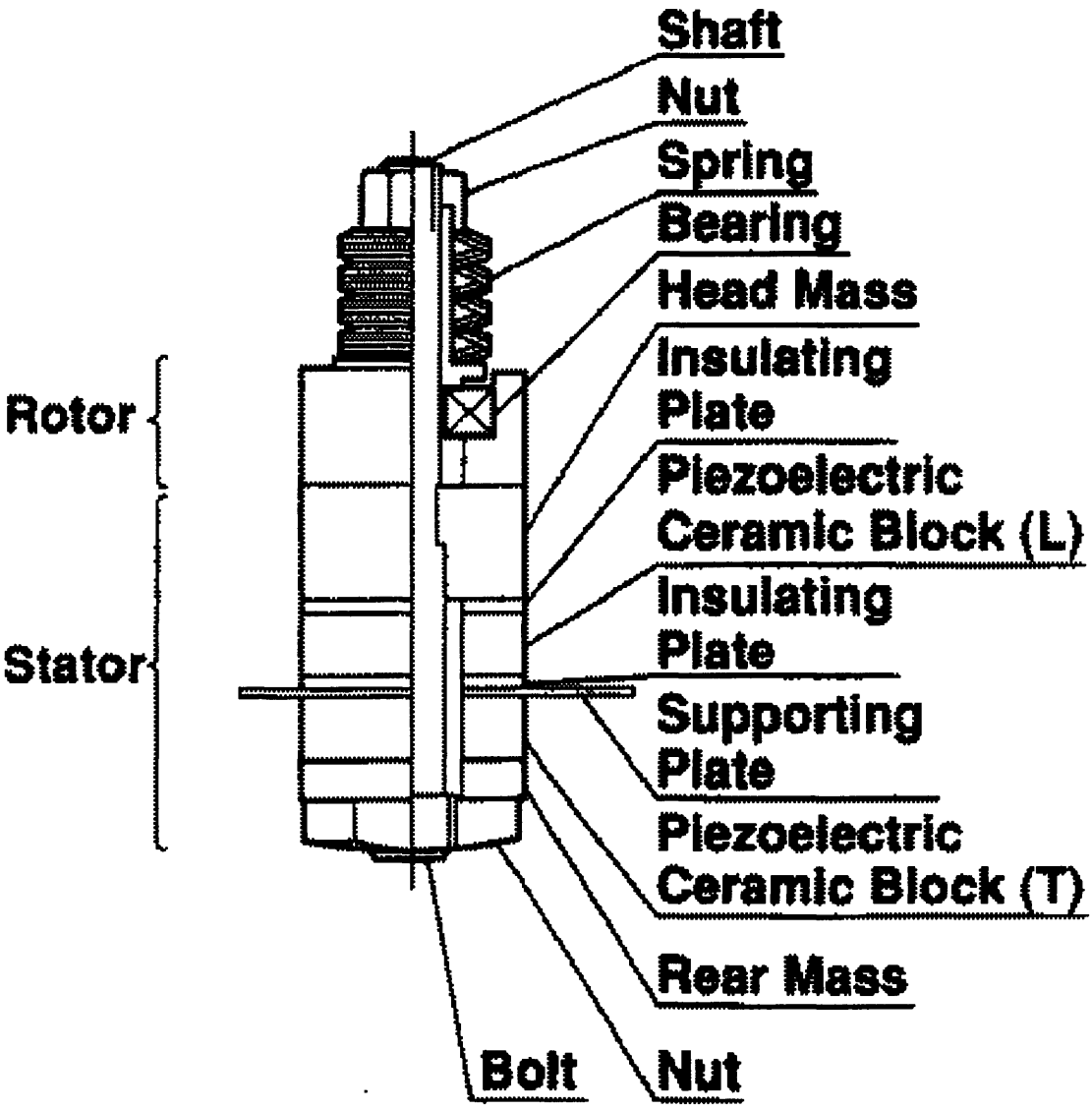


Figure 1-3: A hybrid rotary ultrasonic motor

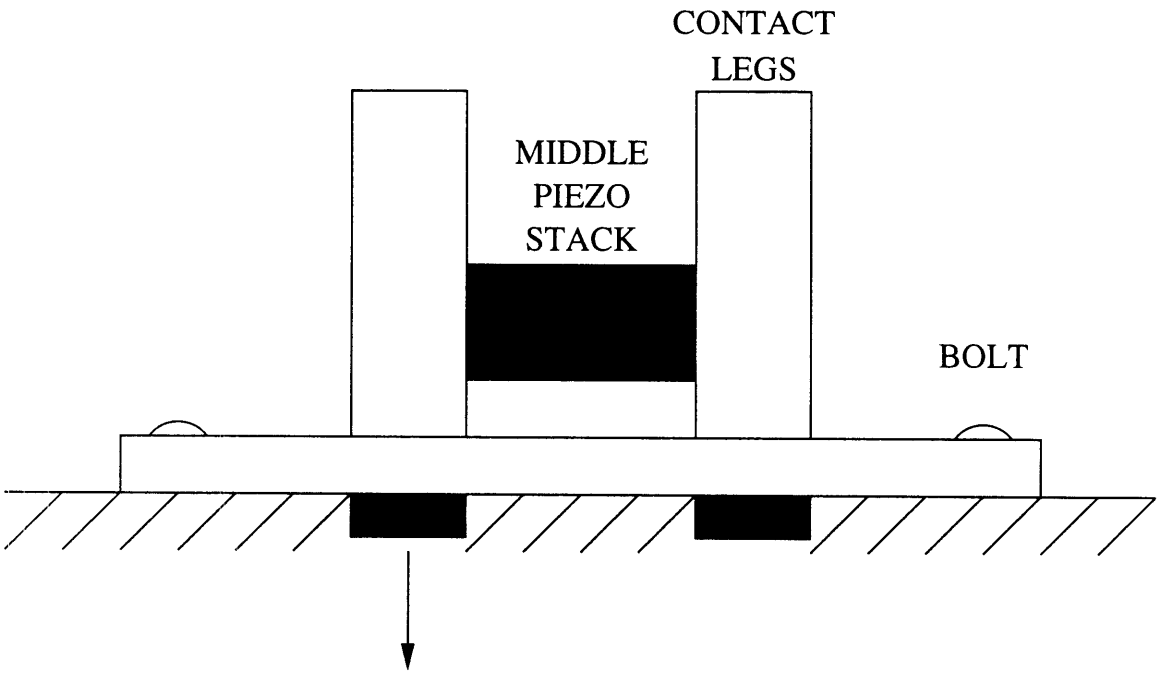


Figure 1-4: A linear ultrasonic motor designed by Toyoda and Murano

two distinct vibrations: that of the two cantilevered legs actually contacting the slider and creating the sideways motion, and the elastic material sitting on top of piezo ceramic, whose bending vibration controls the contact between the motor and the slider. As is discussed in the reference [37], two such distinct vibration mehanisms necessitates precise matching of the two vibration frequency for efficient motor operation. Unfortunately, natural frequency of the two modes are strongly dependent on friction, preload, and loading condition, rendering precise and robust frequency matching impractically difficult in practical situations. Furthermore, this motor is incapable of sub-nanometer accuracy, which is an absolutely critical requirement for a nano-actuator. To be sure, a nanometric resolution ultrasonic stepper motor, whose configuration is shown in Fig 1-4, was reported [31]. Clearly, this design is very complicated. The range of motion on this device is limited by the finite length of the two pairs of bar the piezo stacks push against, so that the range of motion is relatively small. These two factors prevent ready use of this nanometric stepper in practical applications.

Despite the apparent lack of concentrated effort to develop practical ultrasonic motors as nano-actuators, ultrasonic motors are ideally suited for nano-actuation. Not surprisingly, the chief reasons are directly derived from the beneficent properties of piezoelectric material: wide bandwidth, sub-nanometer accuracy, high energy density, and long range among others. Furthermore, with a clever design, high static stiffness, "infinite" travel range and super high speed long travel on the order of $1\frac{m}{s}$ are possible [37]. Though other nano-actuators may match the performance of the ultrasonic motor design proposed here in one or two selected areas, few if any satisfy all around requirements for a nano-actuator as well as this ultrasonic motor does.

1.3 Project description

Recognizing the opportunity to develop a nano-actuator based on the ultrasonic motor principles, we designed and constructed a prototype linear ultrasonic motor with all the necessary supporting experimental setup described in Chapter 2 that includes real time digital control kernel running on a host PC and a high performance digital signal processor (DSP) board jointly, to serve as a test platform not only just for this project but also for similar classes of systems to come in the future. The ultrasonic motor design was derived from the physics of the ultrasonic motor operation, which is established through analytical modeling of the ultrasonic motor principle in Chapter 3. The design successfully fulfills the functional requirements of a practical nano-actuator outlined in the design consideration in Chapter 2. The analytical model was verified with actual data collected during the motor performance test in Chapter 4. The design, construction, and experimentation methods developed here can serve as an effective example for other ideas and systems that require real-time experimental verification which would be too costly to try on a real system.

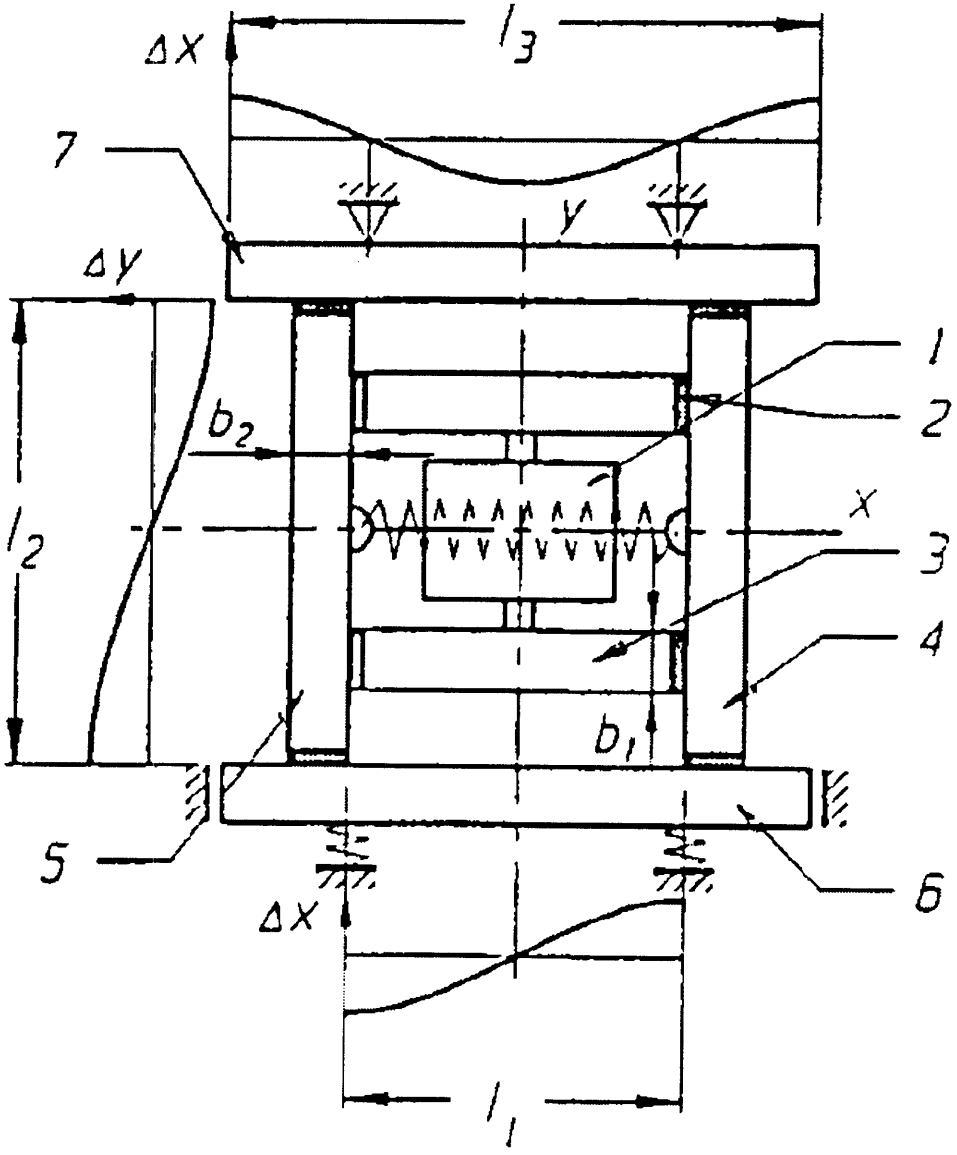


Figure 1-5: Nanometric two coordinate motor positioning stage

Chapter 2

Design and Construction of the Linear Ultrasonic Motor

The centerpiece of the project is verification of the principles behind the operation of ultrasonic motors and analysis of the ultrasonic motor performance for the purposes of assessing its potential as an effective actuator for nano-technology. In this chapter, we present a comprehensive approach to the ultrasonic motor design and control, starting with underlying design considerations, through each physical hardware and their interactions, including the real-time computer control kernel that is custom designed for digital controlled system such as this. The overall experimental setup consists of the ultrasonic motor assembly on top of a linear guide, as shown in Fig 2-1. In this experimental setup, the motor is held stationary and the surface of the linear guide is moved back and forth by the motor. This configuration was chosen because moving the motor with cables attached to it was cumbersome, and a position sensor could not be easily mounted on a small motor unit. While this is a possibility in an industrial setting, the opposite case is equally likely.

2.1 Design goal

Functional requirements for this ultrasonic motor experimental setup is to establish ultrasonic motor as a candidate for nano-technology actuator and to test its potential. To be considered a candidate for nano-actuation, the ultrasonic motor must prove nanometer or better resolution without prohibitively restricted range or speed. While some of the existing actuators such as piezo tube and inchworm motor are capable of nanometer resolution, no commercially available actuator successfully combines infinite range high speed motion

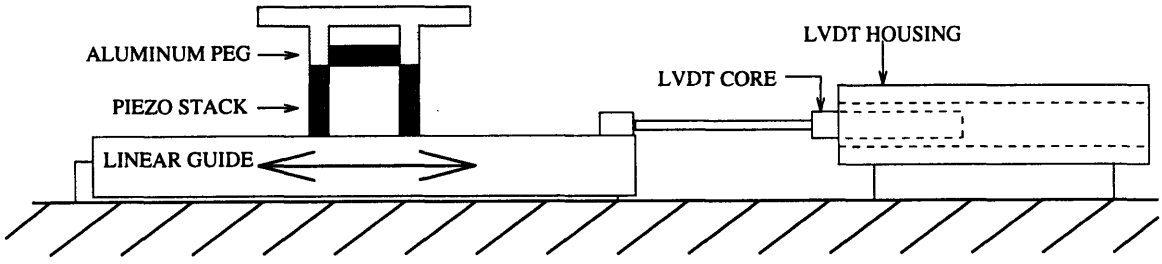


Figure 2-1: Ultrasonic motor experimental setup; the ultrasonic motor presses down on top of the linear guide, which moves the LVDT core in and out to give position signal

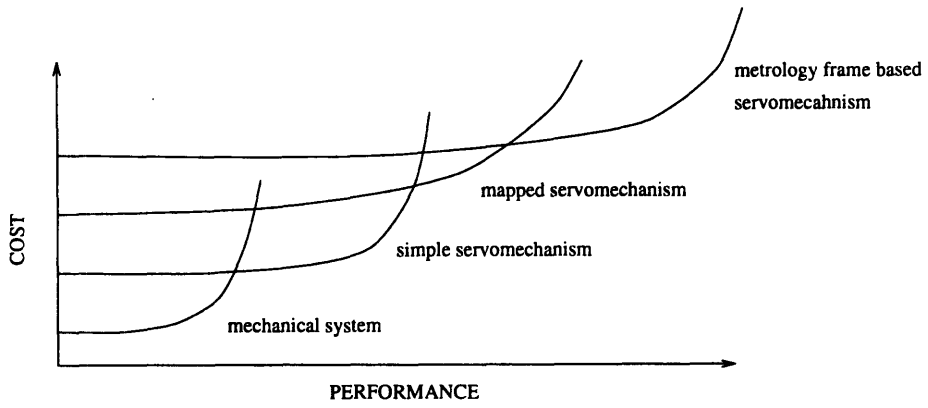


Figure 2-2: Technology-cost trend (Courtesy of Alex Slocum)

with nanometer accuracy for modest cost. Thus, the major thrust of this thesis is toward verification of the infinite range high speed motion combined with nanometer resolution capability using low cost components. In addition, since industrial applications such as wafer stepper are the primary target application, linear motion is preferred over rotary motion. Needless to say, linear and rotary motion are theoretically one and the same problem. And though it may take some adjustments for practical applications, conversion of the linear motion to rotary motion is not a fundamental problem. Indeed, majority of the ultrasonic motors reported in literature are rotary[16, 14, 15, 34, 19, 18, 24, 35, 36].

In all aspects of the design and especially at the immediate hardware level, simplicity was rigorously enforced. Simple design increases the chance of direct and robust control using metrology frame based servomechanism control at the immediate hardware level. The technology-cost trend curve in Fig 2-2 suggests that elaborate mechanical and servomechanical systems are giving way to digitally controlled mechanical system as performance requirement[1] is increased at an ever faster rate [29]. The graph captures the relationship between cost and performance requirement for increasingly advanced systems. A servomechanism is an open loop controlled mechanical system often found in machine tools, for example. When the dynamics of the actuator and the mechanical system is mapped and compensated for by digital computers, a mapped servomechanism fit for high performance precision machines is born. But for the monument machines discussed earlier, nothing less than metrology frame based servomechanisms, where the actual motion of the tools and parts are measured with sensors for real-time control, will suffice. Accordingly, this thesis focuses on integration of simple yet effective electro-mechanical system with high speed, high performance real time digital control system, rather than an elaborate and complicated (and hence easily damaged) stand-alone unit.

Enforcing a simple and compact design yields an added benefit of modularity; by ganging a group of motors in an orthogonal geometric configuration as shown in Fig 2-3, it is possible to achieve two dimensional motion and increase the motor torque at the same time. This configuration was explored with an electromagnetic linear motor floating on a platten, developed in the Laboratory for Manufacturing and Productivity at MIT [9]. But unlike this

[1]Performance requirement can be loosely defined as a combination of environment, load, range, speed, and accuracy requirement.

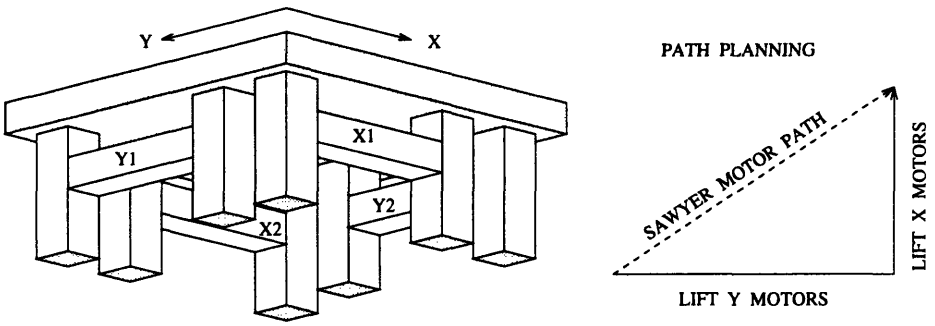


Figure 2-3: 3D isometric view from below of the 2D linear ultrasonic motor made by arranging two pairs of 1D motor in an orthogonal direction. The drive path is rectangular and not diagonal.

2D Sawyer motor, it is inadvisable to drive the 2D ultrasonic motor in a diagonal direction, because while the sawyer motor maintains frictionless contact between the rotor and the stator with air bearing, the ultrasonic motor is based on direct friction contact; difference between the net velocity of the motor and each individual leg's free end results in slipping. Slipping reduces the motor efficiency and life, creates extra heat, produces contaminants that can spoil a clean room environment, and acts as an external disturbance. To avoid these problems, the 2D motor must always move in a rectangular path, lifting the pair of legs that drive in the orthogonal direction clear off the hardened table surface. The high speed and performance of the ultrasonic motor dissolves the problem with having to move in a rectangular path.

