Design and Prototype of an Automated System for Commercially Viable Production Using Micro Contact Printing

by

Karan Chauhan
M.Eng in Manufacturing
Massachusetts Institute of Technology, 2006

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Manufacturing
at the
Massachusetts Institute of Technology
August 18, 2006

Copyright 2006 Massachusetts Institute of Technology. All rights reserved

Signature of the Author .....
Department of Mechanical Engineering
August 15, 2006

Certified by .....
David E. Hardt
Professor, Mechanical Engineering
Thesis Supervisor

Accepted by .....
Lallit Anand
Chairman, Departmental Committee on Graduate Studies
Department of Mechanical Engineering

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JAN 2 3 2007
LIBRARIES

BARKER

Design and Prototype of an Automated System for Commercially Viable Production Using Micro Contact Printing

by

Karan Chauhan

Submitted to the Department of Mechanical Engineering on August 16th, 2006 in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Manufacturing

Abstract

This dissertation is a documentation of the thought process, its justification and the implementation details that went into prototyping a fully automated system employing Micro Contact Printing, an emerging technique for replication and pattern transfer using soft organic materials, for a nano technology based company in Cambridge, Massachusetts during the summer of 2006. The thesis takes the reader through an extensive literature review of the suite of soft lithography techniques - Near Field Soft Optical Lithography, Replica Molding and Micro contact printing - focused on the critical factors for taking these techniques from research facilities to commercial manufacturing facilities. Micro Contact Printing was studied more extensively than the others. We propose a system level design for viable manufacturing using micro contact printing and also present the implementation details of an automated prototype for the same.

Acknowledgements

I would like to dedicate this thesis to my teammates in this project - Hyung Jun Kim & Arthur Cao Yuan (my colleagues in the M.Eng program), Prof David Hardt (Professor, MIT) and Brian Mayers (Senior Scientist, Nano-Terra, LLC). I think we coordinated closely as friends to finish an ambitious project within the limited time.

Prof Hardt was with us whenever we needed advise and also took a most important role of monitoring our progress. The long and rather informal weekly meetings with him always provided us with new ideas as well as deepened our commitment to the success of this project and insight into the discipline of manufacturing. I take this opportunity to congratulate him for this new degree program at MIT and sincerely wish the future batches my best.

I also thank Brian and the rest of the Nano-Terra team for providing a very friendly and supportive atmosphere at Nano-Terra. I appreciate the enormous amount of trust they showed in us and our decisions throughout the project. Without Brian acting as the crucial link between science and engineering, it could have been hard to realize our project goals.

Table of Contents

Abstract 1

Acknowledgements 2

List of Figures 6

Chapter 1

Introduction and Soft Lithography 9

1.1 An Introduction to Soft Lithography 9

1.2 The Soft Lithography Taxonomy 10

1.3 Soft Lithography versus photolithography 15

1.4 Overview of the Thesis 16

1.5 Conclusions 17

Chapter 2 18

Near Field, Phase-Shifting, Soft Optical Lithography .....

2.1 Introduction 18

2.2 Physical principles of the process 19

2.3 A Process Modeling Framework for NFPSOL 22

2.3.1 The input and output parameters of interest 22

2.3.2 Sources of disturbance and variation 24

2.3.3 Attempts on Process Modeling 25

2.4 The Steps Involved in the Process 26

2.5 Comparison with Current Lithography Practices 27

2.5.1 Advantage 27

2.5.2 Currently Unsolved Issues and Challenges 28

2.6 Conclusions 29

Chapter 3

Micro contact Printing 30
3.1 Introduction 30
3.2 Principles and characteristics of Micro contact Printing 30
3.2.1 Formation of Self Assembled Monolayers or SAMs 31
3.3 Micro contact printing Procedure 33
3.4 Critical Factors for μCP 34
3.5 Summary of manufacturing considerations 41
3.6 Industrial design efforts for Micro contact printing 43
3.7 Conclusions 44

Chapter 4
Concept Design and Realization 45
4.1 Introduction 45
4.2 The Inking Process 46
4.2.1 Designs for Inking 47
4.3 The Stamping Process 50
4.3.1 Designs for Stamping 51
4.4 A Manufacturing System for High Volume Production 61
4.5 Conclusions 63

Chapter 5
Automation Hardware and Software 64
5.1 Introduction 64
5.2 The Automation Approach 64
5.3 The Mechanical Elements for Automation 65
5.3.1 The XY-Manipulators 66
5.3.2 The Rotary Positioning Table 68
5.3.3 The 2 Lead Screws for Inking and Lifting 68
5.4 The Electronic and Software Elements for Automation 69
5.4.1 The Motors and Motor Drivers 69
5.4.1.1 Microstepping 74

5.4.2 Motor Driver-Controller Interface 74
5.4.3 NI-Motion Architecture 74
5.4.3.1 NI Motion Controller Architecture 76
5.4.3.2 NI Motion Controller Functional Architecture 75
5.5 Kinematic Analysis of the Motion 78
5.6 Execution of the Application 96
5.7 Conclusions 96

Chapter 6
6.1 Conclusions and Future Work 97

List of References 99

Appendix I 101

List of Figures

Figure 1.2.1: Schematic illustration of the procedure for fabricating PDMS stamps from a master having relief structures on its surface. 10

Figure 1.2.2: Schematic procedures for μCP of hexadecanethiol (HDT) on the surface of gold: (a) printing on a planar surface with a planar stamp. 11

Figure 1.2.3: Schematic procedures for replica molding. 12

Figure 1.2.4: Schematic procedures for MIMIC. 12

Figure 1.2.5: Schematic procedures for *μTM.* 13

Figure 1.2.6: Schematic procedures for SAMIM. 14

Figure 2.1: Some applications of NFPSOL. 19

Figure 2.2: The concept of Phase Shifting Optical Lithography to increase resolution. 20