2.2 Ultrasonic motor

As shown in Fig 2-1, the motor is simply three piezo stacks[2] [33] bonded to a pair of aluminum pegs, which were custom machined. This simple shape leads to a compact yet strong motor. The core motor assembly fits into a space of approximately 1 in^3 (16 cm^3). This is even a conservative estimate however, because the current prototype has long and slender piezo stacks which take up most of the space. Future design iterations may employ more compact piezo stacks for smaller motor size. Incidentally, the long and slender piezo stacks were not the original design choice; a much shorter and thicker piezo stack called for by the original design was not commercially available during construction, and the next available size was chosen. An unfortunate and ironical consequence is that the motor is not truly ultrasonic, which means vibration beyond audible frequency. The original design had a natural frequency of about 20kHz, but since the legs grew much longer and slender, the natural frequency went way down to under 4kHz, resulting in an *ultra sonic* motor.

The piezo stacks were carefully bonded to the aluminum pegs with super glue. Although it seemed unimportant initially, the choice of bonding material affected the robustness and the shock survivability of the motor. The quick dry low viscosity plastic cement proved convenient but weak, as the piezo stack easily snaps off from the peg at moderate bending

[2]Piezo stacks are NLA5×5×18 mm^3 manufactured by the Tokin Corporation, 155 Nicholson Lane, San Jose CA 95134, (408) 432-9020.

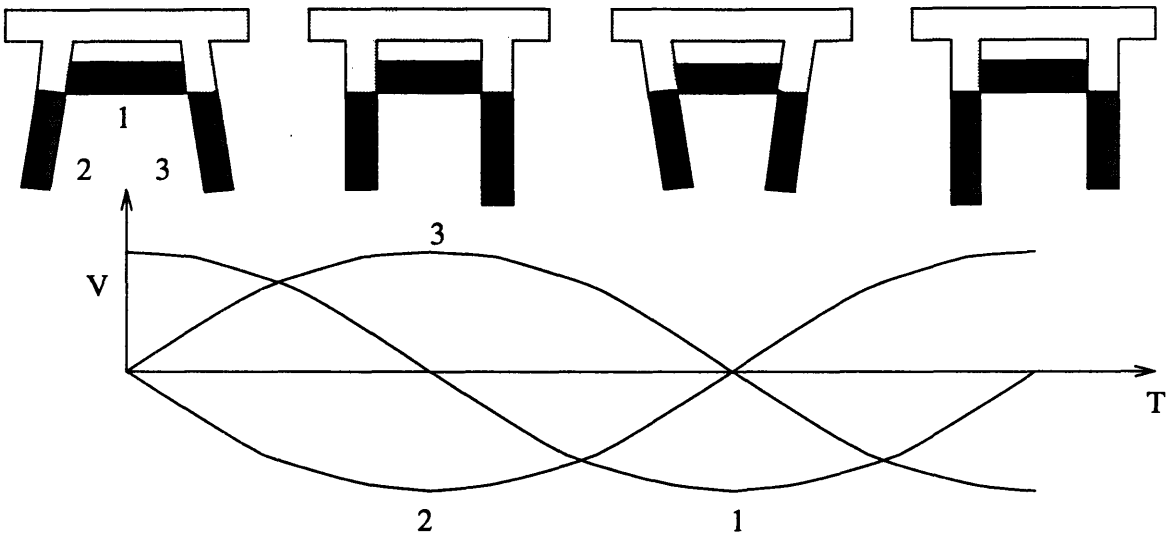


Figure 2-4: One cycle of the ultrasonic motor walk showing the voltage applied to each piezo stacks

loads. Since ultrasonic motors are all based on high frequency vibration and subject to extremely high torque to weight ratio during operation, a stronger bonding material and method must be explored in future for industrial applications. In parallel, since the principle of ultrasonic motor is also based on friction, a study into material for the friction layer between the motor and the linear guide surface should be conducted. If the target application is in the clean room environment, as in semiconductor fabrication settings, future study must consider bonding and friction material and the hardened table surface on which the motor will act that are fit for a class 1 clean room. For this prototype operating in normal room air, Neoprene rubber[3] was coated on to increase friction between the motor and the anodized aluminum linear guide surface, following the advice given in an empirical study of friction material by Endo and Sasaki [11].

The piezo stack bonded to the free end of a cantilevered aluminum peg attached to the solid aluminum base and in between the pegs expands by an amount roughly proportional to the voltage difference applied to its leads in the static case with no extra load on it. Therefore, a linear constitutive piezoelectricity relationship $x = dV$, where x is the displacement, d is the piezoelectric displacement constant, and V is the applied voltage to the electrodes is assumed throughout this thesis. (Nonlinear effects such as saturation, creep and hysteresis are discussed in Chapter 4.) And since negative voltage is never applied lest the piezo stack be depolarized (in fact, the manufacturer even took the pain of differenciating the ground lead and the positive voltage lead of the piezo stack out of this fear) and the piezoelectricity destroyed [7, 17], the piezo stack is at its minimum length when no voltage is applied to it. Using this principle, three modes of operation were explored in this thesis: high speed mode, stepper mode, and finally the nanometer mode.

The operation of the motor during high speed movement is much like the human walking motion, and is depicted in Fig 2-4. If two sinusoidal voltages offset by the reference voltage in amplitude and half a period (180°) in phase are injected to the two legs, they will expand

[3]Gacoflex N-1700 manufactured by Gaco Western, Inc., PO Box 88698, Seattle WA 98138-2698, (206) 575-0450.

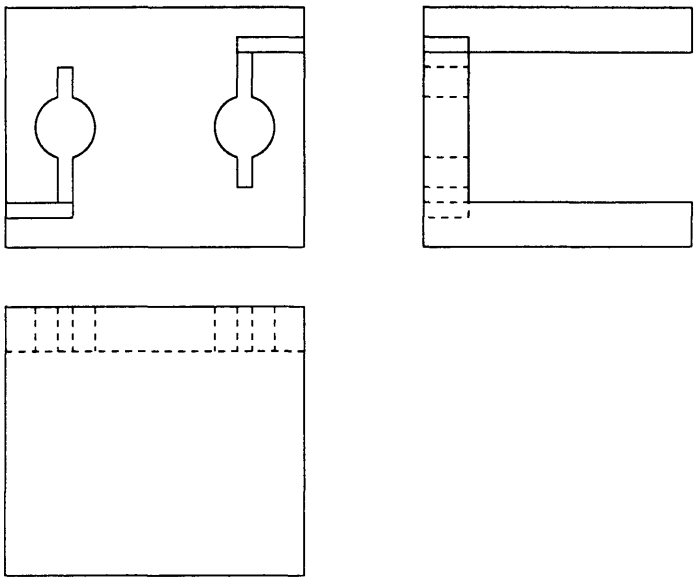


Figure 2-5: Engineering drawing of the ultrasonic motor clamp

and contract alternately. If in addition another sinusoid offset by a quarter of a period (90°) from the sinusoids to the legs is injected to the middle piezo, the walking motion shown in the figure is possible. Offsetting the phase by another half of a period (180°) effectively reverses the direction of the motor. Stepper mode is a straightforward extension of this idea; instead of series of smooth walking movement, only one cycle is applied. The voltage may be a square wave, instead of sinusoids, but the principle is the same. The nanometer precision movement is similar but with one crucial difference: establishing and losing friction contact between the leg and the linear guide surface by expanding and contracting the leg is anything but a nanometer precision type of movement; precision will be lost if the friction contact point changes during operation. Naturally, the solution is to get near the desired position quickly with high speed walking movement, and then maintain only one leg contact during the nanometer precision movement. The results are given in Chapter 4.

2.3 Ultrasonic motor clamp

In this design, the motor is fixed in place and moves the linear guide back and forth. As will be shown in the discussion of experimental results, the downward preload against the linear guide surface is crucial for maintaining optimal friction force while the motor leg is contacting the linear guide surface. The clamp for the ultrasonic motor, whose picture is shown in Fig 2-5, holds it rigidly to the ground to provide the necessary preload force. Half inch thick solid aluminum structure makes the clamp very stiff against excitation from the motor vibration. The double split clamp in the middle tightly hold a pair of aluminum cylinders bolted to the motor, maintaining a constant preload force while the motor is running, but allows the cylinders to slide freely for preload adjustment with the split clamp loose.

2.4 Sensor

Sensor feedback is indispensable not only for closed loop feedback control, but also for open loop performance analysis. Since the project goals are long range high speed movement *and* nanometer precision together, two types of sensors were used in each case to measure the position of the ultrasonic motor with respect to the linear guide. This problem partly delineates the fundamental difficulty of the project: pursuing nanometer precision and long range high speed motion is like chasing after two rabbits running away in opposite directions. That is also why one single affordable sensor to measure long range position with nanometer accuracy was not available. This problem is known as duality, and most people simply accept this and work around it through compromise and trade off. In designing this actuator, the two conflicting goals were satisfied at the same time only by employing dual mode operation; the fundamental problem is by no means completely defeated, and is still lurking out there. With this modicum of philosophy stated, technical details of the sensors used in the experiment now ensue.

For position sensing during the high speed movement, an LVDT[4] (Linear Voltage Differential Transducer) was used. The core rod mounted on the linear guide slides in and out of the LVDT housing which is fixed to the ground, as shown in Fig 2-1. The full range of the LVDT is 1 inch, corresponding to $\pm$10V DC voltage output from the LVDT housing. Since we use a 12 bit A/D converter to read the LVDT position signal, this LVDT setup has a quantizing resolution of

$$\frac{25.4mm}{20V}\frac{20V}{2^{12}} = 6.2\mu m$$

And with the full scale static linearity of 0.25% of full range and the 3 dB bandwidth of 500 Hz [21], it proved adequate for course resolution position sensing.

The raw position signal may be differentiated to approximate the velocity, instead of using a separate velocity sensor. The importance of reliable real-time velocity measurement for a metrology frame based servomechanical system, especially for a high speed and performance system such as this, needs no further emphasis. The choice between installing a separate velocity sensor or digital filter based approximation involves a careful design compromise weighing performance requirement against project constraints that include budget and size. If the design constraints allow the use of velocity sensor, a true metrology frame based full state feedback digitally controlled servomechanical system can be built. On the other hand, a causal digital filter, otherwise known as an observer, may be implemented to estimate the velocity from the position signal. This online estimate can never replace the actual measurement however, since there is always an inherent time delay associated with causal filtering and differentiation of quantized position signal induced noise that worsens as the sampling rate increases. For full state feedback, the indicated issue becomes critical. In this study, performance assenssmentof the open loop system was of primary interest, so the raw position signal from LVDT could be differentiated offline with acausal velocity filter without any performance loss. Fig 2-6 compares the power spectral density and time response of the velocity signal, which was obtained by raw differentiation of the position signal, before and after acausal low pass butterworth filter. Because the motor is vibrating all the time in the high speed mode and the experiment was conducted in a normal

[4]Model 100 DC-D manufactured by Lucas Control Systems Products, 1000 Lucas Way, Hampton VA 23666, (804) 766-4494.

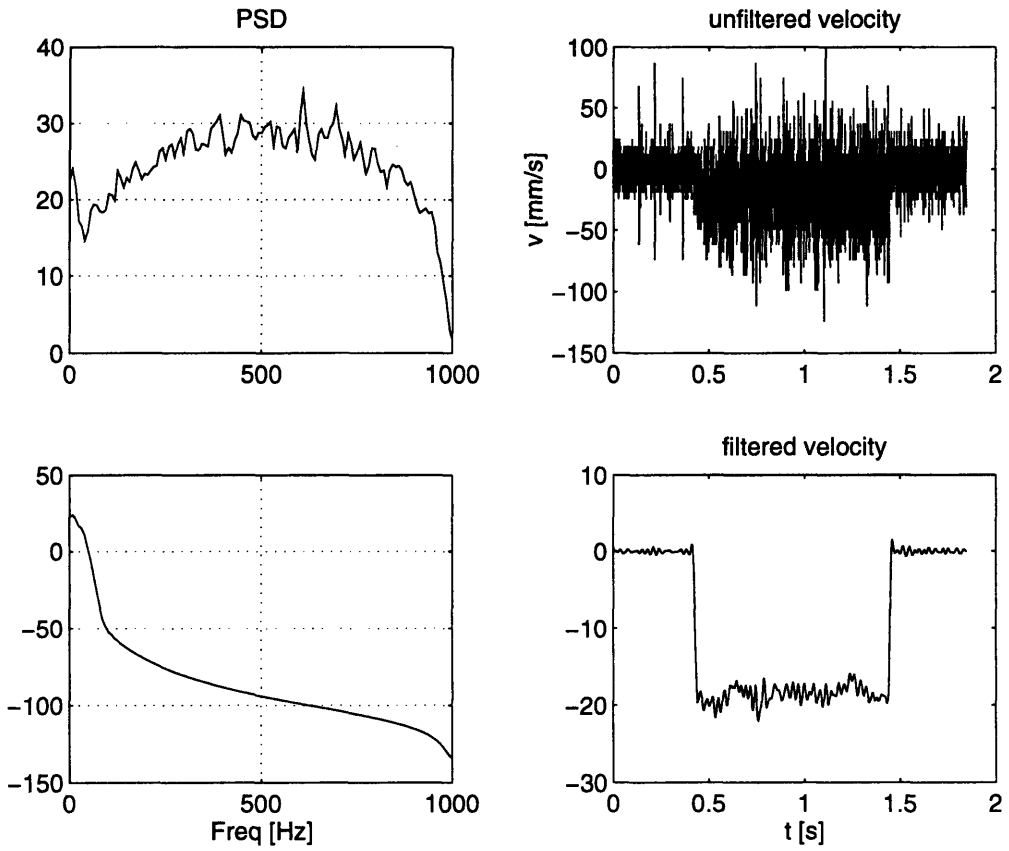


Figure 2-6: Power spectral density and time response of the velocity signal before and after the acausal low pass filter

room atmosphere on a passive vibration isolation table, the raw position signal contains high powered noise. When this noisy position signal is differentiated, the noise in the raw velocity signal becomes even more significant due to high sampling rate of 2kHz used in this experiment. So a high order low pass digital Butterworth filter with 40Hz cutoff frequency was used to attenuate high frequency noise. Ideally, a much lower cutoff is desirable since the true velocity is almost a DC signal, but as was discussed in Chapter 1 and will be demonstrated in Chapter 4, ultrasonic motors have very high bandwidth; too low of a cutoff frequency will distort the fast dynamic response of the motor.

For high precision movement, we aimed at nanometer or better resolution to support the nanometer resolution claim. A parallel plate capacitive probe[5] with a static linearity of $\pm 0.2\%$ of full range and better than 5kHz bandwidth [22] met the accuracy and robustness requirement with some modification: the probe had 25 μm range and 25 nm resolution initially–inadequate for this experiment. Fortunately, the amplifier output was 0 to 10V DC analog voltage, and the resolution could be improved by sacrificing range. The raw signal from the AS-1021-SAI amplifier unit connected to the capacitive probe was first filtered through a low pass RC filter with a cutoff frequency of 24 Hz to attenuate high frequency noise (Such low cutoff frequency was acceptable only because fast dynamics during high precision movement was not considered in this experiment.), and then amplified 25 times through a non-inverting op-amp, yielding 1 nm resolution before it is converted by a 12-bit A/D.

2.5 Computer interface and electrical connection

The experiment is interfaced to an 80486 computer running an MS-DOS based real-time kernel called Sparrow[6] [23], running in parallel with a custom developed DSP board [13] based on Analog Device's ADSP 21020 DSP chip [1]. The computer interface and electrical connection schematic for the experiment is given in Fig 2-7. Detailed wiring information is given in Appendix A. Normally, Sparrow is self sufficient as a real-time operating kernel, with its own servo scheduling, channel interface, and textual screen display; to control a physical system in real-time, a user simply plugs in appropriate I/O board for which a device driver is already written, builds a simple textual display control screen, writes a well defined timer interrupt driven servo routine in C program language, compile the code and link it to the Sparrow run time library with Borland C Compiler [5], and finally execute it on an 80286 or higher model computer. This procedure is less daunting than having to write a whole new real-time operating kernel from scratch, including the device driver and the user interface. Several commercially available real-time control kernels[7] serve essentially the same function for a lot more money and complexity, and with less flexibility for code modification and expansion. It is with these respects that Sparrow holds an edge over more sophisticated commercial products.

Ultra fast D/A converters were required in this experiment to inject three channels of sinusoids of frequency as high as 5kHz to drive the ultrasonic motor during the high speed movement. Unfortunately, due to large overhead inherent in any normal C programs, the fastest sampling rate Sparrow could provide was 2kHz with all the servo loop time devoted

[5]Probe model ASP-1 with probe signal amplifier model AS-1021-SAI, both manufactured by Mechanical Technology, Inc., 968 Albany-Shaker Road, Latham NY 12110, (518) 785-2211.

[6]Sparrow is free and publicly available via anonymous ftp from `avalon.caltech.edu`.

[7]Lab View, LynxOS, etc.

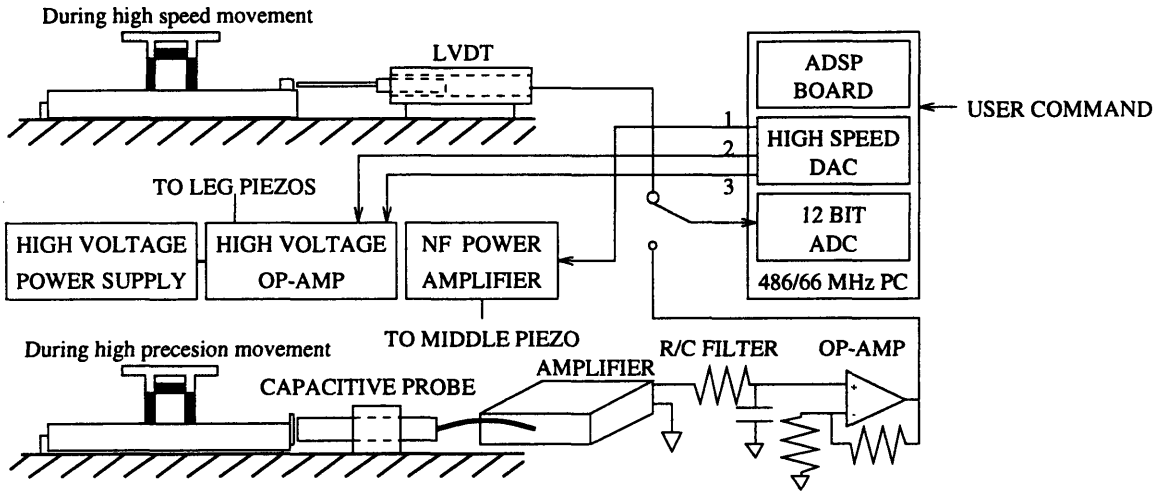


Figure 2-7: Computer interface and electrical connection schematic

to sampling alone. Furthermore, since the ADSP board already came with its own high speed D/A converter, it was decided early on in the project to utilize the ADSP board for all intensive calculations and time critical analog conversions, alleviating the demand on the host PC processor. The host PC then administers all other less time critical tasks such as user interface, data capturing and dumping, real-time data plotting, and of course monitoring the status of the ADSP board. As far as Sparrow is concerned, the ADSP board is treated like another I/O board, albeit more complicated, and a device driver, once written, almost frees the user from having to keep track of the communications protocol between the PC and the ADSP board–but not entirely. The user must still instruct the ADSP board its commanded action in all contingencies, and designate the memory map. Contrary to a popular misconception, DSP is not a well evolved user friendly micro processor; it is just the opposite. It is a hyper rapidly evolving technology that was in an embryonic state until fairly recently. It derives its fantastic performance largely by being ruthlessly simple. For example, the ADSP does not even recognize the `double` data type, which is almost a given in mathematical programming, but shows a blinding performance increase over PC processor in a well defined manipulation of simpler data types. It is with this grain of salt that the high computational power of the DSP was utilized. The net effect is that the ADSP can used used to sample the experiment at whopping 100 kHz or better while the host PC monitors the ADSP board and the user input at a much more benign speed of 2 kHz. In summary, the procedure for developing a Sparrow program to run in conjunction with ADSP board and the high speed A/D and D/A is[8]:

1. Code the ADSP instruction (ADSP can be programmed either in C or ADSP assembly. Choose ADSP assembly language if speed is critical.) and memory map, compile with appropriate compiler, and link with runtime library if necessary to generate an executable.

2. Convert the executable to STK (byte stacked format) file that can be read into the ADSP's program memory.

[8]Refer to the reference manuals specified above for technical terminologies.

3. Design the display screen and write Sparrow program, specifying the name of the file (in STK format) containing the slave ADSP program to run in parallel.

4. Run the Sparrow executable program, from which instructions to load specified ADSP program into the ADSP board program memory and start executing it upon user signal are issued.

Once the software and all individual parts of the experiment are prepared, connecting them with appropriate support electronics is straightforward, and real time data collection and analysis can begin.

Chapter 3

Analytical Modeling of the Ultrasonic Motor

In this chapter, we develop a mathematical model of the system which can be used for design evaluation, performance prediction, control design and finally offline simulation.

As stated in the Introduction, ultrasonic motor operation is based on conversion of the stator vibration into the rotor motion. For our linear ultrasonic motor, the stator is a pair of piezo stack with an aluminum peg combination, and the rotor is a linear guide. Therefore, the mathematical modeling of the piezo stack-aluminum peg combination is vital for analytical modeling of the motor. The structure under investigation can be modeled as a simple cantilevered beam with different material properties at each end of the beam, as shown in Fig 3-1. If the inhomogeneous material properties are temporarily ignored and only the transverse displacement of a homogeneous beam is considered, the relatively simple geometry of the structure readily lends itself to analysis using the Timochenko beam theory. The derivation is presented here for completeness.

Using Hamilton's principle [8], it is straightforward to verify that the governing equation for the bending vibration of the stator peg idealized as a beam in Fig 3-1 is:

$$\rho A \frac{\partial^2 u}{\partial t^2} + EI \frac{\partial^4 u}{\partial x^4} = f(x,t) \tag{3.1}$$

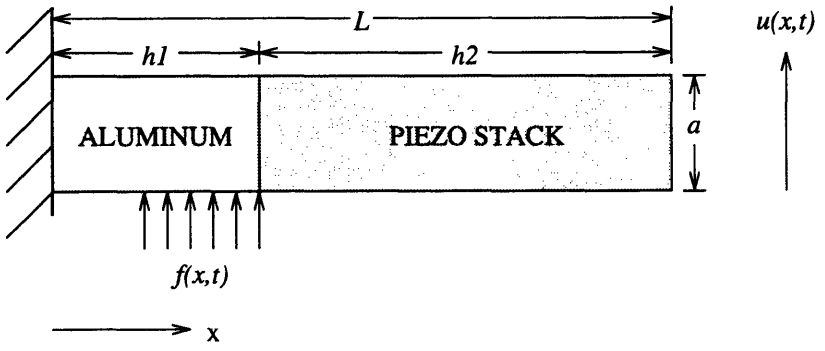


Figure 3-1: Idealization of the piezo stack and aluminum peg combination as a simple cantilevered beam

	Constant		piezo stack	aluminum peg
E	Young's modulus	$\left[\frac{kN}{mm^2}\right]$	44	70
ρ	density	$\left[\frac{g}{cm^3}\right]$	10.0	2.8
h	beam length	$[mm]$	18	9
w	beam width	$[mm]$	5	5
a	beam thickness	$[mm]$	5	5
I	polar moment of inertia	$[mm^4]$	52.08	52.08

Table 3.1: Table of material constants

where $u(x,t)$ is the transverse displacement of a position on the beam and $f(x,t)$ is the external force on the beam. Numerical values for the relevant constants are given in Table 3.1. As is the case in classical beam vibration problems, the boundary condition dictates the actual mode shapes. For this problem, fixed end condition at the wall and the free end condition at the other end dictate the following boundary conditions:

$$\begin{aligned} u(0,t) &= 0 \\ \frac{\partial u(0,t)}{\partial x} &= 0 \\ \frac{\partial^2 u(L,t)}{\partial x^2} &= 0 \\ \frac{\partial^3 u(L,t)}{\partial x^3} &= 0. \end{aligned} \tag{3.2}$$

Once the governing equation and the boundary conditions are established, the mode shapes can be derived by considering the free vibration first, by setting $f(x,t) = 0$, and using the separation of variable technique to let $u(x,t) = a(x)\sin(\omega t)$. This assumption implies that the solution will be harmonic oscillation of mode shapes in time. When substituted into the governing equation (Eqn(3.1), with no forcing term, of course), the time harmonic functions cancel and an ordinary differential equation for mode shape results:

$$-w^2\rho A a(x) + EI\frac{d^4a(x)}{dx^4} = 0, \tag{3.3}$$

whose general solution is

$$\begin{aligned} a(x) &= \alpha_1\sin(kx) + \alpha_2\cos(kx) + \alpha_3\sinh(kx) + \alpha_4\cosh(kx), \\ k^4 &= \frac{\rho A}{EI}w^2, \end{aligned} \tag{3.4}$$

where α_i are constant coefficients. Solving the ordinary differential equation governing the mode shape (Eqn(3.3)) subject to the given boundary condition (Eqn(3.2)) leads to a transcendental frequency equation

$$\cos(kL)\cosh(kL) = -1, \tag{3.5}$$

which is satisfied only at discrete values of k. The sequence of discrete values k_n, which are

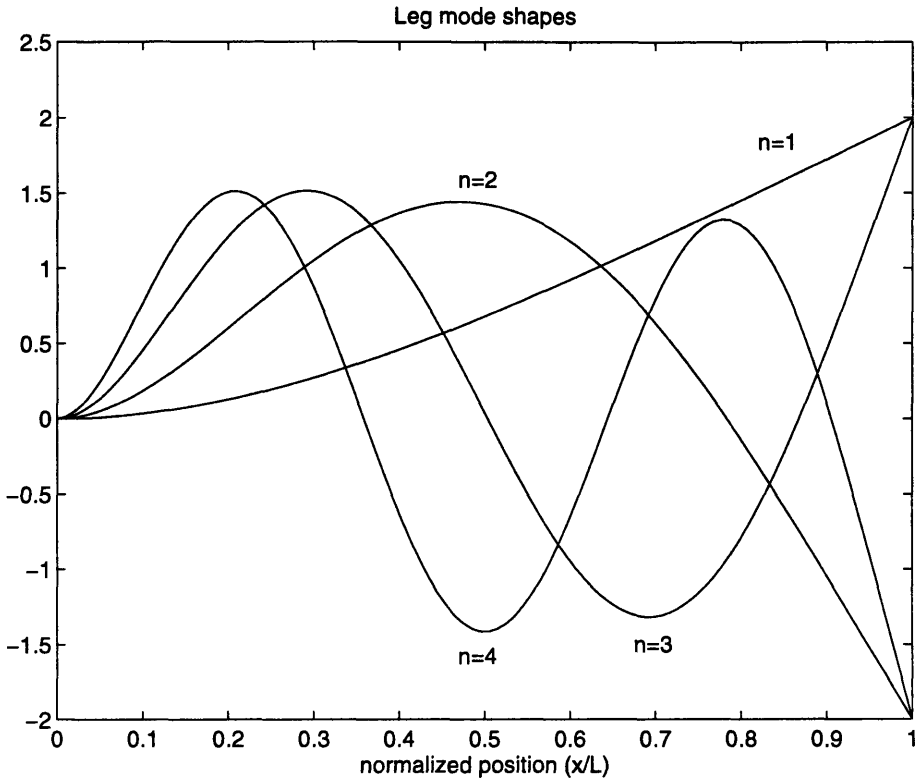


Figure 3-2: The first four bending modes shapes of the linear ultrasonic motor leg

the roots of Eqn (3.5) are the wave numbers for the n^{th} mode with the corresponding mode shapes

$$\phi_n(x) = c_n \left[\cosh(k_n x) - \cos(k_n x) - \frac{\cosh(k_n L) + \cos(k_n L)}{\sinh(k_n L) + \sin(k_n L)} \left(\sinh(k_n x) - \sin(k_n x)\right)\right]. \quad (3.6)$$

Since a closed form expression for k_n that satisfy Eqn(3.5) does not exist, we resort to an approximation by graphical means given in Norton [25]. As Fig 3-2 illustrates, the number of nodes increases linearly with the wave numbers.

Guided by physical insight, we now define two operators of key interest, $M[\cdot]$ and $K[\cdot]$, to clarify the mode shape idea:

$$\begin{aligned} M[f(x)] &\equiv \rho A f(x) \\ K[f(x)] &\equiv EI \frac{\partial^4 f(x)}{\partial x^4}. \end{aligned}$$

These two operators have self-adjoint property [8] with respect to the inner product defined by

$$\langle f(x), g(x) \rangle \equiv \int_0^L f(x) g(x) dx, \quad (3.7)$$

where $f(x)$ and $g(x)$ are arbitrary functions. Self-adjointness is a powerful property that characterizes energy conserving linear systems. Symmetry and reciprocity principles are

mode number n	wave number k	modal stiffness K	modal frequency ω
1	$\frac{1.875}{L}$	$12.36\frac{EI}{L^3}$	$3.52\sqrt{\frac{EI}{ML^3}}$
2	$\frac{4.694}{L}$	$485.45\frac{EI}{L^3}$	$22.034\sqrt{\frac{EI}{ML^3}}$
3	$\frac{4.855}{L}$	$3807.37\frac{EI}{L^3}$	$61.701\sqrt{\frac{EI}{ML^3}}$
4	$\frac{10.995}{L}$	$14612.23\frac{EI}{L^3}$	$120.89\sqrt{\frac{EI}{ML^3}}$

Table 3.2: First four modal stiffness and frequency

some of the direct consequences of self-adjointness. Showing that M is self-adjoint is trivial. K can be shown to be self-adjoint using integration by parts with the said boundary condition. In short,

$$\begin{aligned} \langle \phi_i(x), M[\phi_j(x)] \rangle &= \langle M[\phi_i(x)], \phi_j(x) \rangle \\ \langle \phi_i(x), K[\phi_j(x)] \rangle &= \langle K[\phi_i(x)], \phi_j(x) \rangle . \end{aligned}$$

Furthermore, the mode shapes form an orthogonal basis for all amplitude of harmonic vibration of the beam, *i.e.* ,

$$\begin{aligned} \langle \phi_i(x), M[\phi_j(x)] \rangle &= M_i \delta_{ij} \\ \langle \phi_i(x), K[\phi_j(x)] \rangle &= K_i \delta_{ij}, \end{aligned}$$

where δ_{ij} is the Kronecker delta function, and M_i, K_i are the i^{th} modal mass and stiffness. With this simplification, the complex continuous beam vibration problem transforms into a simple mass-spring system for each mode. To further simplify the problem, we equate each modal mass with the static mass of the beam to reflecting the physical reality and explicitly impose the condition $M_n = \rho AL$.

$$M_n = \langle \phi_n, M[\phi_n] \rangle = \rho AL = M. \tag{3.8}$$

Numerical computation with MAPLE [12] shows that the first four scalar coefficients in Eqn(3.6) (c_1, c_2, c_3, c_4) that satisfy this constraint are all close to 1.0, and the resultant stiffness K_n's agree with predicted values in Table 3.2. The first four modal stiffness and frequency are tabulated in Table 3.2. Since this is a general derivation for this class of problems, the stiffness and frequency are parameterized by constants E, I, L, and M. For this experimental setup, $I = 52.08mm^3$, $L = 27mm$, $M = 5.1g$. The correct value for E is not clear, however, because of the inhomogeneous composition of the beam. If the leg is assumed to be a homogeneous aluminum beam, $E = 70\frac{kN}{mm^2}$, with the corresponding first modal stiffness and natural frequency of $K_1 = 2289\frac{N}{mm}$, and $f_1 = 3362$ Hz. On the other hand, if the leg were made only with piezo stack, $E = 44\frac{kN}{mm^2}$, $K_1 = 1439\frac{N}{mm}$ and $f_1 = 2673$ Hz–lower than those for aluminum, since the piezo stack is heavier and less stiff than aluminum. The actual figures lie between the two extreme cases, but closer to those for aluminum. Most of the stress is concentrated near the fixed end in the aluminum peg, and consequently, the stiffness of the aluminum provides the dominant spring force and is more relevant than that for the piezo stack. For a more accurate model, a Finite Element Analysis program like ABAQUS can easily calculates not only the first but higher modes

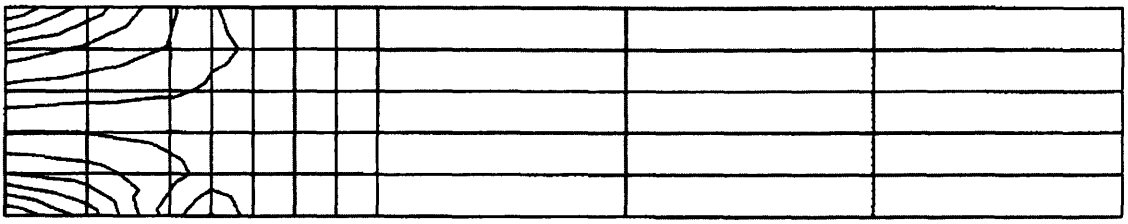


Figure 3-3: Static stress concentration in the leg. Almost all the stress is taken up by the aluminum end of the beam.

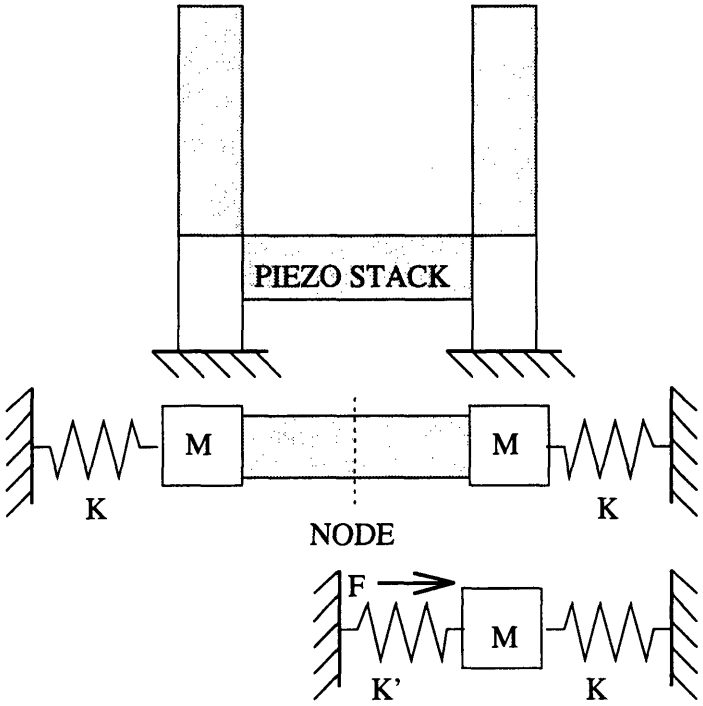


Figure 3-4: Lumped parameter model of the ultrasonic motor

as well. The FEA program predicted the first modal frequency of 3254 Hz, confirming the physical intuition. Fig 3-3 shows the static stress concentration in the leg calculated with ABAQUS. The source code for this FEA program is given in the Appendix B.

So far, only the unforced free vibration has been analyzed. But in practice, the leg is forced on one side just below the piezo stack by another identical piezo stack, indicated by $f(x,t)$ in Fig 3-1. After the two legs have been idealized as a pair of mass-spring system using modal decomposition, the next step is modeling the external forcing function applied by the piezo stack between the legs. As depicted by the lumped parameter model in Fig 3-4, structural symmetry greatly simplifies the model. Since the node at the center of the middle piezo stack is functionally equivalent to the ground due to symmetry, the half section of the piezo stack now acts like another spring connecting the lumped leg mass to the ground. Of course, assuming there is only one node in the piezo stack constrains the operating frequency of the motor to sufficiently below the first modal frequency of the middle piezo stack at 75kHz [33], for otherwise the inertia of the piezo stack itself starts to play a role and the piezo stack dynamics must be considered. (Bending vibration modes of the leg and axial

vibration modes of the piezo stack by itself have no correlation and must not be confused with each other.) However, the motor operates below the first modal frequency of the *leg* at 3kHz–far below the self resonance frequency of the piezo stack–making this constraint but a passing concern. As for the external forcing term F in Fig 3-4, a voltage source drives the piezo stack, producing a force on the mass that is proportional to the applied voltage. We use linear time invariant constitutive relationship to model the piezoelectricity effect: $F(t) = \eta V(t)$, where η is the piezoelectric force constant and $V(t)$ is the voltage applied to the piezo stack. Since this piezo stack with a cross sectional area of $25mm^2$ produces 87 $\frac{kgf}{cm^2}$ pressure under the maximum operation voltage of 100V, the piezoelectric force constant is

$$\eta = 87 \cdot 9.86 \frac{N}{cm^2} \frac{0.25cm^2}{100V} = 2.14 \frac{N}{V}.$$

At this point, only the spring constants K and K' are left for discussion from the lumped parameter model of Fig 3-4, and the modal decomposition method is invoked once again to find the final piece of the puzzle. We treat forced vibration as a natural extension of the free vibration to allow the forcing term $f(x,t)$ back into the governing equation (Eqn(3.1)) and write the general solution as an infinite sum of normal modes[1]:

$$u(x,t) = \sum_{i=1}^{\infty} \phi_i(x)\tau_i(t), \tag{3.9}$$

where the modal amplitude participation factors $\tau_i(t)$ are analogous to the Fourier transform coefficients [28]. Consistent with the separation of variable technique used to derive the modeshapes, we do the same for the forcing function, *i.e.* $f(x,t) = F_x(x)F_t(t)$. When we substitute Eqn(3.9) into the governing equation and take the inner product defined by Eqn(3.7) on both sides with the n^{th} modeshape $\phi_n(x)$,

$$\begin{aligned}
\left\langle \phi_n, \sum_{i=1}^{\infty} (\rho A \phi_i \ddot{\tau}_i + EI\tau_i \frac{d^4\phi_i}{dx^4}) \right\rangle &= \langle \phi_n, F_x(x)F_t(t) \rangle \\
\sum_{i=1}^{\infty} \left(\langle \phi_n, M[\phi_i] \rangle \ddot{\tau}_i + \langle \phi_n, K[\phi_i] \rangle \right) \tau_i &= \langle \phi_n, F_x(x) \rangle F_t(t) \\
M\ddot{\tau}_n + K_n\tau_n &= \hat{f}_n F_t(t) \\
\ddot{\tau}_n + \omega_n^2 \tau_n &= \frac{1}{M} \hat{f}_n F_t(t),
\end{aligned} \tag{3.10}$$

where $\hat{f}_n$ is the generalized forcing shape. This result is valid for all forcing shapes, but the solution depends on the arbitrary time forcing function $F_t(t)$. But as mentioned in Chapter 2, the drive functions are sinusoids, meaning $F_t(t) = \sin(\omega t)$. In this case, the well known solution to the above second order ordinary differential equation is

$$\tau_n(t) = \frac{1}{M(w_n^2 - w^2)} \hat{f}_n \sin(\omega t). \tag{3.11}$$

From Figures 3-1 and 3-4, it is easy to show that the external force $f(x,t)$ on the leg

[1]This is a core idea of modal decomposition method, and can be supported rigorously with generalized Fourier transform.