Figure 2.2.2: The optical phenomenon behind NFPSOL. 20

Figure 2.3.1: A typical feature developed with NFPSOL. 22

Figure 2.3.1: Variation of the output (Line width) with phase shifter depth, wavelength content and edge spacing on the relief face of the mask. 23

Figure 2.3.3: Scalar diffraction theory. 24

Figure 2.3.4: Full Vector Finite Element Method. 24

Figure 2.4: The Process Steps involved in NFPSOL. 26

Figure 3.2.1: Schematic illustration depicting the application of a PDMS stamp containing thiols to a polycrystalline metal film. 30

Figure 3.2.1. Self assembled monolayers. 31

Figure 3.3: Micro contact printing procedure [36]. 32

Figure3.4.1: Relationship between thicknesses printed with 0.2 milli Mole solution of ECT ink. 34

Figure 3.4.2: Relationship between printing time and the number of defects in the pattern. 34

Figure 3.4.3: Quality of Micro contact printed gold structures performed in the case of an ECT contact-inked stamp. 35

Figure 3.4.4: Illustrations of (a) basic geometry of a PDMS stamp and stamps deformed into contact with substrates under (b) required minimum and (c) excess pressures. 36

Figure 3.4.5.1: Three different types of inking for Micro contact printing. 37

Figure 3.4.5.2: Three contact propagation methods. 38

Figure 3.4.5.3: Schematic illustration of possible deformations of microstructures in the surfaces of elastomers 40

Figure 3.6.1: Schematic procedure for conducting μCP with a rolling PDMS stamp. 41

Figure 3.6.2: A vertical Cross-section of the wave printing prototype. 42

Figure 4.2.1.2: The basic concept of polygonal inking. 47

Figure 4.2.1.3: The finalized inking design. 48

Figure 4.3.1.1.A Roll Stamping. 50

Figure 4.3.1.1.B Elastic strain induced in the stamp with Roll Stamping. 51

Figure 4.3.1.2.A: The basic concept of polygonal stamping. 52

Figure 4.3.1.2.B: The stamping tool proposed for Polygonal Stamping. 52

Figure 4.3.1.2.C: The hinge effect of the polygonal stamping tool. 53

Figure 4.3.1.3: An illustration of the steps in Pseudo Roll Stamping. 54

Figure 4.3.1.4: Two foam covered cylindrical wipers apply the gold coated PEN sheet onto the PDMS stamp. 58

Figure 4.3.1.5: A step by step illustration of the stamping design developed. 58

Figure 4.4.1: A schematic illustration of a process centric manufacturing system 61

showing the integration between the inking-stamping steps and etching.

Figure 4.4.2: Another version of the manufacturing system discussed before. 62

Figure 5.4: A device level schematic of the hardware-software integration for automation. 68

Figure 5.4.1: Speed-torque characteristics for the Motor. 70

Figure 5.4.2: The basics of a stepper motor system. 71

Figure 5.4.2: The layout of the acquired driver MD2S-P. 72

Figure 5.4.3: NI Motion Control Hardware and Software Interaction. 74

Figure 5.4.3.1: Physical NI Motion Controller Architecture. 75

Figure 5.4.3.2: Typical NI 73xx Motion Controller Functional Architecture. 77

Figure 5.5.A: A simplified illustration of the elements involved in the stamping step. 79

Figure 5.5.B: Kinematic analysis for Move 1. 80

Figure 5.5.1.1: The calculated angular speed of the supply spool vs time. 81

Figure 5.5.1.2: The computed and used electronic gear ratios. 82

Figure 5.5.2.2: The calculated angular speed vs time for the SUPPLY axis for Move 2. 84

Figure 5.5.2.2: The computed and used electronic gear ratios. 84

Figure 5.5.3.1: The calculated angular speed of the DRIVE axis vc time for Move 4. 87

Figure 5.5.3.2: The computed and used electronic gear ratios. 88

Figure 5.5.4.1: The calculated angular speed vs time for the DRIVE axis for Move 6. 90

Figure 5.5.4.2: The computed and used electronic gear ratios. 91

Figure 5.5.5.1: The calculated angular speed vs time for the DRIVE axis for Move 8. 93

Figure 5.5.5.2: The computed and used electronic gear ratios. 94

Chapter 1

Introduction and Soft Lithography

1.1 An Introduction to Soft Lithography

Soft Lithography comprises a set of techniques that uses soft materials to enable replication and pattern transfer on a wide range of dimensional scales, ranging from nanometers to centimeters. These techniques follow a non-photolithographic strategy for pattern transfer based on self-assembly[1] of molecular layers and replica molding for carrying out micro and nanofabrication.

Most Soft Lithography techniques have been recently developed and have attracted significant attention from both academia and industry due their tremendous potential to support or even replace conventional means of micro manufacturing. The low capital costs and potential for high volume manufacturing with a variety of materials are significant attractions as well. The control over surface chemistry, required for some applications in medicine, is possible if using soft lithography. Other potential applications of soft lithography in the near future could include simple optical devices such as polarizers, filters, wire grids, and surface acoustic wave (SAW) devices [1]. Longer term goals include working toward optical data storage systems, flat panel displays, and quantum devices [2].

[1] Self-assembly is the spontaneous aggregation and organization of subunits (molecules or meso-scale objects) into a stable, well defined structure via non covalent interactions. The information that guides the assembly is coded in the properties (e.g. topologies, shapes, and surface functionalities) of the subunits; the individual subunits will reach the final structure simply by equilibrating to the lowest energy form. Because the final self assembled structures are close to or at thermodynamic equilibrium, they tend to form spontaneously and to reject defects.

1.2 The Soft Lithography Taxonomy

Several different techniques are known collectively as soft lithography. Holistically, every soft lithography technique formally consists of three steps:

(1) Fabrication of a topographically patterned master, for example on a silicon wafer, using a conventional process like photolithography.