(the M coupled to K and *not* K') is $F_x(x)\,(F - K'\nu)$, where

$$F_x(x) = \begin{cases} \frac{1}{a} & h_1 - a \le x < h_1 \\ 0 & \text{otherwise,} \end{cases}$$

F is the piezoelectric force $\eta V(t)$ mentioned earlier, and ν is the position of the lumped mass. This ν in turn is equal to

$$\nu = u(x', t) = \sum_{i=1}^{\infty} \phi_i(x')\tau_i(t),$$

x' being the point at which the piezo stack pushes against the leg. Of course, the piezo stack is not pushing at a point but rather over a finite area, specifically from 4 to 9 mm from the fixed end of the leg. Nevertheless, some averaging may be used to find the mean value for x' with this caveat noted. Incidentally, since x' is the average of $u(x,t)$ where $F_x(x)$ is nonzero, $\phi_n(x') = \hat{f}_n$. Finally, when

$$f(x,t) = F_x(x)\left(\eta V(t) - K' \sum_{i=1}^{\infty} \phi_i(x')\tau_i(t)\right)$$

is substituted into the governing equation (Eqn(3.1)) and the same steps as in Eqn(3.10) are carried out,

$$\begin{aligned} \left\langle \phi_n, \sum_{i=1}^{\infty} (\rho A \phi_i \ddot{\tau}_i + EI\tau_i \frac{d^4\phi_i}{dx^4}) \right\rangle &= \left\langle \phi_n, F_x(x)\left(\eta V(t) - K' \sum_{i=1}^{\infty} \phi_i(x')\tau_i(t)\right)\right\rangle \\ M\ddot{\tau}_n + K_n\tau_n &= \hat{f}_n\left(\eta V(t) - K' \sum_{i=1}^{\infty} \phi_i(x')\tau_i(t)\right) \\ M\ddot{\tau}_n + K_n\tau_n + \hat{f}_n K' \sum_{i=1}^{\infty} \phi_i(x')\tau_i(t) &= \hat{f}_n \eta V(t). \end{aligned} \tag{3.12}$$

Care must be taken to avoid confusing the index n with the piezoelectrical constant η.

Obtaining a closed form solution from Eqn(3.12) is not amenable because τ_i's are infinitely coupled–unless the coupling is very weak and negligible. This is indeed the case, for our operation frequency is well below the first modal frequency. In more concrete terms, Eqn(3.11) implies that τ_n decreases like $1/\omega_n^2$ for small ω and since ω_2 is more than six times greater than ω_1 according to Table 3.2, coupling between the first and the second mode is attenuated approximately forty times. Obviously, coupling between the first mode and higher mode is attenuated by an even greater amount. Based on this reasoning, if second or higher modes in Eqn(3.9) are neglected and $u(x,t)$ is approximated with the first mode only, *i.e.*

$$u(x,t) \cong \phi_1(x)\tau_1(t), \tag{3.13}$$

Eqn(3.12) can be uncoupled as follows:

$$\begin{aligned} M\ddot{\tau}_1 + \underbrace{(K_1 + \hat{f}_1^2 K')}_{K_{eff}}\tau_1 &\cong \hat{f}_1 \eta V(t) \\ \ddot{\tau}_1 + \omega_{sys}^2 \tau_1 &\cong \frac{\hat{f}_1 \eta}{M} V(t), \end{aligned} \tag{3.14}$$

where ω_{sys} is the natural frequency of the peg and the middle piezo stack put together as a whole, parameterized with $M, K_1, \hat{f}_1$, and K'. M and K_1 are already known, and numerical integration reveals that $\hat{f}_1 \cong 0.188$. The spring constant of the piezo stack K' can be found by measuring the free deflection of the piezo stack with no load on it; for free deflection, the piezoelectric force is opposed by the internal spring back force of the piezo stack, *i.e.* $\eta V = K' x_{free}$. Hence,

$$K' = \frac{\eta V}{x_{free}} = 2.14 \frac{N}{V} \frac{100V}{15\mu m} = 14.23 \frac{N}{\mu m}.$$

The free deflection of $15\mu m$ per 100 V is supplied by the piezo stack manufacturer. The end result is a slightly faster system with $f_{sys} = 3729$Hz, compared to $f_1 = 3362$Hz. Once again, the drive voltage is a pure sinusoid, $V(t) = V_0 \sin(\omega t)$ into Eqn(3.14), and the familiar second order solution is attained:

$$\tau_1(t) \cong \frac{\hat{f}_1 \eta}{M(\omega_{sys}^2 - \omega^2)} V_0 \sin(\omega t). \tag{3.15}$$

Note that the amplitude blows up when $\omega = \omega_{sys}$ because damping has been neglected.

The simplest model of internal friction forces during vibration is the viscous damping, in which the friction force is proportional to the velocity of the member, *i.e.*

$$f_d(x,t) = b \frac{\partial u(x,t)}{\partial t},$$

where b is the coefficient of damping per unit length. If this damping model is included, an additional term is added to the governing equation, so that

$$\rho A \frac{\partial^2 u}{\partial t^2} + b \frac{\partial u}{\partial t} + EI \frac{\partial^4 u}{\partial x^4} = f(x,t). \tag{3.16}$$

The damping term is now included, and we follow the same line of reasoning as the undamped case. First, we define the damping operator $B[f(x)] \equiv bf(x)$ and see that it is self-adjoint. Furthermore, inclusion of the damping term leaves the mode shape of Eqn(3.6) unaffected, and the n^{th} modal damping term is simply the damping coefficient, *i.e.* $B_n = b$, so that under the same operating conditions as in the undamped case,

$$\begin{aligned} \tau_n(t) &= \frac{\hat{f}_n}{M\sqrt{(\omega_n^2 - \omega^2)^2 + (\frac{b\omega}{M})^2}} \sin(\omega t - \varphi_n) \\ \varphi_n &= \arctan\left(\frac{b\omega}{M(\omega_n^2 - \omega^2)}\right). \end{aligned} \tag{3.17}$$

If the stator is driven at one of the natural frequencies, a large but finite peak in amplitude results. In practice, this is not a big concern, since the operating frequency is much lower than even the first modal frequency.

With the damping included, the dynamics of the ultrasonic motor is completely analyzed, which is to say that movement of any part of the motor at any given time is known. What are not implied here but patently clear from the steps taken during the derivation of the motor dynamics are associated qualifications for the above models to hold. Only a linear time invariant system with no external load forced by an ideal piezo stack with all

nonlinearities neglected was considered. But even before the importance of the neglected dynamics is discussed, the loading assumption must be examined. Firstly, the purpose of a motor is to drive some load, which may be unknown or changing in time. Furthermore, since the ultrasonic motor operation is based on friction, the effective load on the motor at a given instant depends on the preload and surface friction condition. The usual practice in modeling surface friction is to lump all friction characteristics into one parameter called the coefficient of friction μ, and assume that the friction force is proportional to the normal force N, *i.e.* $F_{friction} = \mu N$. The trouble is that this coefficient of friction is dependent on variables that are hard to control, such as temperature and humidity, so that no simple model can be given here. Nevertheless, the model of the motor under no load is still eminently useful because any difference in the motor and the linear guide velocity results in friction, which works to cancel that velocity difference, so that ultimately, in the steady state, the motor and the linear guide are no longer slipping with respect to each other and the unloaded model is a close approximation–but only an approximation. The velocity of the free end of the beam, $\dot{u}(L,t)$, where the friction contact between the motor and the linear guide takes place, can be derived by differentiating the approximate solution for the position derived earlier:

$$\dot{u}(L,t) \cong \phi_1(L)\frac{\hat{f}_1\eta\omega V_0}{M(\omega_{sys}^2 - \omega^2)}\cos(\omega t). \tag{3.18}$$

Since the velocity of the motor at the contact point is sinusoidal, there will always be some slipping even at constant linear guide speed, which will be naturally lower than, but roughly proportional to the maximum speed of the free end of the beam given by Eqn(3.18), *i.e.*

$$v_{max} \cong \alpha\phi_1(L)\frac{\hat{f}_1\eta V_0\omega}{M(\omega_{sys}^2 - \omega^2)}, \tag{3.19}$$

where α is the slip factor, to be determined empirically. If the heat generated from this slip in friction contact zone is not minimized and conducted away from the contact surface, the motor efficiency can decrease significantly. Slipping also poses an ill omen if the ultrasonic motor is to be used in a clean room environment. At submicron level, even nanometer scale surface irregularities at the contact region on a specially hardened surface (*e.g.* ceramic) are subject to high local stress concentration due to atomic interactions during slipping, and are liable to break off into the surrounding atmosphere, contaminating the clean room air. Hence, practical applications such as semiconductor fabrication mandate absolute minimization of slipping and slip related material loss.

Infinite technicalities notwithstanding, the focus of this modeling effort is not an exact and microscopically detailed analytical model (such things are impossible anyway), but rather a general understanding of the underlying physics, so that important issues can be brought out and possible problems may be isolated. In addition, the mathematical model derived suggests useful methods of motor control. For example, Eqn(3.19) implies that the motor speed may be controlled with either the frequency or amplitude of the applied voltage. Pictorial understanding is also helpful; Fig 2-4 shows that the motor speed may be also controlled via phase difference between the driving sinusoid to the legs and the middle piezo, as mentioned in Chapter 2. These control methods are explored in the next chapter.

Chapter 4

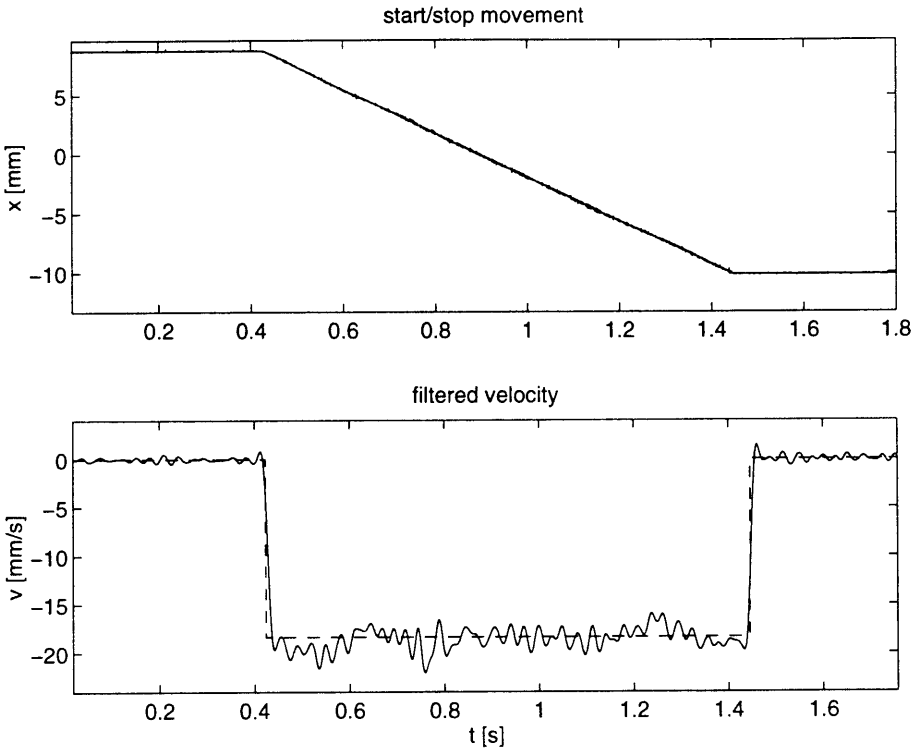
Experimental Results

In this chapter, the analytical model derived in the Chapter 3 is validated through experimental results, and control methods suggested in the same chapter are explored using the setup described in Chapter 2, in which three modes of operation were discussed: high speed transition mode, stepper mode, and high precision mode. This chapter is particularly concerned with experimental performance evaluation of these modes, with emphasis on their respective merits and contingent problems for practical application of the ultrasonic motors for nano-technology, as well as the effects of unmodeled dynamics that results in deviation of the motor behavior from the model are presented. In parallel, other less touted features of the motor are also explored.

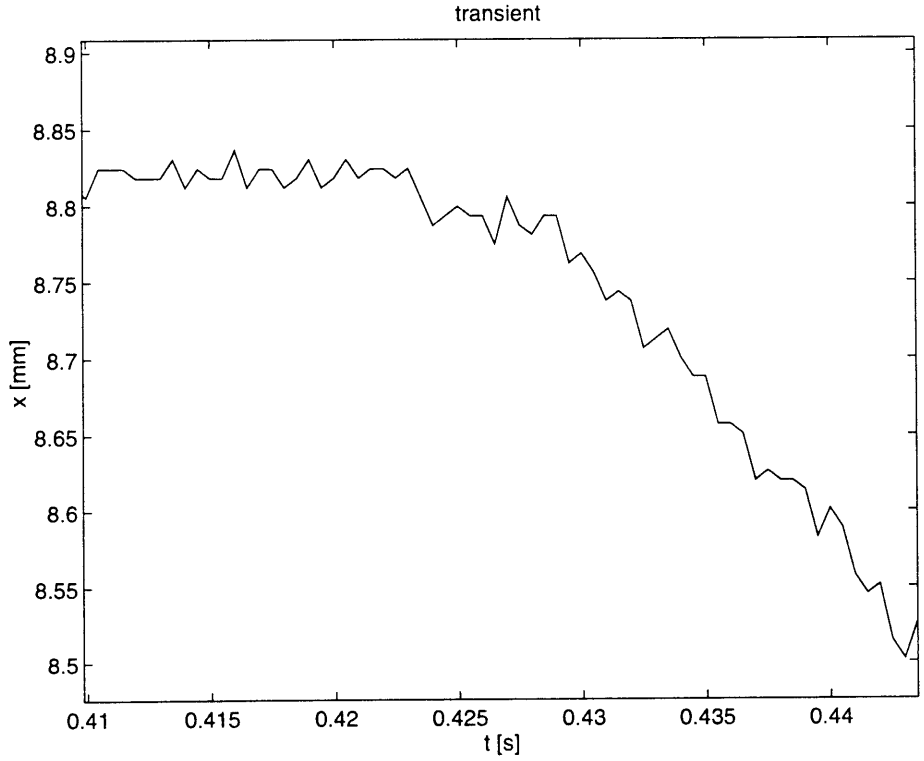
4.1 High speed mode

One of the distinguishing features of this linear ultrasonic motor from other nano-actuators is the high speed capability–as high as $280\frac{mm}{s}$ with this prototype alone–which is decoupled from its nanometer precision capability. The driving voltage to the three piezo stacks and the corresponding leg movement during the high speed mode are depicted in Fig 2-4. Integrated electronic and computer interface explained in the same chapter enable direct motor control, including not only the start and stop command, but also the speed and direction, all with a simple key stroke on the host PC. Fig 4-1 shows typical high speed motor start/stop movement, including the very fast transient, and Fig 4-2 shows the high speed cyclic movement with virtually no time delay in direction reversal. The apparent high frequency velocity ripple in Fig 4-1 is an artifact of the benign low pass filtering. Differentiating noisy quantized position signal from LVDT placed most of the power in the high frequency range, as shown in Fig 2-6. A very agressive low pass filter will surely reduce the distracting ripple in velocity, but will also distort the ultrasonic motor's fast dynamic response. Initially, the motor is at rest with power off. Upon user's start command, the ADSP board starting injecting the three sinusoidal voltages shown in Fig 2-4 to the piezo stacks. During the short transient period going from rest to constant speed and vice versa, the motor imparts the force on the load partly with friction force created during slipping. But as shown in Fig 4-1(c), the response time of the motor is virtually unnoticeable; a closer inspection reveals less than $5ms$ response time, which is difficult to duplicate with a comparable torque DC motor.

Switching the direction of the motor movement is equally straightforward: by offsetting the phase of the sinusoidal voltage applied to the middle piezo stack 180°, as discussed



(a) start/stop movement



(b) detailed look at transient

Figure 4-1: High speed movement

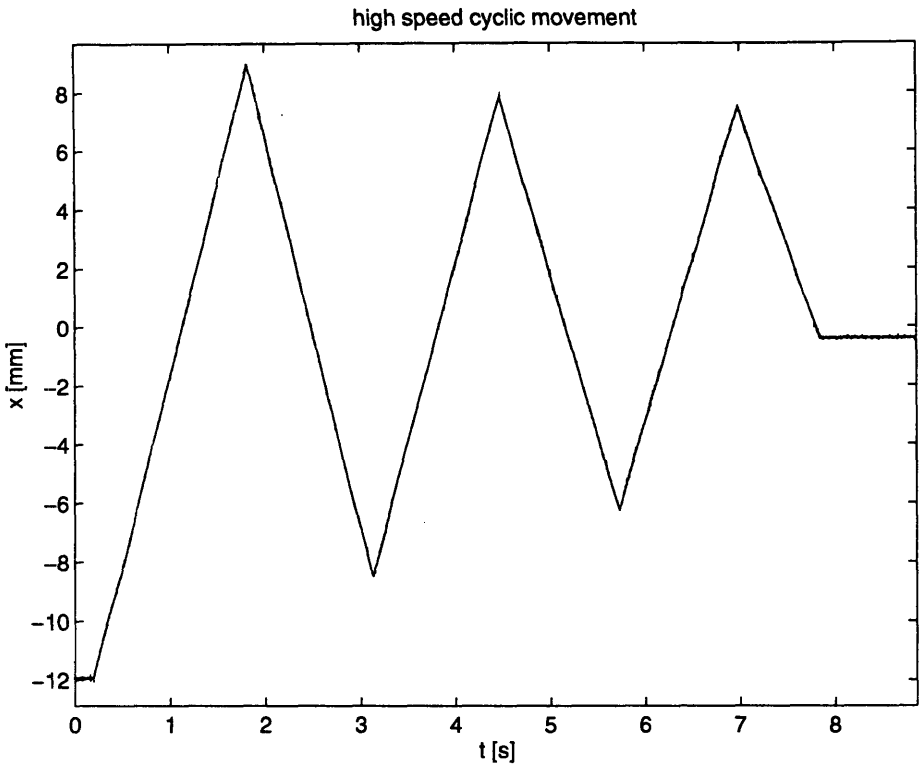
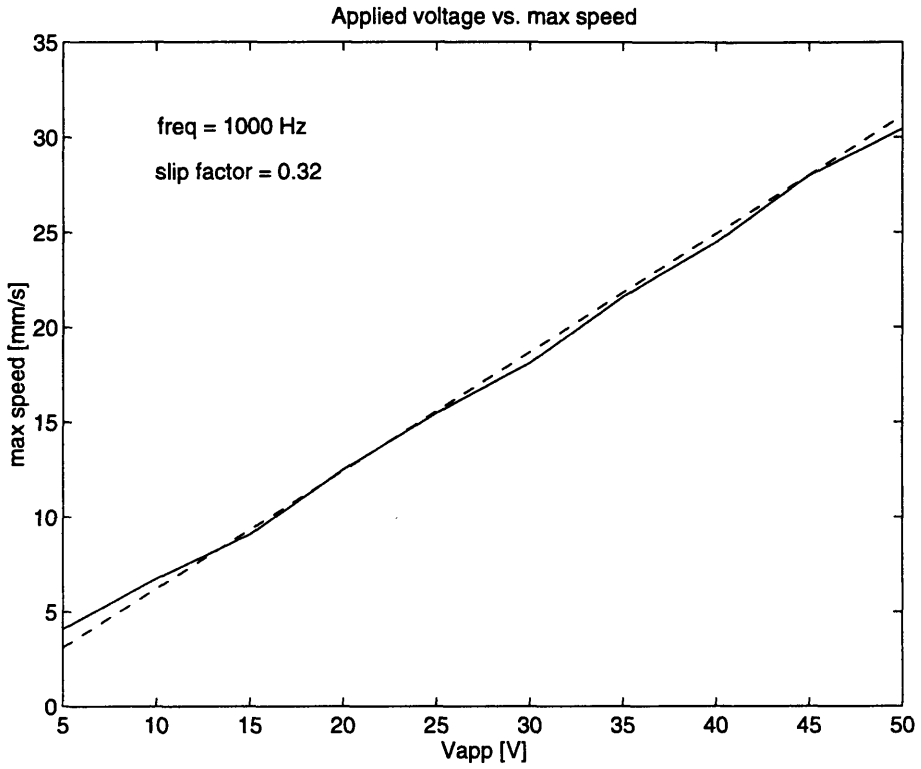