(2) Molding this master with a functional organic material (usually Polydimethyl siloxane or PDMS) to generate a patterned template.

(3) Generating a replica of the original template in a functional material or a 1:1 projection of the pattern on a surface by applying the stamp.

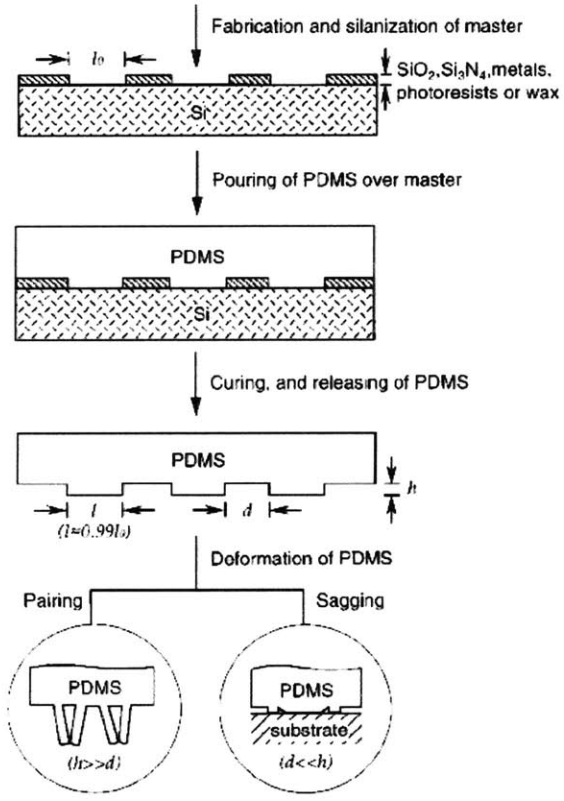


Figure 1.2.1: Schematic illustration of the procedure for fabricating PDMS stamps from a master having relief structures on its surface [3].

The techniques which are collectively known as Soft Lithography techniques are:

(A) Near Field Optical Lithography

A transparent PDMS mask with relief on its surface is placed in conformal contact with a layer of photo resist. Light, from a source, passing through the stamp is modulated in the near-field. If the relief on the surface of the stamp shifts the phase of light by an odd multiple of π, a null in the intensity is produced. Features with dimensions between 40 and 100 nm are produced in photo resist at each phase edge [4]. This process shall be discussed in detail in Chapter 2.

(B) Micro-contact Printing(μCP)

A thin layer composed of an alkanethiol and ethanol, called "ink" is spread on a patterned PDMS stamp. The stamp is then brought into conformal contact with the substrate, which can range from coinage metals to oxide layers. The thiol ink is transferred to the substrate where it forms a self-assembled monolayer, or SAM[2], that can act as a resist against etching. Features on the substrate are revealed after etch treatment. Features as small as 300 nm have been produced in this way [5]. This process shall be discussed in detail in Chapter 3.

[2] SAMs can be easily prepared by immersion of a substrate in the solution containing a ligand ($Y(CH_2)_nX$) reactive toward the surface, or by exposure of the substrate to the vapor of a reactive species.

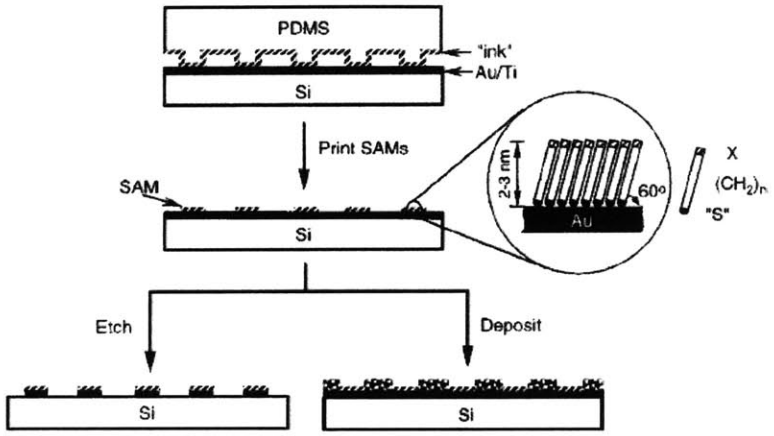


Figure 1.2.2: Schematic procedures for μCP of hexadecanethiol (HDT) on the surface of gold: (a) printing on a planar surface with a planar stamp [11].

(C) Replica Molding

A PDMS stamp is cast against a conventionally patterned master. Polyurethane is then molded against the secondary PDMS master. In this way, multiple copies can be made without damaging the original master. The technique can replicate features as small as 30 nm [6].

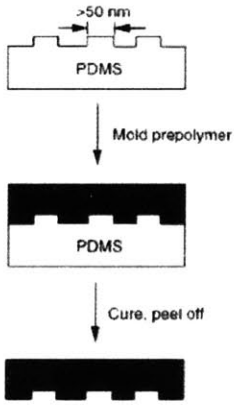


Figure 1.2.3: Schematic procedures for replica molding [4].

(D) Micro molding in Capillaries (MIMIC).

Continuous channels are formed when a PDMS stamp is brought into conformal contact with a solid substrate. Capillary action fills the channels with a polymer precursor. The polymer is cured and the stamp is removed. MIMIC is able to generate features down to 1 μm in size [7].

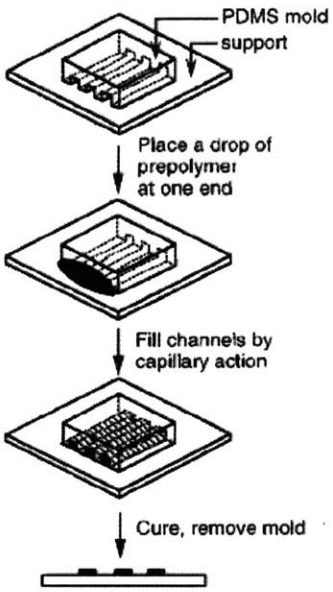


Figure 1.2.4: Schematic procedures for MIMIC [4].