Figure 4-2: Ultrasonic motor position during high speed cyclic movement

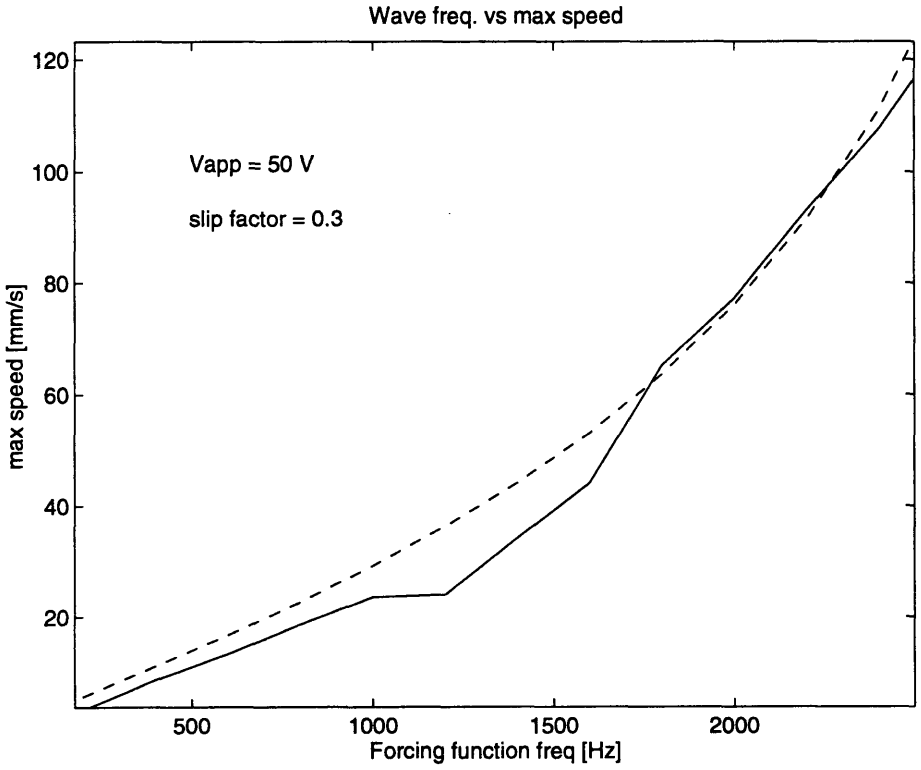
in Chapter 2. In fact, all the motor control methods proposed in Chapter 3 based on Eqn(3.19) were validated through experiments. According to Eqn(3.19), the motor speed is proportional to the applied voltage to the middle piezo stack and also $\omega/(\omega_{sys}^2 - \omega^2)$ as a function of the driving sinusoid frequency ω, where ω_{sys} is the natural frequency of the whole system given in Eqn(3.14). All constants except the slip factor α were known, so that simple least squares fit was sufficient to determine empirical value for α. As Fig 4-3 shows, the experimentally observed behavior closely matches the model with the approximate slip factor of 0.32. While α is a convenient empirical parameter, it is important to bear in mind that this slip factor is a lumped parameter incorporating complicated unmodeled nonlinear and time varying nature of the system such as the effect of preload, temperature and humidity on the friction in the contact zone into a manageable empirical form. However, the slip factor of 0.32 implies the motor is slipping more than is desirable; while $\alpha \geq 1.0$ is impossible, an efficient ultrasonic motor should achieve a slip factor of 0.5 or higher easily. Low slip factor is indicative of an inefficient motor that is rubbing off too much contaminants to be acceptable for a class 1 clean room. Given that this prototype was not designed for efficiency and is operating in a normal room atmosphere, the current slip factor is acceptable.

4.2 Stepper mode

Stepper mode is a natural extension of the high speed mode in its operation principle and stands in between the high speed mode and high precision mode. As Fig 4-4 shows, the natural measurement length scale for the stepper mode is μm, as supposed to mm for the



(a) V_{app} vs. v_{motor}



(b) f vs. v_{motor}

Figure 4-3: Comparison of the observed data (solid curve) and model (dashed curve) for the motor speed

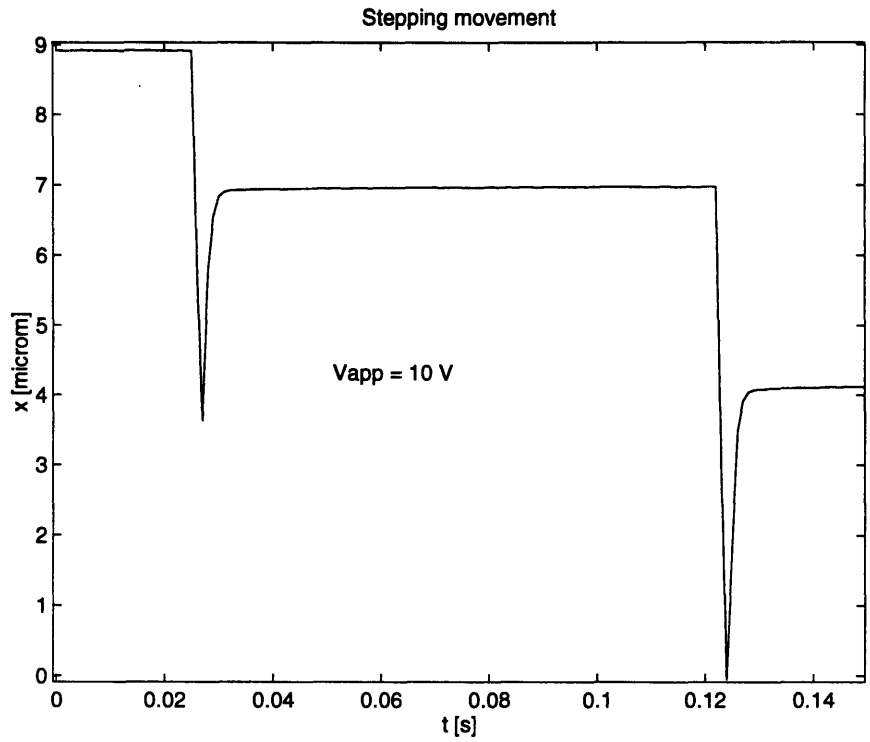


Figure 4-4: Typical stepping movement measured with the capacitive probe

high speed mode and *nm* for the high precision mode. For the high speed mode, long sequences of sinusoids were injected to the three piezo stacks. In contrast, only one such period is used in the stepper mode. Of course, the waveform is not restricted to smooth sinusoids; any periodic waveform is allowed. Sinusoids leave too much room for slipping, as seen by the prototype's low slip factor, and cannot quite move the same distance per step possible with square waves. If the motor is visualized as a pair of legs, the stepper mode operation with square wave is intuitively obvious. Initially, only one of the two legs is contacting the linear guide surface. Sufficient preload ensures high stiffness in this contact, so that the leg does not slip on the linear guide when the middle piezo stack expands due to a step voltage applied to its leads. After the middle piezo stack has expanded, the motor (or the linear guide, depending on where the reference frame is) has traveled half the step and the two legs switch roles. In this experiment, no input smoothing was used for simplicity's sake, and the friction contact force and location changes abruptly when the legs contract and expand. Once the new friction contact is established, the middle piezo stack now contracts to its original length, moving the motor another half step to its final position. Throughout the stepping procedure, only step voltages are applied to the piezo stacks, so that the motor movement is jerky, producing sharp overshoot in position shown in Fig 4-4. In fact, these spikes are more of an artifact of the vibration of the electrode plate put on the linear guide to serve as a ground for the capacitive sensor, caused by the impulse force imparted on the linear guide by the piezo stack moving under the step command voltage. Since the linear guide itself much more massive than the ultrasonic motor, the linear guide exhibiting the high frequency motion under the debate is unlikely. Rather, the small, yet stiff electrode

plate is suspected. However, after the transient dies out, the steady state position change is a genuine movement commanded by the stepper mode. If small movement and slip related efficiency loss and contamination are tolerable, sinusoids instead of square wave will produce a smoother response.

4.3 High precision mode

Even though the high speed and stepper capabilities give the ultrasonic motor a critical edge over other nano-actuators, it must still prove its nanometer resolution movement capability in order to substantiate its claim as a nano-actuator. As expected, the high precision mode displayed sub-nanometer resolution, dispelling all reservations.

The high precision mode is essentially the same as the first half of the stepper mode; only one leg contacts the ground to move the motor by expanding or contracting the middle piezo stack. But unlike the stepper mode, the contact point is never changed, so that the range of motion is fundamentally limited by the maximum expansion of the middle piezo stack. Since this is structurally analogous to amplifying the piezo stack displacement with a lever arm, the resolution of the entire motor is limited only by the intrinsic resolution of the piezo stack and the digital circuit used to apply voltage to its leads. Hence, the theoretical resolution limit is proportional to the quantization resolution of the ± 5 V signal from 16 bit D/A on the ADSP board, amplified 20 times by a power amplifier. Furthermore, a close inspection of the high precision movement such as those in Fig 4-5 reveals that a 1 volt step change to the piezo stack corresponds to approximately $80nm$, leading to the final theoretical position resolution of

$$20\frac{10V}{2^{16}}\frac{80nm}{1V} = 0.244nm.$$

In practice, such fantastic resolution cannot be confirmed because the capacitive probe used to measure the position is guaranteed to only $1nm$ resolution. Nevertheless, Fig 4-5 clearly shows that the actuator resolution is comparable to the sensor resolution, if not better.

Observation of the high precision movement shown in Fig 4-5 also reveals a curious nonlinear phenomenon that is not apparent in other modes: creep, which is a steady drift in position even after the commanded voltage is constant and transient dynamics has decayed. In addition, though it is not apparent in Fig 4-5, hysteresis is often a prominent phenomenon in piezoelectric material that is extensively discussed in literature [7, 17]. These nonlinear phenomena are not easily modeled with simple linear time invariant model such as those used in this thesis, but easily dealt with a feedback controller.

4.4 Other advantages of linear ultrasonic motor

The advantages of using a linear ultrasonic motor go far beyond the high speed and high precision capability. Although the prototype built here is quite noisy due to an accidental design constraint explained in Chapter 2, ultrasonic motors can be noiseless by definition, and the control methods and results presented earlier clearly indicate that future design iterations will be noiseless and ergonomic, adding value to a product. Even with ergonomics aside, the sheer performance gain favors the ultrasonic motors over other nano-actuators. The response time of the ultrasonic motor is usually smaller than that of a comparable

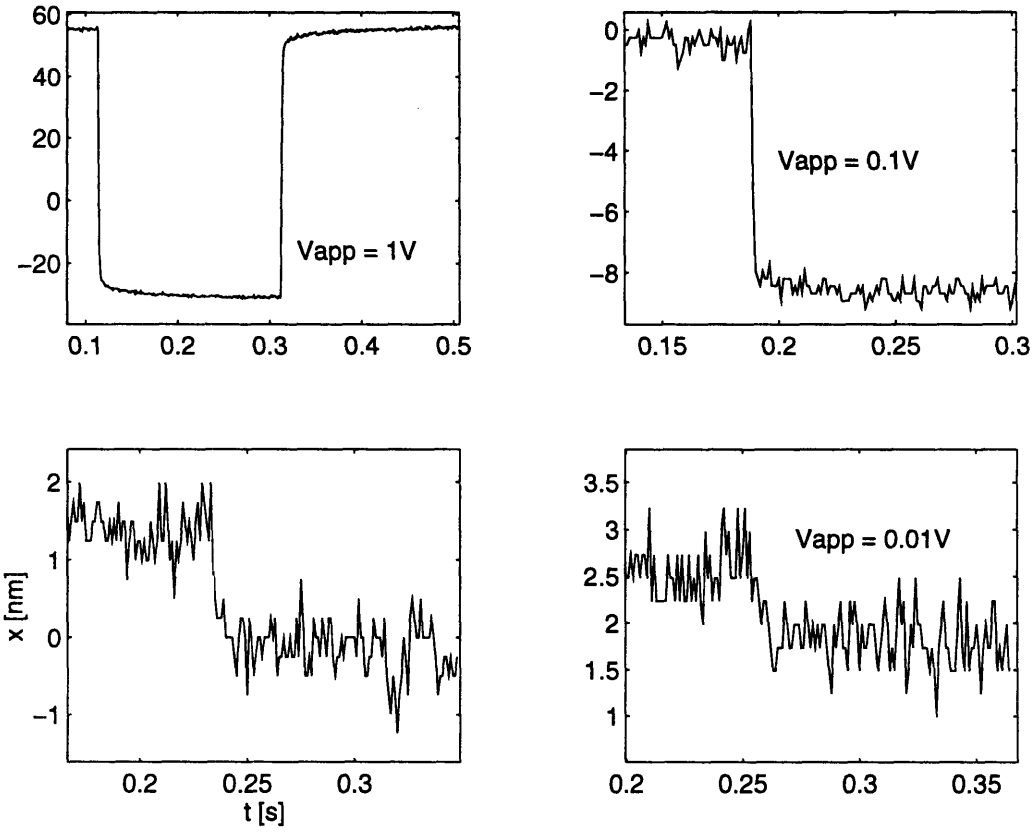


Figure 4-5: High precision movement measured with the capacitive probe

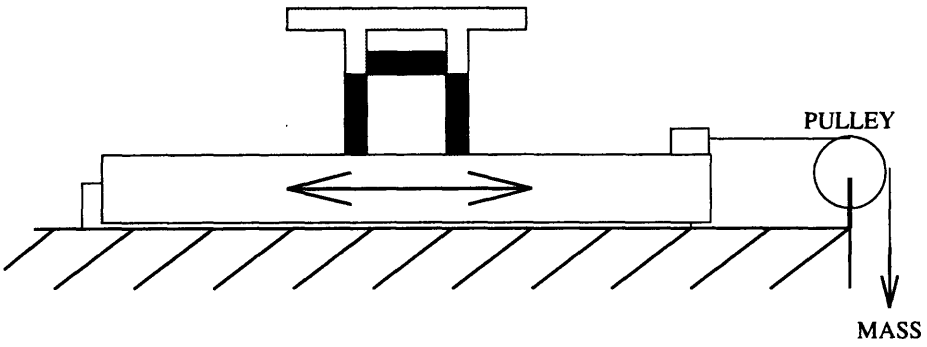


Figure 4-6: Static stiffness measurement setup

torque DC motor by an order or magnitude or less, and it achieves this without sacrificing the high static stiffness. Once appropriate preload is put on the motor, both legs make direct friction contact with the linear guide surface, rendering a very high static stiffness–even with the power off. An actuator with high static stiffness is well suited for effective disturbance rejection during trajectory tracking movement and especially when it is commanded to hold its position, for example, during the photolithography process during semiconductor fabrication.

Coefficient of friction μ was chosen as a universal parameter for measuring the static stiffness throughout the experiment. It is the ratio of the holding force to the known preload on the motor. The holding force is measured by pulling on the linear guide until the motor slips, as shown in Fig 4-6. The smallest value for μ is 0.5 with the power off. Applying voltage to the leg piezo stacks effectively increase the preload, and the holding force goes up even higher. This is a remarkable figure of merit, especially when compared against comparable size DC motors, which have zero static stiffness with power off, and only the large DC motors capable of handling high current can match the static stiffness of the ultrasonic motors. In short, the direct friction contact and the high energy density coupled with stiff structure of the piezoelectric material make the high static stiffness and bandwidth possible.

Chapter 5

Conclusion and Future Work

The linear ultrasonic motor presented here possesses excellent potential as a nano-actuator that will drive tomorrow's high performance devices. Extremely high speed motion capability was achieved without sacrificing either the sub-nanometer positioning accuracy, high bandwidth, high static stiffness, or high torque to weight ratio. These performance advantages over other nano-actuators stem from effective design that couples superior material property of piezoelectric material with a simple structure, and allows direct friction contact between the motor and the linear guide.

The eventual goal of the project is to develop a fully integrated 2D digital controlled nano-actuator and nano-sensor servo system based on micro machining technology currently being developed, such as in the Microelectromechanical systems (MEMS) [6, 39, 20]. For now, the dominant fabrication technique is based on semiconductor fabrication technique. But other non traditional techniques such as micro electrodischarge machining (MEDM), micro stereolithography, and hybrid biological and artificial micro-fabrication have been reported as well [10]. In parallel, active investigations are discovering alternatives to silicon, such as conducting polymers, in an effort to build complex 3D micromachined structures with various desirable properties [30]. Numerous variety of material and fabrication technique notwithstanding, however, the ultimate objective is to scale down currently proven electromechanical devices to a micron sized integrated unit to be used in a metrology frame based servomechanical system.

If the ultrasonic motor can be embedded into a small integrated device, the performance advantages already inherent in the ultrasonic motors are enhanced even further. Smaller size and lighter weight will boost the already high bandwidth and make the motor even more compact with high torque to weight ratio, and integrated manufacturing process will render the motor more robust. In one possible scenario assuming the current MEMS technology, all the supporting electronic circuit that now connects motor unit to the DSP board can be shrunk down on a single semiconductor based MEMS chip, and the actuator-sensor unit can literally fit into the palm of a hand, replacing the complex, bulky and delicate systems currently in use. Still, the nano-technology is in an embryonic state, and the promise of such advanced system can only be fulfilled by gaining more experience–some of it by trial and error–in design and control of nano devices. The experiments presented in this thesis are indicative samples of such required efforts. Some of the issues that need to be resolved in the future for practical application of ultrasonic motors in nano-technology are:

1. Robust design: The high stiffness and torque to weight ratio are two of the advantages of ultrasonic motors, but they also put the motor under considerable stress during

operation. In fact, the robustness of the motor proved to be a major performance limit for this prototype, as the super glue type bonding layer between the piezo stack and the aluminum peg would fracture at super high speed movement or under large load.

2. Friction: Although ultrasonic motors derive their high static stiffness and accuracy from direct friction contact between the rotor and the stator, maintaining optimal friction condition is a difficult affair because of the complex inter-atomic forces that determine overall friction effect. Interdependence of the friction layer material property on preload and environment conditions such as temperature and humidity have so far defeated all efforts–both theoretical and empirical–to capture its essence in a manageable form, at least certainly not in a form amenable for real time control of the friction condition. Despite this clear and present challenge, a headlong rush to in-depth investigation of friction behavior without fundamental improvement in friction material or contact method to be used in future design will yield low return on investment. In case of grade 1 clean room environment, the particle contamination must be also considered. Sophisticated controllers for time varying system are costly and should be used only after fundamental improvements in materials and design are made.

3. 2D motor design: Modularity of this prototype allows a straightforward construction of the 2D motor discussed in Chapter 2. While the concept may be uninvolved, actual implementation of real-time control of such 2D motor presents a steep engineering challenge. With current design, each motor has three piezo stacks to be controlled; the 2D version has four such motors, with 12 piezo stacks requiring 12 high speed D/A. The minimum number of inputs required is three, measuring position in each axis and the angle of rotation; if velocity sensors will be used, possibly six A/D may be necessary. Control of a system with so many inputs and outputs, running at ultrasonic frequency, varying in time with large uncertainty is a monumental task indeed.

Bibliography

[1] Analog Devices. *ADSP-21020 User's Manual*, first edition, 1991.

[2] Analog Devices. *Amplifier Reference Manual*, 1992.

[3] Apex Microtechnologies Corporation, 5980 N. Shannon Road, Tucson AZ, 85741-5230. *APEX Data Book.*

[4] Jungmok Bae. Modeling, Control, and Experimentation of a Scanning Tunneling Microscope. Master's thesis, MIT, May 1994.

[5] Borland International, Inc., 100 Borland Way, Scotts Valley CA 95067-3249. *Borland C++ Version 4.0 User's Guide*, 1993.

[6] Janusz Bryzek, Kurt Petersen, and Wendell McCulley. Miconmachines on the March. *IEEE Spectrum*, 1994.

[7] Walter G. Cady. *Piezoelectricity.* Dover Publications, 1964.

[8] Stephen H. Crandall, Dean C. Karnopp, Edward F. Kurtz Jr., and David C. Pridmore-Brown. *Dynamics of Mechanical and Electromechanical Systems.* Krieger Publishing Company, 1968.

[9] Douglas S. Crawford. Sensor Design and Feedback Motor Control for Two Dimensional Linear Motors. Master's thesis, MIT, May 1995.

[10] P. Dario, M. C. Carrozza, N. Croce, M. C. Montesi, and M. Cocco. Nontraditional Technologies for Microfabrication. *Journal of Micromechanics and Microengineering*, 5(2):64–71, June 1995.

[11] A. Endo and N.Sasaki. Inverstigation of Frictional Material for Ultrasonic Motor. volume Supplement 26, pages 197–199. Japanese Journal of Applied Physics, 1987.

[12] Bruce W. Char et al. *Maple V Library reference manual.* Waterloo Maple Software, 1st edition, 1991.

[13] James Gort. *21020 DSP Board Manual.* JRG Signal Processing System, preliminary edition, May 1994.

[14] Takeshi Hatsuzawa, Kouji Toyoda, and Yoshihisa Tanimura. Speed Control Characteristics and Digital Servosystem of a Circular Traveling Wave Motor. *Review of Scientific Instruments*, 57(11):2886–2890, November 1986.

[15] Hiroshi Hirata and Sadayuki Ueha. Characteristics Estimation of a Traveling Wave Type ultrasonic Motor. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 40(4):402–406, July 1993.

[16] IEEE Workshop on Microelectromechanical Systems. *Ferroelectric Thin Film Ultrasonic Micromotors*, 1991.

[17] Takuro Ikeda. *Fundamentals of Piezoelectricy.* Oxford University Press, 1990.

[18] R. Inaba, A. Tokusguma, O. Kawasaki Y. Ise, and H. Yoneno. Piezoelectric ultrasonic motor. pages 747–755. IEEE, 1987.

[19] Satoshi Iwamatsu, Sadayuki Ueha, Minoru Kuribayashi, and Eiji Mori. Rotary Ultrasonic Motor using Extensional Vibration of a Ring. *Japanese Journal of Applied Physics*, Supplement 25(1):174–176, 1986.

[20] S. Kota, G. K. Ananthasuresh, S. B. Crart, and K. D. Wise. Design and Fabracation of Microelctromechanical Systems. *Journal of Mechanical Design*, 116:1081–1088, December 1994.

[21] Lucasconrtol Systems Products, 1000 Lucas Way, Hampton VA 23666. *Lucascontrol Systems Products DC-LVDT Manual.*

[22] Mechanical Technology Inc., 968 Albany-Shaker Road, Latham NY 12110. *MTI Instruments Division Instruction Manual.*

[23] Richard M. Murray. *Sparrow 2.1d Reference Manual.* California Institute of Technology, December 1994.

[24] Kentaro Nakamura, Minoru Kurosawa, and Sadayuki Ueha. Design of a Hybrid Transducer Type Ultrasonic Motor. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 40(4):395–400, July 1993.

[25] M. P. Norton. *Fundamentals of Noise and Vibration Analysis for Engineers.* Cambridge University Press, 1989. p. 85.

[26] Tetsuo Ohara. Dynamics and Control of Piezotube Actuators for Subnanometer Precision Applications. American Controls Conference, 1995.

[27] Osamu Ohnishi, Osamu Myohga, Tadao Uchikawa, Masashi Tamegai, Takeshi Inoue, and Sadayuki Takahashi. Piezoelectric Ultrasonic Motor Using Longitudinal-Torsional Composite Resonance Vibration. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 40(6):687–693, November 1993.

[28] Reymond M. Redheffer. *Differential Equations: Theory and Applications.* Jones and Barlett Publishers, 1991.

[29] Alexander H. Slocum. *Precision Machine Design.* Prentice Hall, 1992.

[30] E. Smela, O. Inganas, and I. Lundstrom. Controlled Folding of Micrometer-Size Structures. *Science*, 268(5218):1735–1738, June 1995.

[31] V. Snitka, V. Mizariene, and V. Baranauskas. Nanometric Resolution Ultrasonic Stepper Motor. pages 553–556. IEEE Ultrasonics Symposium, 1994.

[32] Takehiro Takano, Yoshiro Tomikawa, Toshiharu Ogasawara, and Chiharu Kusakabe. Characteristics and a New Control Method of a Same-Phase Drive-Type Ultrasonic Motor. *Japanese Journal of Applied Physics*, 30(9B):2277–2280, September 1991.

[33] Tokin Corporation. *Multilayer Piezoelectric Actuator*, 4th edition.

[34] R. Inaband A. Tokushima, O. Kawasaki, Y. Ise, and H. Yoneno. Piezoelectric Ultrasonic Motor. pages 747–756. IEEE Ultrasonics Symposium, 1987.

[35] Yoshiro Tomikawa, Kazunari Adachi, Manabu Aoyagi, Tadatsu Sagae, and Takehiro Takano. Some Constructions and Characteristics of Rod-Type Piezoelectirc Ultrasonic Motors Using Longitudinal and Torsional Vibrations. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 39(5):600–607, September 1992.

[36] Yoshiro Tomikawa, Tetsuya Kondo, Toshiharu Ogasawara, Sumio Sugawara, and Masashi Konno. Fundamental Considerations of Excitation of a Flexural Progressive Wave and its Application. *Japanese Journal of Applied Physics*, Supplment 26(1):194–196, 1987.

[37] Junichi Toyoda and Kanji Murano. A Small-Size Ultrasonic Linear Motor. *Japanese Journal of Applied Physics*, 30(9B):2274–2276, September 1991.

[38] K. Uchida. *Piezoelectric Actuator*. Morikita, 1986. In Japanese.

[39] A. B. Frazier R. O. Warrington and C. Friedrich. The Miniaturization Technologies–Past, Present, and Future. *IEEE Transactions on Industrial Electronics*, 42(5):423–430, October 1995.

Appendix A

Hardware Setup

A.1 Electronic interface

The custom made signal filtering and amplyfying electronics interface unit serves as the go-between for the physical experimental setup and the real-time operating kernel. As hinted in Fig 2-7, the control signals to the motor are issued from the three high speed D/A converters connected directly to the ADSP board. Ideally, all three channels should be amplified by multi-channel high speed power amplifier; but only one high voltage bipolar power amplifier[1] with one channel was available at the experimental setup stage. Since the middle piezo stack requires higher operating voltage than the leg piezo stacks, the NF PowerAmpifier is dedicated to the middle piezo stack, and the two others are driven with custom designed electronic unit, whose pin outs are shown in Fig A-1, that includes a pair of high voltage, high speed op-amps[2] [3] a pair of isolation amplifiers[3] [2]. Both output to the

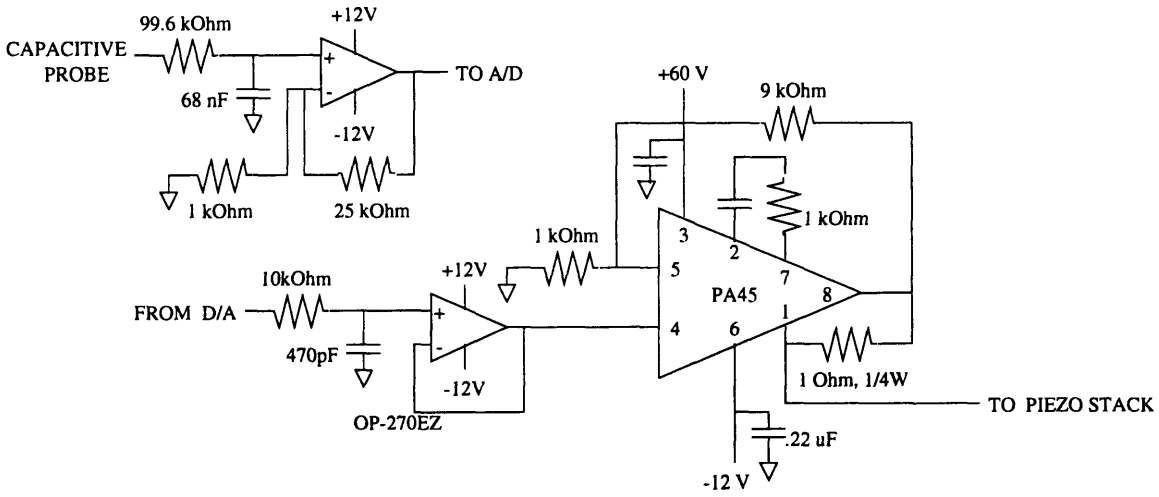


Figure A-1: Wiring diagram

[1]Manufactured by NF Electronics, Japan

[2]Model number PA-45 manufactured by Apex Microtechnology Corporation, 5980 N. Shannon Road, Tucson AZ, 85741–5230, (520) 690–8600

[3]Model OP-270 EZ manufactured by Analog Devices, One Technology Way, PO Box 9106, Norwood MA 02062–9106, (617) 329–4700

pin	assignment
1	D/A 1
2	D/A 2
3	D/A 3
4	To NF amplifier
5	From NF amplifier
6	PA45 unit 1 pin 1
7	PA45 unit 1 pin 2
8	PA45 unit 1 pin 3
9	PA45 unit 1 pin 4
10	PA45 unit 1 pin 5
11	PA45 unit 1 pin 6
12	PA45 unit 1 pin 7
13	PA45 unit 1 pin 8
14	GND
15	GND
16	null
17	null
18	PA45 unit 2 pin 1
19	PA45 unit 2 pin 2
20	PA45 unit 2 pin 3
21	PA45 unit 2 pin 4
22	PA45 unit 2 pin 5
23	PA45 unit 2 pin 6
24	PA45 unit 2 pin 7
25	PA45 unit 2 pin 8

Table A.1: DB-25 connector pin assignment

motor and the position signal input from sensors are routed through the DB-25 connector on the front panel of the box. The pin assignment is tabulated in Table A.1.

A.2 Hardware manufacturers

Piezo stack (NLA5×5×18 mm^3)	Tokin Corporation 155 Nicholson Lane San Jose CA 95134 (408) 432-9020
Friction coat (Gacoflex N-1700)	Gaco Western, Inc. PO Box 88698 Seattle WA 98138-2698 (206) 575-0450
LVDT (100 DC-D)	Lucas Control Systems Products 1000 Lucas Way Hampton VA 23666 (804) 766-4494
ADC (CIO-DAS1400)	Computer Boards, Inc. 125 High Street #6 Mansfield MA 02048 (508) 261-1123
Capacitive sensor (ASP-1 with AS-1021-SAI)	Mechanical Technology, Inc. 968 Albany-Shaker Road Latham NY 12110 (518) 785-2211
Power op-amps (PA45)	Apex Microtechnology Corporation 5980 N. Shannon Road Tucson AZ, 85741–5230 (520) 690–8600
Isolation amplifier (OP-270 EZ)	Analog Devices One Technology Way, PO Box 9106 Norwood MA 02062–9106 (617) 329–4700

Appendix B

Software

As already discussed in Chapter 2, the backbone of the present experimental setup is the real time control kernel that links the host PC with custom made ADSP board. Here, the actual programs are given and the procedure for using them is outlined.

B.1 PC-ADSP interface

The basic building block of the operating kernel is the Sparrow package discussed earlier in Chapter 2. Interested reader will need to consult the manual [23] for details. The following C program links the host PC with the high speed ADSP board was constructed on top of the existing Sparrow architecture.

```
/* adsp.h - definitions for ADSP21020 library
 * HOC August 9 95
 */

#ifndef __ADSP_INCLUDED__
#define __ADSP_INCLUDED__

enum num_format {INT, UNSIGNED, FLOAT, HEX=16};

#define IS_ADSP_PM 0x80
#define IS_ADSP_DM 0x00

#define BUFSIZE 1024        /* used in downloading stk file */
#define DEAD_HEADER "000000000000000000000000\n"
#define MEMERR(str) \
  fprintf(stderr, "Out of Memory Error - %s (%d)\n", str, __LINE__)
#define FUNCERR(str1, str2) \
  fprintf(stderr, "Error encountered in %s - %s (%d)\n", str1, str2, __LINE__)

#define adsp_read_int(adsp_base, adr, datap) \
  adsp_read_dm(adsp_base, adr, (unsigned short *) datap)
#define adsp_read_unsigned(adsp_base, adr, datap) \
  adsp_read_dm(adsp_base, adr, (unsigned short *) datap)
```

```
#define adsp_read_float(adsp_base, adr, datap) \
  adsp_read_dm(adsp_base, adr, (unsigned short *) datap)
#define adsp_write_int(adsp_base, adr, datap) \
  adsp_write_dm(adsp_base, adr, (unsigned short *) datap)
#define adsp_write_unsigned(adsp_base, adr, datap) \
  adsp_write_dm(adsp_base, adr, (unsigned short *) datap)
#define adsp_write_float(adsp_base, adr, datap) \
  adsp_write_dm(adsp_base, adr, (unsigned short *) datap)

/* ADSP memory structure
 * refer to 21020 board user's manual PC interface section
 */
typedef struct DM_struct {      /* data memory 40 bits long by 32k */
  unsigned short dmdh;          /* 1 word */
  unsigned short dmdm;          /* 1 word */
  unsigned char dmdl;           /* 1 byte, but use word operation to read
                and write, then mask off bits 0-7 (pad) */
  unsigned char pad;            /* unused but needed to make a word together
                with dmdl */
} ADSP_DM;

typedef struct PM_struct {      /* program memory 48 bits long by 32k */
  unsigned short pmdh;
  unsigned short pmdm;
  unsigned short pmdl;
} ADSP_PM;

/* main ADSP I/O function prototypes */
void asc2DM(char *buffer, ADSP_DM *dmp);
void asc2PM(char *buffer, ADSP_PM *pmp);
void conv_endian(short memtype, unsigned char *data,  unsigned char *ans);
void adsp_read(short adsp_base, short memtype, short adr,  unsigned short *data);
void adsp_read_dm(short adsp_base, short adr, unsigned short *data);
void adsp_write(short adsp_base, short memtype, short adr, unsigned short *data);
void adsp_write_dm(short adsp_base, short adr, unsigned short *data);
int adsp_rdfloat_driver(DEVICE *dp, CHANNEL *cp, DEV_ACTION action);
int adsp_wrfloat_driver(DEVICE *dp, CHANNEL *cp, DEV_ACTION action);
int adsp_load_stk(char *fname, short adsp_base);
int adsp_stat(short adsp_base);
int adsp_start(short adsp_base);
int adsp_stop(short adsp_base);
int adsp_set_user_var(short adsp_base, short adr, double ddata);

#endif /* __ADSP_INCLUDED__ */
/* adsp.c - I/O routines for ADSP21020 library
 * HOC August 95
 */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include "channel.h"
#include "adsp.h"

/* Main ADSP I/O routines:
 * asc2DM
 * asc2PM
 * adsp_read
 * adsp_read_int
 * adsp_read_unsigned
 * adsp_read_float
 * adsp_write
 * adsp_write_int
 * adsp_write_unsigned
 * adsp_write_float
 * adsp_load_stk
 * adsp_stat
 * adsp_start
 * adsp_stop
 * adsp_set_user_var
 */


/* This is already being done with the device driver, but we don't want to use the
 * channel interface for all the user variables that need updating once in a while
 */
int adsp_set_user_var(short adsp_base, short adr, double ddata) {
  float fdata;
  fdata = (float) ddata;
  adsp_write_float(adsp_base, adr, &fdata);
  return 0;
}


/* Parse headers for codes in stk file; return data block length
 * header format: aabbccddeeeeeeeeffffffff where
 *   aa       : width of address and length fields
 *   bb       : reserved
 *   cc       : PROM splitting flag ( 80=PM, 00=DM)
 *   dd       : user defined flags, loaded using -u switch
 *   eeeeeeee : start address of data block
 *   ffffffff : number of bytes that follow (until next header or
 *              termination); must be nonzero
 */
int getheader(char *buffer, int *memtype, short *addr, int *hline) {
```

```
  int width, blklen = 0;
  char tmpstr[10];
  union {
   unsigned short nib[2];
   unsigned long  tot;
  } temp;

  strncpy(tmpstr, buffer, 2);    /* copy width info */
  width = (int) strtol(tmpstr, NULL, HEX);
  if(width > (sizeof(long) << 3)) {
    fprintf(stderr, "Address width too large (%d) - getheader\n", width);
    return 0;
  }
  strncpy(tmpstr, buffer+4, 2);    /* copy memtype info */
  (*memtype) = (int) strtol(tmpstr, NULL, HEX);
  strncpy(tmpstr, buffer+16, 8); /* copy data length info */
  temp.tot = strtol(tmpstr, NULL, HEX);
  if((*memtype) == IS_ADSP_PM) {
    blklen = (int) (temp.tot/6);
    (*hline) += blklen;
  }
  else if(!(*memtype)) {
    blklen = (int) (temp.tot/5);
    (*hline) += blklen;
  }
  else {
    fprintf(stderr, "Unknown memtype (%d) - getheader\n", (*memtype));
    return 0;
  }
  (*hline)++;
  strncpy(tmpstr, buffer+8, 8);    /* copy start address info */
  temp.tot = strtol(tmpstr, NULL, HEX);
  if(temp.nib[1] ≠ 0) {    /* ADSP board address only go up to 7FFF */
    fprintf(stderr, "Bogus address (%ld) - getheader\n", temp.tot);
    return 0;
  }
  (*addr) = temp.nib[0];
  return blklen;
}


/* Download STK file to DSP board (ordinarily to PM)
 * returns number of line processed upon success
 */
int adsp_load_stk(char *fname, short adsp_base) {
  FILE *fp = NULL;
  char *buffer;
  short addr;
```

```
int line, hline, memtype;    /* line #, header line #, adsp memory type */
ADSP_DM dm;
ADSP_PM pm;

if((fp = fopen(fname, "r")) == NULL) { /* Look for the stk file */
  fprintf(stderr, "Unable to open byte-stacked formatted file %s\n", fname);
  return 0;
}
          /* allocate space for input buffer */
if((buffer = (char *) calloc(BUFSIZE, sizeof(char))) == NULL) {
  MEMERR("adsp_load_stk");
  fclose(fp);
  return 0;
}
line = 0;       /* line # never preceeds header line # */
hline = 1;
while(fgets(buffer, BUFSIZE, fp)) {
  if(strlen(buffer) == BUFSIZE-1) {
    fprintf(stderr, "Input buffer overflow; exiting...\n");
    return 0;
  }
  line++;          /* process next line */
  /* read header line if code has been processed up to the header line */
  if(line == hline) {
    if(!strcmp(buffer, DEAD_HEADER)) { /* use dead header to signal end of file */
  fprintf(stdout, "Finished processing file (line %d)...\n", line);
  fclose(fp);
  free(buffer);
  return line;
    }
    if(!getheader(buffer, &memtype, &addr, &hline)) {
  FUNCERR("getheader", "adsp_load_stk");
  fclose(fp);
  free(buffer);
  return 0;
    }
    else if(memtype == IS_ADSP_PM)
  fprintf(stdout, "Processing PM data (line %d)...\n", line);
    else if(!memtype)
  fprintf(stdout, "Processing DM data (line %d)...\n", line);
    else {
   fprintf(stdout, "Unknown type encountered (type %x, line %d)...\n", memtype,
line);
  fclose(fp);
  free(buffer);
  return 0;
    }
    continue;
```

```
    }
    switch(memtype) {    /* use generic adsp_write for full 40/48 bit code */
    case IS_ADSP_DM:  /* write to DM */
      asc2DM(buffer, &dm);
      adsp_write(adsp_base, memtype, addr, (unsigned short *) &dm);
      break;
    case IS_ADSP_PM:  /* write to PM */
      asc2PM(buffer, &pm);
      adsp_write(adsp_base, memtype, addr, (unsigned short *) &pm);
      break;
    }
    addr++;        /* next code address in the consecutive code block */
  }
  fprintf(stderr, "WARNING: Termination header not found\n");
  fclose(fp);
  free(buffer);
  return line;
}


/* Get ADSP board status */
int adsp_stat(short adsp_base) { return inpw(adsp_base+0x10); }


/* Start ADSP operation from the *top* and returns the status
 * toggles the RESET bit of the ADSP CNTL register and consequently resets the
 * ADSP chip, but not the board
 */
int adsp_start(short adsp_base) {
  outpw(adsp_base+0x04, 0x1);    /* release RESET bit of the control register */
  return adsp_stat(adsp_base);
}


/* Stop ADSP operation and return status
 * clears the RESET bit of the ADSP CNTL register and halts the execution
 */
int adsp_stop(short adsp_base) {
  outpw(adsp_base+0x04, 0x0);    /* clear RESET bit of the control register */
  return adsp_stat(adsp_base);
}


/* Convert a string to DM */
void asc2DM(char *buffer, ADSP_DM *dmp) {
  buffer[10] = '\0';
  dmp→pad = (unsigned char) '\0';
  dmp→dmdl = (unsigned char) strtol(buffer+8, NULL, HEX);
```

```
  buffer[8] = '\0';
  dmp→dmdm = (int) strtol(buffer+4, NULL, HEX);
  buffer[4] = '\0';
  dmp→dmdh = (int) strtol(buffer, NULL, HEX);
}


/* Convert a string to PM */
void asc2PM(char *buffer, ADSP_PM *pmp) {
  buffer[12] = '\0';
  pmp→pmdl = (int) strtol(buffer+8, NULL, HEX);
  buffer[8] = '\0';
  pmp→pmdm = (int) strtol(buffer+4, NULL, HEX);
  buffer[4] = '\0';
  pmp→pmdh = (int) strtol(buffer, NULL, HEX);
}


/* DSP data format is forward and DOS is backward */
void conv_endian(short memtype, unsigned char *data, unsigned char *ans) {
  /* handle pm and dm differently on low word (DMDL or PMDL) */
  if(memtype) {                /* pm */
    ans[0] = data[4];
    ans[1] = data[5];
  }
  else {                     /* dm */
    ans[0] = data[5];
    ans[1] = data[4];
  }
  ans[2] = data[2];            /* DMDM and PMDM */
  ans[3] = data[3];
  ans[4] = data[0];            /* DMDH and PMDH */
  ans[5] = data[1];
}


/* read one data from DM using only DMDM and DMDH
 * note DMDM is now *ahead* of DMDH because of DOS endian format
 */
void adsp_read_dm(short adsp_base, short adr, unsigned short *data) {
  outpw(adsp_base+0x12, adr);
  data[0] = inpw(adsp_base);    /* fetch bus */
  data[0] = inpw(adsp_base+0x1c); /* DMDM */
  data[1] = inpw(adsp_base+0x1a); /* DMDH */
}


/* read one DSP memory; may be inefficient if reading same place repeatedly
```

```
 * because the read address registers do not need to be updated if same
 */
void adsp_read(short adsp_base, short memtype, short adr, unsigned short *data) {
 if(memtype) {                  /* pm */
  outpw(adsp_base+0x10, adr);     /* write addr of desired read to PMADR */
  data[0] = inpw(adsp_base+0x2);  /* request ADSP bus by reading PMRD */
  data[2] = inpw(adsp_base+0x18); /* PMDL */
  data[1] = inpw(adsp_base+0x16); /* PMDM */
  data[0] = inpw(adsp_base+0x14); /* PMDH */
 }
 else {                         /* dm */
  outpw(adsp_base+0x12, adr);     /* write adr of desired read to DMADR */
  data[0] = inpw(adsp_base);      /* request ADSP bus by reading DMRD */
  data[2] = inpw(adsp_base+0x1e); /* DMDL; use word read even though we will
                 * throw away garbage byte (pad)
                 */
  data[1] = inpw(adsp_base+0x1c); /* DMDM */
  data[0] = inpw(adsp_base+0x1a); /* DMDH */
 }
}


/* Write 4 bytes data to DM */
void adsp_write_dm(short adsp_base, short adr, unsigned short *data) {
 outpw(adsp_base+0x12, adr);    /* write adr of desired write to DMADR */
 outpw(adsp_base+0x1c, data[0]); /* DMDM */
 outpw(adsp_base+0x1a, data[1]); /* DMDH */
 outpw(adsp_base, 0x01);        /* write to DMWR to initiate dm write */
}


/* write to one DSP memory; similar to adsp_read */
void adsp_write(short adsp_base, short memtype, short adr, unsigned short *data) {
 if(memtype) {
  outpw(adsp_base+0x10, adr);     /* write adr of desired write to PMADR */
  outpw(adsp_base+0x18, data[2]);
  outpw(adsp_base+0x16, data[1]);
  outpw(adsp_base+0x14, data[0]);
  outpw(adsp_base+0x2, 0x01);     /* write anything to PMWR to initiate
                 * pm write
                 */
 }
 else {
  outpw(adsp_base+0x12, adr);     /* write adr of desired write to DMADR */
  outpw(adsp_base+0x1e, data[2]);
  outpw(adsp_base+0x1c, data[1]);
  outpw(adsp_base+0x1a, data[0]);
  outpw(adsp_base, 0x01);        /* write to DMWR to initiate dm write */
```

```
  }
}


/* Sparrow device driver part of ADSP library */

/* adsp board read float device driver */
int adsp_rdfloat_driver(DEVICE *dp, CHANNEL *cp, DEV_ACTION action) {
  register int i;
  int status = -1;        /* return status */
  float fdata;

  switch(action) {
  case Init:              /* Nothing to do for initialization */
    break;
  case Read:              /* Read each of the channels in turn */
    for(i=0; i<dp->size; ++i) { /* Read floats from ADSP DM */
      adsp_read_float((short) (dp->address), (short) (dp->index+i), &fdata);
      cp[i].data.d = fdata;
    }
    status = 0;
    break;
  case Write:             /* No write action */
    break;
  case Zero:              /* No write action */
    break;
  case NewChannels:    /* no dev-sp fields supported by adsp_float */
    if(dp->index<0x0) {
      fprintf(stderr, "adsp_float: bad index value, reseting to zero\n");
      dp->index = 0x0;
    }
    if((long) (dp->index+dp->size)>0xFFFF) {
      fprintf(stderr, "adsp_float: too many channels, reseting to max\n");
      dp->size = 0xFFFF - dp->index;
    }
    status = dp->size;
    break;
  case HandleFlag:
    break;       /* no dev-sp fields supported by adsp_float */
  }
  return status;
}


/* adsp board write float device driver */
int adsp_wrfloat_driver(DEVICE *dp, CHANNEL *cp, DEV_ACTION action) {
  register int i;
  int status = -1;        /* return status */
```

```
  float fdata;          /* temporary hold for data */

  switch(action) {
  case Init:          /* Initialize hardware and reset channels */
    for(i=0; i<dp→size; ++i) {    /* reset all channel */
      fdata = 0.0;
      cp[i].data.d = 0.0;
      adsp_write_float((short) (dp→address), (short) (dp→index+i), &fdata);
    }
    status = dp→size;
    break;
  case Read:          /* No read action */
    break;
  case Write:          /* Write each channel to ADSP DM */
    for(i=0; i<dp→size; ++i) {    /* Write the data out to hardware */
      fdata = (float) cp[i].data.d;
      adsp_write_float((short) (dp→address), (short) (dp→index+i), &fdata);
    }
    status = 0;
    break;
  case Zero:          /* reset 1 chn */
    fdata = 0.0;
    cp→data.d = 0.0;
    cp→raw = cp→offset;
    adsp_write_float((short) (dp→address), (short) (dp→index+cp→chnid), &fdata);
    status = 0;
    break;
  case NewChannels:    /* no dev-sp fields supported by adsp_float */
    if(dp→index<0x0) {
      fprintf(stderr, "adsp_float: bad index value, reseting to zero\n");
      dp→index = 0x0;
    }
    if((long) (dp→index+dp→size)>0xFFFF) {
      fprintf(stderr, "adsp_float: too many channels, reseting to max\n");
      dp→size = 0xFFFF − dp→index;
    }
    status = dp→size;
    break;
  case HandleFlag:
    break;        /* no dev-sp fields supported by adsp_float */
  }
  return status;
}
```