(E) Micro transfer Molding (μTM)

A PDMS stamp is filled with a prepolymer or ceramic precursor and placed on a substrate. The material is cured and the stamp is removed. The technique generates features as small as 250 nm and is able to generate multilayer systems [8].

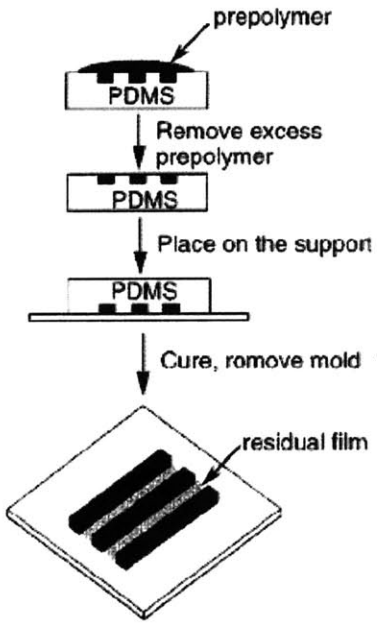


Figure 1.2.5: Schematic procedures for μTM [4].

(F) Solvent-assisted Micro contact Molding (SAMIM).

A small amount of solvent is spread on a patterned PDMS stamp and the stamp is placed on a polymer, such as photo resist. The solvent swells the polymer and causes it to expand to fill the surface relief of the stamp. Features as small as 60 nm have been produced [9].

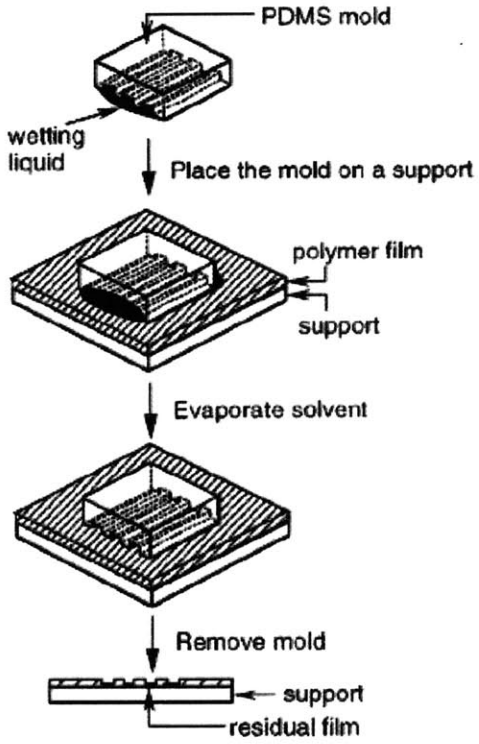


Figure 1.2.6: Schematic procedures for SAMIM [4].

1.3 Soft Lithography versus photolithography

Soft Lithography techniques not only promise a significantly cheaper alternative to the currently employed photolithography techniques, but also allow for more flexibility in product choices (for example, manufacturing structures on a non-planar surface, controlling chemistries of a surface etc), resolutions and feature sizes.

Comparison between photolithography and soft lithography

	Photolithography	Soft lithography
Definition of patterns	Rigid photomask (patterned Cr supported on a quartz plate)	Elastomeric stamp or mold (a PDMS block patterned with relief features)
Materials that can be patterned directly	Photoresists (polymers with photosensitive additives)	Photoresists[a-e]
	SAMs on Au and SiO_2	SAMs on Au, Ag, Cu, GaAs, Al, Pd, and SiO_2[a] Unsensitized polymers[b-e] (epoxy, PU, PMMA, ABS, CA, PS, PE, PVC) Precursor polymers[c,d] (to carbons and ceramics) Polymer beads[d] Conducting polymers[d] Colloidal materials[a,d] Sol-gel materials[c,d] Organic and inorganic salts[d] Biological macromolecules[d]
Surfaces and structures that can be patterned	Planar surfaces 2-D structures	Both planar and nonplanar Both 2-D and 3-D structures
Current limits to resolution	~250 nm (projection) ~100 nm (laboratory)	~30 nm[a,b], ~60 nm[e], ~1 μm[d,e] (laboratory)
Minimum feature size	~100 nm (?)	10 (?) - 100 nm

[a-e]Made by (a) μCP, (b) REM, (c) μTM, (d) MIMIC, (e) SAMIM. PU:polyurethane; PMMA: poly(methyl methacrylate); ABS: poly(acrylonitrile-butadiene-styrene); CA: cellulose acetate; PS: polystyrene; PE: polyethylene; and PVC: poly(vinyl chloride)

Table 1.3: A tabular comparison between Soft Lithography and conventional optical lithography [10].

1.4 Overview of the Thesis

This thesis is a documentation of the thought process, its justification and the implementation details that went into prototyping an automated micro contact printing system for a nano technology based company in Cambridge MA during the summer of 2006. The objectives of this project were:

1. To understand and evaluate the capability for high volume production using soft lithography techniques based on the current state of knowledge about the

processes and ongoing industrial and laboratory efforts. The techniques studied were near field soft optical lithography, replica molding and micro contact printing.

2. To conceptualize a manufacturing system built upon micro contact printing, to achieve economically viable production.
3. Based on a feasible manufacturing system design, implement a prototype that demonstrates automated inking, stamping and the tool/substrate transfer for micro contact printing.

Chapter 2 describes Near Field Phase Shifting Soft Optical Lithography- a soft lithography process- that was studied for feasible manufacturing. Chapter 3 describes the principles of micro contact printing, the current state of the art and critical factors for manufacturing including some feasible designs. Chapter 4 describes the overall design of the prototype that was developed. Chapter 5 provides details about the hardware and software integration to achieve automation in the prototype. Chapter 6 concludes this thesis with scope for future work. The C/C++ source code for the software application and references can be found in the appendices.

1.5 Conclusions

The chapter introduced the reader to the scope and objectives of this project. A brief overview of the various soft lithography techniques was also presented. The next chapter will discuss one of the soft lithography techniques called Near Field Soft Optical Lithography in greater depth.

Chapter 2

Near Field, Phase-Shifting, Soft Optical Lithography

2.1 Introduction

This chapter discusses Near Field Optical Lithography - an innovative optical lithography technique, which falls within the realms of soft lithography. The technology uses patterned elastomeric PDMS masks in conformal contact with a layer of photo resist to selectively expose (through the pattern on the mask) the photo resist to light and cure it before conventional chemical etching to generate edged patterns. In a laboratory setup, this approach requires only a phase mask and a handheld UV lamp to generate structures with dimensions as small as 50 nm over large areas (many cm^2, limited only by the size of the mask)[12]. The technique has been applied to make nano wires, micro rings and arbitrary shapes as shown in figure 2.1.

To distinguish from other forms of optical lithography (which also use phase shifters) we shall refer to this process henceforth as "Near Field, Phase Shifting, Soft optical Lithography" or as NFPSOL in the abbreviated form.

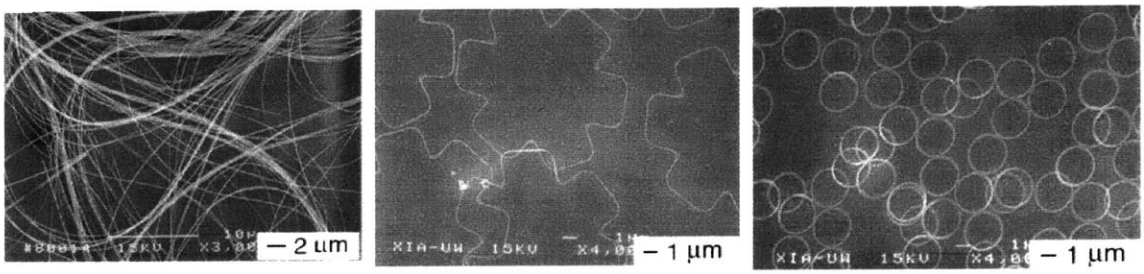


Figure 2.1: Some applications of NFPSOL [13].

First, this chapter discusses the physical principles, the process steps involved to produce parts, the current state of development of this technology and possible applications. Then, the technology is critically compared on various parameters to the commercially established technologies for optical lithography. The chapter concludes with an analysis on the feasibility of high volume production using this technique along with some system and process level concept proposals.

2.2 Physical principles of the process

Optical lithography exploits the behavior of certain photosensitive polymers (called photo-resists) when exposed to light. Some photosensitive polymers, by the nature of their chemical composition, may absorb the light and harden (or cure) while others may disintegrate. A layer of such a polymer is uniformly applied onto a substrate, selectively exposed to light through an optical system including a lens and a mask and the uncured polymer is removed. Then, during the chemical etching of the substrate, the cured polymer protects the covered portions of the substrate to result in an image of the pattern on the substrate.

There exist several ways in which patterns on a mask could be transferred onto the photo resist and then the substrate. The conventional forms of lithography use a hard mask with a chrome coat to block incident light selectively onto a layer of photo resist and then develop the photo resist.

The use of relief features on masks is not new. Phase-shifting optical lithography with hard masks in the far field has been commercially employed to achieve very high resolutions [14]. The intensity of the light at the substrate can be modulated by appropriately inserting optical media and thereby changing the optical path length and the phase difference. Figure 2.2 explains the concept of phase shifting lithography.

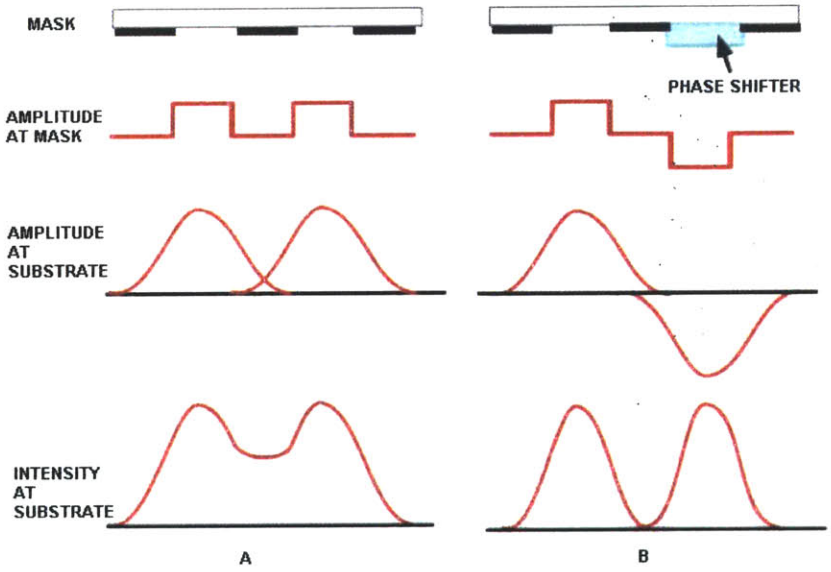


Figure 2.2: The concept of Phase Shifting Optical Lithography to increase resolution. (A) Without using phase shifting media in the optical path. (B) Inserting a phase shifting media to alter the phase difference at the substrate and hence the intensity distribution.