B.2 Building the user display

The user interface can be rapidly built with Sparrow's text driven dynamic display module, which allows the user to change program parameters and issue commands from starting and stopping the motor, to capturing and saving the experimental data online with only a stroke of a key. The display screen is built from the user's explicit preference, in the form of a text file–`.dd` file describing the screen configuration, much like the following.

```
/* nanoum.dd - display file for nanometer precision ultrasonic
 * motor control program
 * HOC Aug 95
 */

%%
        %TITLE

PIEZO |   Volt       Ultrasonic Motor
------|----------   ----------------
Middle|   %v0        Position [nm]: %pos
Leg1  |   %v1        Step Size [V]: %step_size
Leg2  |   %v2

Dump file: %dumpfile

<F1> %um        <f> step forward
<F2> um zero        <b> step backward
<F5> %sim_capt
<F6> toggle capture
<F7> dump data
<F9> vscope
<F10> refresh screen
%%

tblname: nanoum;
bufname: nanoumbuf;

label:   %TITLE      "Nanometer Actuation Demo"   -fg=YELLOW;

button: %um       "START UM"    um_toggle   -fg=GREEN -idname=UMB;
button:   %sim_capt   "SIM CAPT OFF"   sim_capt_toggle -fg=GREEN -idname=SIMB;

double:   %v0       v[0]       "%.2f"   ;
double:   %v1       v[1]       "%.2f"   ;
double:   %v2       v[2]       "%.2f"   ;
double:   %pos       pos        "%.2f"   -ro;
double:   %step_size   step_size   "%.2f"   ;

string: %dumpfile    dumpfile    "%s"   ;
```

B.3 Hardware interface

Sparrow comes with hardware interface routines, along with a library of most commonly used devices. If the device at hand is in the library, all you have to do is identify it with a `.dev` file similar to the following file. On the other hand, if you are unfortunately stuck with other relatively uncommon devices, and still want to use Sparrow, you must write the device driver yourself. This is not an herculian task, because the PC-ADSP interface routine shown above is in an extended way a very complicated device driver. Consult the Sparrow manual to write a custom device driver [23].

```
# device config file for nanoum
# read position with DAS−1402, which is compatible with DAS−1602
# input is +/−10V DC from capacitive probe and my circuit
# with 2x voltage amplification:
#device: das16−adc 1 0x340 −start=1 −scale=6.2012 −offset=2048;
# with 5x voltage amplification:
#device: das16−adc 1 0x340 −start=1 −scale=1.2386 −offset=2048;
# with 5x voltage amplification:
device: das16−adc 1 0x340 −start=1 −scale=0.248282 −offset=2048;
# voltage commanded to the piezos
device: adsp_write 3 0x300 −nodump;
# save command action with virtual channel
device: virtual 1 0x0000;
```

B.4 Primary host execution

With all the low level hardware control, user interface, data capturing and dumping already managed by the Sparrow library functions, the main program controlling the experiment is only left to handle actual control routines, either openloop or closed, and call appropriate routines upon user input. A simple example program follows.

```
/* nanoum.c - ultrasonic motor control program with dsparrow
 * HOC Dec 95
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Sparrow include files */
#include "channel.h"
#include "servo.h"
#include "display.h"
#include "keymap.h"
#include "vscope.h"
#include "adsp.h"

/* ADSP memory map; refer to nanoum.ach */
#define ADSP_BASE 0x300
```

```
/* user defined DM address */
enum adsp_var_addr {SERVOHZ_ADR=0x3, SERVOPD_ADR};

/* Physical defines */
#define V0MAX 100.0
#define V0MIN 0.0
#define V1MAX 50.0
#define V1MIN 0.0


/* Local function declarations */
static int set_adsp_servo_freq(void), set_pc_servo_freq(void);
static int um_toggle(void), um_on(void), um_off(void), um_zero(void);
static int sim_capt_toggle(void), sim_capt_on(void), sim_capt_off(void);
static int cptr_toggle(void), dump_data(void), user_quit(void);
static int inc_v0(void), dec_v0(void);
static int step_forward(void), step_backward(void);
static void reset_defaults(void), user_init(void), SLoop(void);

/* global variables */
int um_flag, sim_capt_flag, step_flag, step_ctr;
double v[3], pos_cmd;     /* commanded voltage to piezos; written to ADSP directly */
double adsp_servohz, pc_servohz;
double pos, step_size;

/* Default files */
#define FNAMESIZE 128
char dumpfile[FNAMESIZE] = "um.dat";

#include "nanoum.h"        /* display table */


void main() {
 if(chn_add_driver("adsp_write", adsp_wrfloat_driver) < 0) {
  puts("Failed to add ADSP driver; Exiting...");
  exit(1);
 }
 if(chn_config("nanoum.dev") < 0) {/* init devices */
  puts("Channel Initialization Failure; Exiting...");
  exit(1);
 }
 if(!adsp_load_stk("nanoum.stk", ADSP_BASE)) { /* load ADSP program */
  puts("Failed to load stk file to ADSP; Exiting...");
  exit(1);
 }

 /* Init dynamic display */
 if (dd_open() < 0) {        /* init screen */
```

```
    puts("Display Initialization Failure; Exiting...");
    exit(1);
  }

  /* key binding for easy control */
  dd_bindkey(K_F1, um_toggle);
  dd_bindkey(K_F2, um_zero);
  dd_bindkey(K_F5, sim_capt_toggle);
  dd_bindkey(K_F6, cptr_toggle);
  dd_bindkey(K_F7, dump_data);
  dd_bindkey(K_F9, vscope);
  dd_bindkey(K_F10, dd_redraw);
  dd_bindkey(K_CNTRL_UP, inc_v0);
  dd_bindkey(K_CNTRL_DOWN, dec_v0);
  dd_bindkey('f', step_forward);
  dd_bindkey('b', step_backward);

  dd_usetbl(nanoum);        /* set display description tbl */
  user_init();              /* initialize user specified parameters */
  set_pc_servo_freq();
  dd_loop();                /* pass control to display mgr */
  dd_close();               /* clear screen and free memory when done */
  servo_cleanup();

  /* User cleanup goes here */
  user_quit();
}


/*************** main servo loop ****************/
void SLoop() {
  chn_read();
  pos = chn_data(0);

  /* Determine and output desired voltage to piezo, depending
   * on whether we are in the stepping mode; otherwise specify manually
   */
  switch(step_flag) {
  case 1:        /* Forward stepping */
    switch(step_ctr) {
    case 0:
      v[0] = 0.0; v[1] = V1MAX; v[2] = V1MAX; step_ctr++;
      break;
    case 1:
      v[0] = 0.0; v[1] = V1MAX; v[2] = 0.0; step_ctr++;
      break;
    case 2:
      v[0] = step_size; v[1] = V1MAX; v[2] = 0.0; step_ctr++;
```

```
    break;
  case 3:
    v[0] = step_size; v[1] = 0.0; v[2] = V1MAX; step_ctr++;
    break;
  case 4:
    v[0] = 0.0; v[1] = 0.0; v[2] = V1MAX; step_ctr++;
    break;
  case 5:       /* Stop and reset piezos */
    v[0] = 0.0; v[1] = V1MAX; v[2] = V1MAX;
    step_ctr = 0;
    step_flag = 0;
    break;
  }
  break;
case -1:        /* Backward stepping */
  switch(step_ctr) {
  case 0:
    v[0] = 0.0; v[1] = V1MAX; v[2] = V1MAX; step_ctr++;
    break;
  case 1:
    v[0] = 0.0; v[1] = 0.0; v[2] = V1MAX; step_ctr++;
    break;
  case 2:
    v[0] = step_size; v[1] = 0.0; v[2] = V1MAX; step_ctr++;
    break;
  case 3:
    v[0] = step_size; v[1] = V1MAX; v[2] = 0.0; step_ctr++;
    break;
  case 4:
    v[0] = 0.0; v[1] = V1MAX; v[2] = 0.0; step_ctr++;
    break;
  case 5:       /* Stop and reset piezos */
    v[0] = 0.0; v[1] = V1MAX; v[2] = V1MAX;
    step_ctr = 0;
    step_flag = 0;
    break;
  }
  break;
default:        /* User control mode */
  break;
}
pos_cmd = v[0];    /* Save commanded position/voltage w/ virtual channel */

chn_data(1) = v[0];        /* Write voltages to ADSP D/A */
chn_data(2) = v[1];
chn_data(3) = v[2];
chn_data(4) = pos_cmd;
chn_write();
```

```
}
/************** end of servo loop ***************/

static int step_forward() {    /* Signal beginning of forward step */
  step_ctr = 0;
  return step_flag = 1;
}
static int step_backward() {
  step_ctr = 0;
  return step_flag = -1;
}

static int inc_v0() {
  if(v[0] == V0MAX) {
    dd_beep(); dd_prompt("V0 at maximum allowable");
  }
  else v[0] += 1.0;
  return 0;
}
static int dec_v0() {
  if(v[0] == V0MIN) {
    dd_beep(); dd_prompt("V0 at minimum allowable");
  }
  else v[0] -= 1.0;
  return 0;
}

static void user_init() {
  pc_servohz = 100.0;
  reset_defaults();
}
static int user_quit() {
  um_zero();
  return DD_EXIT_LOOP;
}

static int um_zero() {        /* output 0 volt to all chns */
  v[0] = v[1] = v[2] = 0.0;
  step_flag = 0;        /* Turn off stepper */
  step_ctr = 0;
  if(!um_flag) {
    adsp_start(ADSP_BASE);
    adsp_stop(ADSP_BASE);
  }
  return 0;
}

static void reset_defaults() {
```

```
  um_off();       /* turn off ADSP */
  adsp_servohz = 1000.0; set_adsp_servo_freq();
  step_size = 50.0;   /* Apply 50V impulse to middle piezo */
  um_zero();
}


/* ADSP RESET bit must be held low while resetting ADSP servo frequency */
static int set_adsp_servo_freq() {
  double adsp_servopd;
  um_off();          /* Turn off ADSP; redundant in reset_defaults */
  if(adsp_servohz < 100.0) {
   dd_beep(); dd_prompt("Below min adsp servo frequency; resetting to min...");
    adsp_servohz = 100.0;
  }
  adsp_servopd = 1.0/adsp_servohz;
  return(adsp_set_user_var(ADSP_BASE, SERVOHZ_ADR, adsp_servohz) &
     adsp_set_user_var(ADSP_BASE, SERVOPD_ADR, adsp_servopd));
}

/* Servo must be disabled to reset servo frequency */
static int set_pc_servo_freq() {
  um_off();          /* Turn off ADSP */
  servo_cleanup();
  pc_servohz = servo_setup(SLoop, pc_servohz, 0);
  return servo_enable();
}

static int um_toggle() { return um_flag ? um_off() : um_on(); }
static int um_off() {
  um_zero();
  adsp_stop(ADSP_BASE);
  dd_setcolor(UMB,BLACK,GREEN);
  dd_setlabel(UMB, "START UM");
  return um_flag=0;
}
static int um_on() {
  adsp_start(ADSP_BASE);
  dd_setlabel(UMB, "STOP UM");
  dd_setcolor(UMB,BLACK,RED);
  return um_flag=1;
}

/* Capture callbacks */
static int sim_capt_toggle() { return sim_capt_flag ? sim_capt_off() : sim_capt_on(); }
static int sim_capt_off() {
  dd_setlabel(SIMB, "SIM CAPT OFF");
  dd_setcolor(SIMB,BLACK,GREEN);
```

```
  return sim_capt_flag=0;
}
static int sim_capt_on() {
  dd_setlabel(SIMB, "SIM CAPT ON");
  dd_setcolor(SIMB,BLACK,RED);
  return sim_capt_flag=1;
}

static int cptr_toggle() {
  return chn_capture_flag ? chn_capture_off() : chn_capture_on();
}

/* Dump captured data to disk */
static int dump_data() {
  int nfields;
  char str[80];
  char tmpfile[FNAMESIZE] = "c:/users/hoc/";

  chn_capture_off();        /* turn off capture */
  servo_disable();         /* turn off servo for a bit */
  dd_redraw();             /* redraw the screen */
  dd_prompt("Saving the data to the file");
  strcat(tmpfile,dumpfile);
  nfields = chn_capture_dump(tmpfile);    /* dump the actal data */
  sprintf(str, "Wrote %i data fields", nfields);
  dd_prompt(str);          /* let the user know we are done */
  servo_enable();          /* turn servo back on */
  return 0;
}
```