Soft lithography uses relief and recess features, patterned on a transparent and elastomeric PDMS photo mask. NFPSOL achieves selective blocking of light in the near field through relief features in the mask so that the each step on the relief surface of the mask translates to constructive or deconstructive interference of light on the photo resist surface, depending on the relief height used. A relief height of odd multiples of ($\lambda/2$) gives destructive interference at the relief edge which that of multiples of λ give constructive interference.

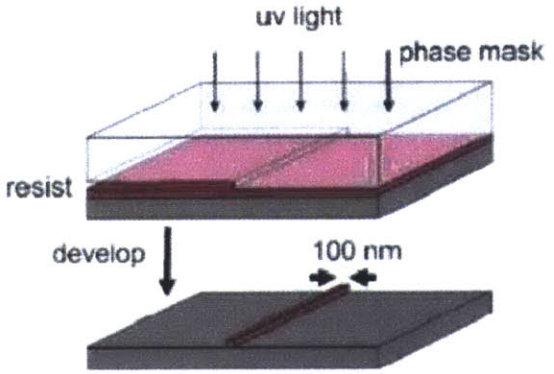


Figure 2.2.2: NFPSOL- Formation of features [24].

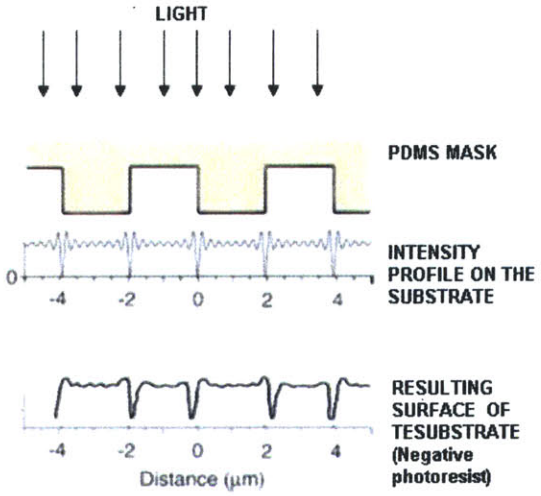


Figure 2.2.2: The optical phenomenon behind Near Field, Phase Shifting, Soft Optical Lithography [15].

The elasticity of the mask material, PDMS, and its near perfect conformance with a mating surface, give soft optical lithography some significant advantages over conventional optical lithography despite some limitations. We shall discuss them in section 2.4.

2.3 A Process Modeling Framework for NFPSOL

Causal modeling of any process provides a framework for identifying the critical parameters that affect output quality from a manufacturing process. The unavailability of relevant data from experiments or academia will restrict this section to identification of the input and output parameters of interest and also the sources of disturbance cause variance in the process output.

2.3.1 The input and output parameters of interest

The three most important input parameters which would determine the output and its quality are listed as below:

1. The incident light
    - The intensity of the incident light
    - The exposure time
    - The wavelength content
2. The PDMS mask
    - Density of relief features on the mask
    - Height of the relief features or the phase shifter depth.
    - Transmission coefficient of PDMS.
3. The photo resist and development parameters.
    - Behavior to light exposure (i.e, threshold intensity etc)
    - Thickness of the photo resist layer.
    - Post exposure bake (PEB) temperature and time.

The process outputs of interest are:

1. Line Width and section profile of the lines on the photo resist which can be expressed as an "unknown" function of incident light intensity, density of features on the mask and the wavelength content of the incident light as the prime factors.

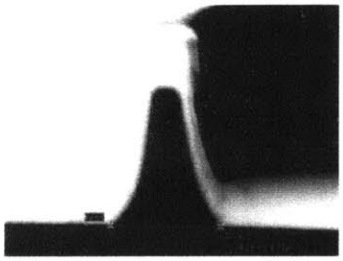
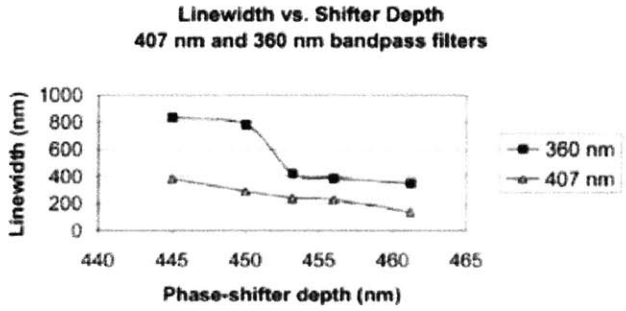


Figure 2.3.1: A typical feature developed with NFPSOL. Line width is defined as the width of the feature at half the height from the bottom [13].

2. Resolution or the spacing between the lines on the photo resist which is related to the density of features on the mask and the incident light intensity.



Linewidth vs phase-shifter depth using 360 nm and 407 nm bandpass filters (Exposure time = 20 s)

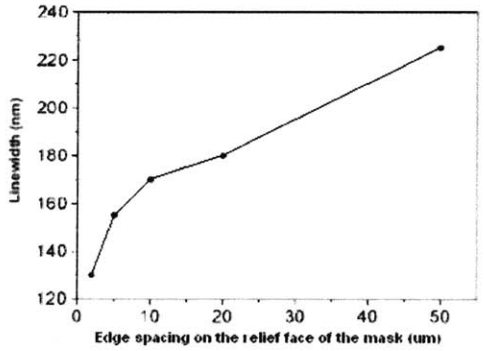


Figure 2.3.1: Variation of the output (Line width) with phase shifter depth, wavelength content and edge spacing on the relief face of the mask [16].

2.3.2 Sources of disturbance and variation

There are some factors which could affect output quality in the process even if perfect incident light and photo resist's exposure behavior is ensured. They are listed below.

1. The distortions in the PDMS photo mask while peel off from the master pattern, while handling during the process and on cyclic exposure to light on absorption of radiation.
2. The contact between the PDMS mask and the photo resist surface could be non-conformal due to possible trapping of air or particulate matter.
3. The post exposure treatment time of the photo resist has been known to have large variation [17]

2.3.3 Attempts on Process Modeling

The observed results have deviated slightly from those predicted by optics theory. Figure 2.3.3 shows that the resulting features on the photo resist layer are not as expected by the scalar diffraction theory [18].