B.5 ADSP execution

The ADSP board runs in parallel with the host PC, handling all speed critical computations and hardware interface. The ADSP board has a pair of 32 kBytes of RAM. One is the program memory on which the executable is written during the board initialization. The other is the data memory, which can be used temporarily to store data of all kinds. With the current PC-ADSP interface routine, PC-ADSP communication takes place by reading from, and writing to predefined segments of the data memory. The ADSP board is also equipped with a status register for more direct signaling of the ADSP state to the PC. Although the Sparrow library has almost obviated the need to keep track of the PC-ADSP interface, the user must still designate the ADSP memory map and instruct the ADSP board what to do when certain commands are issued from the PC. The memory map is declared in the `.dd` architecture file, and the instructions are compiled from either C or assembly languages. Naturally, programming in C is easier than programming in ADSP assembly, but relatively large overhead associated with C codes hamper the effectiveness of the ADSP board for demanding applications. Consequently, it is a good programming practice to write em all ADSP programming ADSP assembly such as the following.

```
.SYSTEM UM;
.PROCESSOR = ADSP21020;

.SEGMENT /ROM /BEGIN=0x000000 /END=0x000007 /PM emu_svc;
.SEGMENT /RAM /BEGIN=0x000008 /END=0x00000F /PM rst_svc;
.SEGMENT /ROM /BEGIN=0x000010 /END=0x000017 /PM resrvd1;
.SEGMENT /ROM /BEGIN=0x000018 /END=0x00001F /PM sovf_svc;
.SEGMENT /RAM /BEGIN=0x000020 /END=0x000027 /PM tmzh_svc;
.SEGMENT /ROM /BEGIN=0x000028 /END=0x00002F /PM irq3_svc;
.SEGMENT /ROM /BEGIN=0x000030 /END=0x000037 /PM irq2_svc;
.SEGMENT /ROM /BEGIN=0x000038 /END=0x00003F /PM irq1_svc;
.SEGMENT /ROM /BEGIN=0x000040 /END=0x000047 /PM irq0_svc;
.SEGMENT /ROM /BEGIN=0x000048 /END=0x00004F /PM resrvd2;
.SEGMENT /ROM /BEGIN=0x000050 /END=0x000057 /PM resrvd3;
.SEGMENT /ROM /BEGIN=0x000058 /END=0x00005F /PM cb7_svc;
.SEGMENT /ROM /BEGIN=0x000060 /END=0x000067 /PM cb15_svc;
.SEGMENT /ROM /BEGIN=0x000068 /END=0x00006F /PM resrvd4;
.SEGMENT /ROM /BEGIN=0x000070 /END=0x000077 /PM tmzl_svc;
.SEGMENT /ROM /BEGIN=0x000078 /END=0x00007F /PM fix_svc;
.SEGMENT /ROM /BEGIN=0x000080 /END=0x000087 /PM flto_svc;
.SEGMENT /ROM /BEGIN=0x000088 /END=0x00008F /PM fltu_svc;
.SEGMENT /ROM /BEGIN=0x000090 /END=0x000097 /PM flti_svc;
.SEGMENT /ROM /BEGIN=0x000098 /END=0x00009F /PM resrvd5;
.SEGMENT /ROM /BEGIN=0x0000A0 /END=0x0000A7 /PM resrvd6;
.SEGMENT /ROM /BEGIN=0x0000A8 /END=0x0000AF /PM resrvd7;
.SEGMENT /ROM /BEGIN=0x0000B0 /END=0x0000B7 /PM resrvd8;
.SEGMENT /ROM /BEGIN=0x0000B8 /END=0x0000BF /PM resrvd9;
.SEGMENT /ROM /BEGIN=0x0000C0 /END=0x0000C7 /PM sft0_svc;
.SEGMENT /ROM /BEGIN=0x0000C8 /END=0x0000CF /PM sft1_svc;
.SEGMENT /ROM /BEGIN=0x0000D0 /END=0x0000D7 /PM sft2_svc;
.SEGMENT /ROM /BEGIN=0x0000D8 /END=0x0000DF /PM sft3_svc;
.SEGMENT /ROM /BEGIN=0x0000E0 /END=0x0000E7 /PM sft4_svc;
.SEGMENT /ROM /BEGIN=0x0000E8 /END=0x0000EF /PM sft5_svc;
.SEGMENT /ROM /BEGIN=0x0000F0 /END=0x0000F7 /PM sft6_svc;
.SEGMENT /ROM /BEGIN=0x0000F8 /END=0x0000FF /PM sft7_svc;

.SEGMENT /RAM /BEGIN=0x000100 /END=0x005FFF /PM pm_code;

.SEGMENT /RAM /BEGIN=0x00000000 /END=0x00000FFF /DM user_var;
.SEGMENT /RAM /BEGIN=0x00001000 /END=0x00001FFF /DM int_var;

.SEGMENT /PORT /BEGIN=0X800000   /END=0X800000   /PM ioadin;
.SEGMENT /PORT /BEGIN=0X800001   /END=0X800001   /PM iostat;

.SEGMENT /PORT /BEGIN=0X20000000 /END=0X20000000 /DM status;
.SEGMENT /PORT /BEGIN=0X20000001 /END=0X20000001 /DM timer;

.SEGMENT /PORT /BEGIN=0X40000000 /END=0X40000000 /DM iodaout;
```

```
.SEGMENT /PORT /BEGIN=0X40000001 /END=0X40000001 /DM iocntrl;
.SEGMENT /PORT /BEGIN=0X40000002 /END=0X40000002 /DM iochans;

.BANK /PM0 /WTSTATES=0 /WTMODE=INTERNAL /BEGIN=0X000000;
.BANK /DM0 /WTSTATES=0 /WTMODE=INTERNAL /BEGIN=0X00000000;
.BANK /DM1 /WTSTATES=1 /WTMODE=INTERNAL /BEGIN=0X40000000;

.ENDSYS;
/* nanoum.asm - low level ADSP board control for nanometer precision
 *      movement demo; This program simply reads the voltages
 *      commanded by the host PC real time control program (nanoum.c)
 *      and outputs it to the D/A
 */

#include "defmacro.h"

/* The following ports are found on the ADSP board */
.SEGMENT /DM status;
.VAR DSPSTAT;
.ENDSEG;

.SEGMENT /DM timer;
.VAR DSPTIMER;
.ENDSEG;

/* The following ports are found on the 32-channel ADC board */
.SEGMENT /PM ioadin;
.VAR ADFIFO;
.ENDSEG;

.SEGMENT /PM iostat;
.VAR IOSTAT;
.ENDSEG;

.SEGMENT /DM iodaout;
.VAR DAFIFO;
.ENDSEG;

.SEGMENT /DM iochans;
.VAR CHANNELS;
.ENDSEG;

.SEGMENT /DM iocntrl;
.VAR CONTROL;
.ENDSEG;


#define CPUHZ    33333333.333    /* Timer constants */
```

```
#define DANUM    3
#define ADNUM    0

#define V0MAX 100.0        /* maximum voltage allowed on each piezo */
#define V0MIN 0.0
#define V1MAX 50.0
#define V1MIN 0.0

#define DA0CONV 327.67        /* D/A constants */
#define DA1CONV    630.13
#define DA2CONV    630.13
#define V0OFFSET 0.07
#define V1OFFSET 0.06
#define V2OFFSET 0.075

.SEGMENT /DM user_var;    /* Declare user supplied variables */
.VAR V0;
.VAR V1;
.VAR V2;
.VAR SERVOHZ;        /* servo freq and period are reciprocals of one another */
.VAR SERVOPD;
.ENDSEG;

.SEGMENT /DM int_var;    /* Declare internally used variables */
.VAR DA0;        /* voltage commanded to D/A */
.VAR DA1;
.VAR DA2;
.ENDSEG;


/* Toggling RESET bit of ADSP CNTL register resets the ADSP chip
 * Upon reset, program starts at the rst_svc
 */
.SEGMENT /PM rst_svc;
   PMWAIT = 0x0010C2;
   DMWAIT = 0x00431842;

/* Set FLAG2 to output mode by setting bit 17 of the MODE2 register
 * so we can trigger I/O board conversion when we need to
 */
   MODE2 = 0x00020000;
   JUMP initialize;

.ENDSEG;

.SEGMENT /PM pm_code;    /* program code segment */
```

```
/* Initialize registers and internal variables and setup timer interrupt and I/O board
 * User variables need to be initialized externally before execution command is given
 */
initialize:
   R6 = 0;          /* Signal that we are in initialization */
   DM(DSPSTAT) = R6;

   IMASK = 0x0;       /* init regs */
   MODE1 = 0x00012000;
   BIT CLR ASTAT 0x00200000;/* clear FLAG 2 to stop conversion */
   NOP; NOP; NOP;

   R0 = 0x0;        /* reset io board */
   DM(CONTROL) = R0;
   NOP; NOP; NOP;

   R0 = DANUM;        /* Set A/D and D/A channel numbers */
   R1 = ADNUM;
   R0 = LSHIFT R0 BY 3;
   R0 = R0+R1;
   DM(CHANNELS) = R0;
   NOP; NOP; NOP;

   R0 = 0x80;        /* start io board w/ GOMODE 0, IRMODE 0, STSEL 0(?) */
  DM(CONTROL) = R0;   /* ADSP board manual says STSEL should be 1, but WRONG
*/
   NOP; NOP; NOP;

   F0 = CPUHZ;        /* Setup servo sampling rate */
   F12 = DM(SERVOHZ);    /* TPERIOD = CPUHZ/SERVOHZ - 1 */
   DIV;
   R0 = FIX F0;
   R0 = R0-1;
   TPERIOD = R0;
   TCOUNT = R0;

   F0 = 0.0;        /* First values to be output to D/A */
   R0 = FIX F0;
   R9 = 0x8000;
   R0 = R0 XOR R9;
   DM(DAFIFO) = R0;
   DM(DAFIFO) = R0;
   DM(DAFIFO) = R0;

   BIT SET IRPTL 0x0;    /* Reset interrupt latch register */
   BIT SET IMASK 0x12;    /* Allow timer interrupts; only rst_svc
            * and tmzh_svc are recognized here
            */
```

```
   BIT SET MODE2 0x20;    /* Enable timer */
   BIT SET MODE1 0x1000;    /* Global interrupt enable */

gag:   IDLE;            /* wait for timer interrupt */
   JUMP gag;

/************************** Main servo loop ************************/
sloop:
   STARTCONV;        /* start D/A and A/D conversion */
/* Do usefule calculation while waiting for conversion */
/* Get 3 voltage signals for next round of D/A conversion */
   F3 = DM(V0);        /* Drive voltage to the middle piezo */
   F0 = V0OFFSET;        /* Subtract constant voltage offset value */
   F3 = F3-F0;
   F0 = DA0CONV;        /* chn0 scale factor: DABITSMAX/V0MAX */
   F4 = F3*F0;        /* DABITS0 = DABITSMAX*V0/V0MAX */
   DM(DA0) = F4;        /* Output this to DA0, which feeds NF amplifier */
   F3 = DM(V1);        /* Drive voltage to the leg1 */
   F0 = V1OFFSET;
   F3 = F3-F0;
   F0 = DA1CONV;
   F4 = F3*F0;
   DM(DA1) = F4;
   F3 = DM(V2);        /* Drive voltage to leg2 */
   F0 = V2OFFSET;
   F3 = F3-F0;
   F0 = DA2CONV;
   F4 = F3*F0;
   DM(DA2) = F4;

/* Wait for I/O conversion */
   R1 = 0x03;        /* check if A/D *and* D/A are busy */
wait:   R0 = PM(IOSTAT);
   R7 = R0 AND R1;
   IF NE JUMP wait;

/* Shove output values out to D/A FIFO */
   B0 = DA0;
   LCNTR = DANUM;
   DO daout UNTIL LCE;
    F0 = DM(I0,1);        /* Get desired D/A output, Vda */
    R0 = FIX F0;
/*    R0 = ASHIFT R0 BY 2;*/
    R6 = 0x7FFF;        /* Check if saturate */
    R0 = CLIP R0 BY R6;
offsetbin:    /* change from 2's complement to offset binary form by toggling bit 15 */
    R9 = 0x8000;
    R0 = R0 XOR R9;
```

```
daout:    DM(DAFIFO) = R0;

   RTI;          /* return and wait for next timer interrupt */
/************************ End of the main servo loop ************************/


.ENDSEG;


.SEGMENT /PM tmzh_svc;
   JUMP sloop;
.ENDSEG;
```

B.6 Matlab source codes

Matlab analysis code for the ultrasonic motor peg vibration:

```
% matlab file for UM leg vibration analysis without mass consideration
% define constants
n = 87*9.86/400;% piezo constant [N/V]
d33 = .15;    % displacement at [um/V]
k = n/d33;    % [N/um]
F0 = 0.0;    % normal pressing force [N]
Voff = 25;    % [V]
Vamp = 25;    % [V]
f = 1000;    % [Hz]
T = 1/f;
w = 2*pi*f;

t = linspace(0,T);
v1 = Vamp*sin(w*t);
v2 = -v1;
x1 = d33*v1; x1 = sat(x1,0,'+');
x2 = d33*v2; x2 = sat(x2,0,'+');
f1 = n*v1; f1 = sat(f1,0,'-');
f2 = n*v2; f2 = sat(f2,0,'-');

subplot(311); plot(t,v1, t,v2); title('drive volt')
subplot(312); plot(t,x1, t,x2); title('piezo displacement');
subplot(313); plot(t,f1, t,f2); title('piezo force');

% Plot peg mode shapes
k = [1.875 4.694 7.855 10.995];    % first 4 wave #s
x = linspace(0,1);    % let peg go from 0 to 1
phi = zeros(100,4);
for i=1:100
  for j=1:4
    phi(i,j) = cosh(k(j)*x(i))-cos(k(j)*x(i))-(cosh(k(j))+cos(k(j)))*\
```

```
      (sinh(k(j)*x(i))-sin(k(j)*x(i)))/(sinh(k(j))+sin(k(j)));
  end
end
plot(x,phi(:,1),x,phi(:,2),x,phi(:,3),x,phi(:,4));
title('Leg mode shapes');
xlabel('normalized position (x/L)');
gtext('n=1')
```

Matlab analysis code for obtaining a real time causal velocity filter from position measurement:

```
% velocity filter design
% HOC Nov 95

fsample=2000;  %Sampling frequency in Hz
fro=500;       %LP cutoff freq in HZ
fco=500;       %Differentiator Roll-Of in Hz
lpord=3;       %order of lowpass filter

fprintf(1,'Calculating Velocity Filter for %g Hz sampling rate\n',fsample);
fprintf(1,'Nyquist Frequency is %g rad/sec\n',fsample*pi);

nums=[1 0];
dens=[1 2*pi*fco];
[numd,dend]=c2dm(nums,dens,1/fsample,'zoh');

[numl,denl]=butter(lpord,2*fro/fsample);

numz=conv(numd,numl);
denz=conv(dend,denl);

fprintf(1,'Discrete Time Poles:\n')
roots(denz)

%Normalize frequency to 1;
[mag,phase]=dbode(numz,denz,1/fsample,1);
numz=numz/mag;


dbode(numz,denz,1/fsample);
pause;
w=logspace(-1,3,100);
dbode(numz,denz,1/fsample,w);

pause;
[A,B,C,D]=tf2ss(numz,denz);
dimpulse(A,B,C,D);
title('Discrete Filter Impulse Response');
% file for the um analysis
servopd = 0.001; servohz = 1000;
```

```
% off-line filter the position
% first look at the signal frequency content
psd(x,[],servohz,[]);

% After looking at the PSD curve, decide on
Fp=40; Fs=100;
nb = buttord(2*Fp/servohz,2*Fs/servohz,3,40)
[nump,denp] = butter(nb,2*Fp/servohz);
x_f = filtfilt(nump,denp,x);

% Now get the velocity estimation by bilnear difference
v = (servohz/2)*(x(3:n)-x(1:n-2));
v_f = (servohz/2)*(x_f(3:n)-x_f(1:n-2));
subplot(221);psd(v,[],servohz,[]);title('PSD');xlabel('');ylabel('');
subplot(222);plot(t(2:n-1),v);title('unfiltered velocity');ylabel('v [mm/s]');
subplot(223);psd(v_f,[],servohz,[]);xlabel('Freq [Hz]');ylabel('');
subplot(224);plot(t(2:n-1),v_f);title('filtered velocity');xlabel('t [s]');

subplot(211);plot(t,x, t,x_f,'--');
title('start/stop movement');ylabel('x [mm]');
subplot(212);plot(t(2:n-1),v_f, t,mean(v(1300:2700))*um210(:,2),'--');
title('filtered velocity');xlabel('t [s]');ylabel('v [mm/s]');

% load .dat file
x_f = filtfilt(nump,denp,x);
% velocity estimation based on low pass filtering
% on the raw velocity
Fp=40; Fs=60;
nb = buttord(2*Fp/servohz,2*Fs/servohz,3,40)
[numv,denv] = butter(nb,2*Fp/servohz);
v = (servohz/2)*(x(3:n)-x(1:n-2));
v_f = filtfilt(numv,denv,v);
%plot(t,x_f, t(2:n-1),v_f);
plot(i,x_f, i(2:n-1),v_f);

% Manually recorded the Vapp-vmax chart
vv = [
5 4.0842;
10 6.7529;
15 9.0938;
20 12.5081;
25 15.4615;
30 18.1193;
35 21.5678;
40 24.4524;
45 28.0017;
50 30.4449
```

```
];
plot(vv(:,1),vv(:,2))
title('Applied voltage vs. max speed')
xlabel('Vapp [V]'),ylabel('max speed [mm/s]')


i_start = 6895;
i_stop = 7129;
speed = servohz*25.4/(i_stop-i_start)

% Manually record freq-v_max chart
fv = [
200 2.9767;
400 8.4723;
600 13.1811;
800 18.6354;
1000 23.6059;
1400 34.1398;
1600 44.0972;
1800 65.1282;
2000 77.2036;
2200 93.0403;
2400 107.6271;
2600 125.7426;
3000 137.9540;
3200 254.0;
3400 279.1209
];
plot(fv(:,1),fv(:,2));
title('Wave freq. vs max speed')
xlabel('Excitation freq [Hz]')
ylabel('max speed [mm/s]');

% load nano*.dat
x = nano13(:,1);
n = length(x);
i = 0:n-1;
t = servopd*i;
subplot(224); plot(t,x);gtext('Vapp = 1V');
subplot(223);xlabel('t [s]');ylabel('x [nm]');
```

B.7 Finite Element Analysis source code

The FEA program was used to predict the dynamics of the ultrasonic motor peg which is a composition of aluminum-piezo stack. Only the input code to the program is given.

```
*HEADING
Natural Freq
```

```
*NODE
1,0.0,0.0
6,0.005,0.0
21,0.0,0.004
26,0.005,0.004
71,0.0,0.009
76,0.005,0.009
101,0.0,0.027
106,0.005,0.027
*NGEN,NSET=A1
1,6
*NGEN,NSET=A12
21,26
*NGEN,NSET=A2
71,76
*NGEN,NSET=A3
101,106
*NFILL
A1,A12,2,10
A12,A2,5,10
A2,A3,3,10
*ELEMENT,TYPE=CPS4
1,1,2,12,11
81,71,72,82,81
*ELGEN,ELSET=EL1
1,5,1,1,7,10,10
*ELGEN,ELSET=EL2
81,5,1,1,3,10,10
*ELSET,ELSET=ALL
EL1,EL2
*SOLID SECTION,ELSET=EL1,MATERIAL=JUNK1
0.005
*SOLID SECTION,ELSET=EL2,MATERIAL=JUNK2
0.005
*MATERIAL,NAME=JUNK1
*ELASTIC
55.E9,0.31
*DENSITY
8.1E3
**ELASTIC
**70.E9,0.34
**DENSITY
**2.8E3
*MATERIAL,NAME=JUNK2
*ELASTIC
55.E9,0.31
*DENSITY
8.1E3
```

```
*BOUNDARY
A1,1
A1,2
*ELSET,ELSET=EL3
25,75,10
*ELSET,ELSET=ALL
EL1,EL2,EL3
*RESTART,WRITE,FREQ=100
*STEP
*STATIC
*DLOAD
EL3,P2,20000000
*EL PRINT,ELSET=ALL,FREQ=100
COORD
S
E
*EL FILE,ELSET=ALL,FREQ=100
COORD
S
E
*ENDSTEP
*STEP
*FREQUENCY
5,1000000.
*ENDSTEP
```