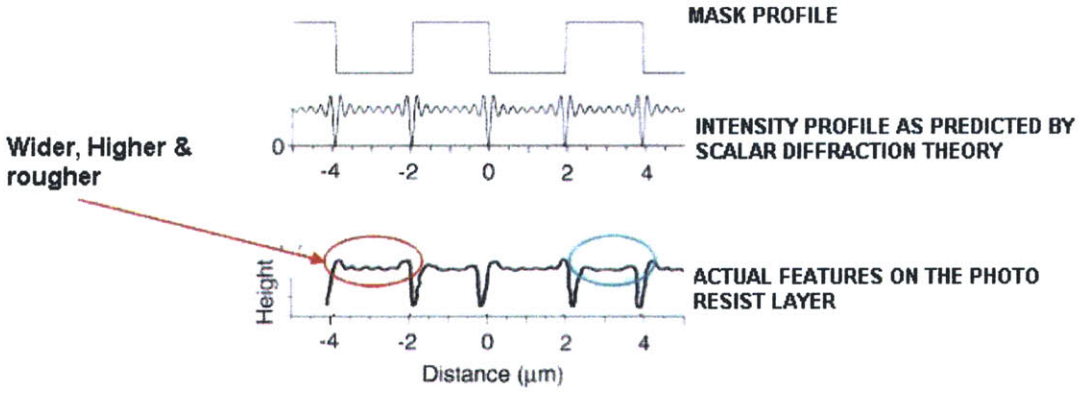


Figure 2.3.3: Scalar diffraction theory. The features formed corresponding to the relief features of the mask are wider, rougher and have greater average height than those corresponding to the recess features on the mask [11].

A better explanation of the process can be found with Full Vector Finite Element Methods [19]. Their method explains the anomalies in the scalar diffraction theory.

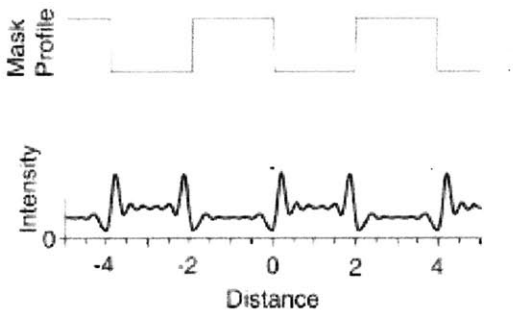


Figure 2.3.4: Full Vector Finite Element Method [11] provides a better explanation than the scalar diffraction theory.

2.4 The Steps Involved in the Process

Manufacturing parts using NFPSOL begins with the fabrication of a "master pattern" on a silicon wafer using any of the conventional lithography techniques. The master pattern is used a mold and liquid PDMS is poured into it. The solid PDMS is peeled off to give a PDMS mask with the desired relief and recess features. This mask is placed in conformal contact with a layer of baked photo resist on a substrate and exposed to UV light. Post exposure development is followed by etching to get the desired pattern onto the substrate. Further refinements like oxidation to reduce feature size or metallization are product specific.

1. Acquisition of a defect free PDMS mask.
2. Setting the process parameters.
3. Spin coating photo resist (about 3000 RPM for about 60 Seconds).
4. Soft bake ~100-135 0C (for about 60 to 90 Seconds on a preheated plate).
5. Conformably place the PDMS mask onto the photo resist layer.
6. Expose to light (between 1-15 Seconds).
7. Post exposure baking (for about 30 Seconds).
8. Develop the exposed resist (30 to 60 Seconds).

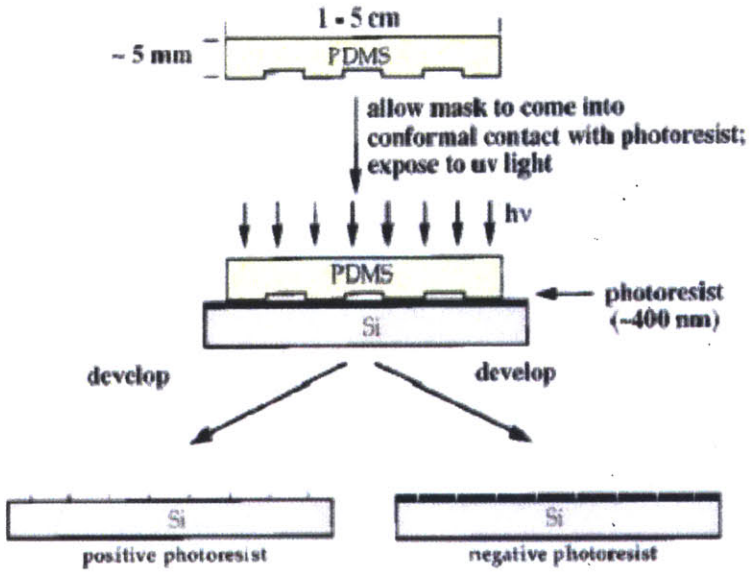


Figure 2.4: The Process Steps involved in NFPSOL.

2.5 Comparison with Current Lithography Practices.

NFPSOL has several advantages in terms of capability over the current lithography practices due to the properties of the mask material - PDMS. The elasticity of PDMS and its ability to make a near perfect conformal contact with any arbitrary shape allows its use for patterning on non-planar surfaces- which only the expensive and slow "Scanning near field optical lithography", or SNOL, can accomplish [20]. The other great advantage is the cost effectiveness of the process which does not use the expensive optical systems used in current commercial practices. Also, it is likely that the automation required might be cheaper than that for the convention means of optical lithography owing to the simplicity of the process. We shall first briefly discuss the key advantages and then move to the issues and problems which prevent this technique from being applied commercially as of today.

2.5.1 Advantage

The advantages of Near Field, Phase Shifting, Soft Optical Lithography over conventional lithography techniques are listed below.

1. Capable of patterning features on planar as well as non-planar surfaces.
2. Capable of patterning large area (up to several sq meters). The area is limited only by the area of the PDMS mask used.
3. Low capital and running costs.
4. Parallel energy transfer and thus faster processing as compared to the step and scan optical lithography systems.
5. Feature size and resolution as low as 30nm has been demonstrated in laboratories [21]

2.5.2 Currently Unsolved Issues and Challenges

Despite its capabilities and promise of low cost for high volume production, Near Field, Phase Shifting, Soft Optical Lithography has not been used commercially for production as of today. Some of the assignable causes for the same could be:

1. No single process model can accurately predict the output given a set of inputs. That is, the knowledge about the process physics is probably still under-developed.
2. Deformation of the PDMS masks while peeling off from the master, while handling during the process (due to gravity as well as adhesion forced with the photo resist layer [22]) and on cyclic exposure to light- is detrimental to output quality.
3. Difficult to produce multi-layered patterns as the elastic nature of the mask makes registration very difficult. This limits its use for only certain single layer applications. [23]

4. Obtaining multiple feature sizes and resolutions for the same product is not addressed.

2.6 Conclusions

In this chapter, we described a soft lithography technique which achieves pattern transfer by modulation of light in the near field using elastomeric masks with relief features on its surface. The process is still under development as academia works towards a better explanation of the phenomenon. The process promises great potential in the future and the major challenge involves achieving registration for multi layered applications. In the subsequent chapters, we will discuss micro contact printing in detail and our efforts towards taking this technique from the laboratory to industry.

Chapter 3

Micro contact Printing

3.1 Introduction

Micro contact Printing (or *μCP)*, as it will be frequently referred to in this text) uses the relief patterns on a PDMS stamp to form patterns of self-assembled monolayers (SAMs) of inks on the surface of a substrate through conformal contact. Micro contact printing differs from other printing methods, like inkjet printing or 3-D printing, in the use of self-assembly (especially, the use of SAMs) to form micro patterns and microstructures of various materials [25].

This chapter provides a literature review of the process including its principles and characteristic features. We also discuss some industrial design efforts to take this technology from the laboratory to commercial production.

3.2 Principles and characteristics of Micro contact Printing

Micro contact printing is a method for patterning Self-Assembled Monolayers (SAMs) on surfaces using elastomeric stamp [26]. The most distinct characteristic of *μCP* is its use of self-assembly (especially, the use of SAMs) to form micro patterns on a substrate [27]. SAMs are formed by contact between a topographically patterned elastomeric stamp, wetted with an ink consisting of molecules that are capable of forming SAMs, and the surface of substrate.

Micron scale SAMs by Micro contact printing can be formed manually in a conventional chemical laboratory and it does not require any photolithographic equipment or a clean room environment. Therefore, the simplicity and economic efficiency for patterning micron scale layer are major benefits of Micro contact printing.

Another Characteristic of MCP is conformability. Because the Micro contact printing uses PDMS stamps, the stamp is able to conform to substrate with little external normal force and at the same time can compensate for the surface roughness of the substrate. This is important in transporting molecular level SAMs.

3.2.1 Formation of Self Assembled Monolayers or SAMs

Self-Assembled Monolayers (SAMs) are layers formed on a solid surface by spontaneous organization of molecules. It has been reported that a polymer inked with an alkanethiol and brought into contact with a gold-coated surface can form a monolayer of these molecules in the areas of contact [29].

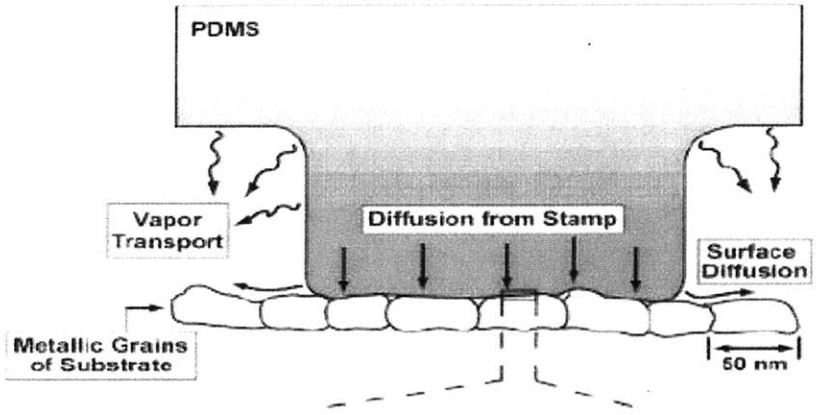


Figure 3.2.1: Schematic illustration depicting the application of a PDMS stamp containing thiols to a polycrystalline metal film. The mechanisms of mass transport from the stamp to the substrate are also shown [30].

The ink used in Micro contact printing to form the SAMs is mainly transported through diffusion at the stamp-substrate interface. Diffusion from the edges of the stamp and vapor transport are the non-contact mechanisms that can also form SAMs (see figure 3.2.1). When the target feature sizes are smaller than 500 nm, these non-contact transport mechanisms become significant enough to degrade the output quality [31].

The substrates, on which the SAMs are formed, are generally prepared by common physical vapor deposition (PVD) methods, i.e., thermal or electron beam evaporation. Among a wide range of materials used for substrates, gold is the most commonly used as it is easily available as a thin film and easy to pattern by a combination of lithographic tools (photolithography, micromachining) and chemical etchants. In addition, gold is a very inert material, so it does not oxidize at temperatures below its melting point. However, silver [13] and copper [14] have been used as a substrate for forming SAMs through Micro contact printing.

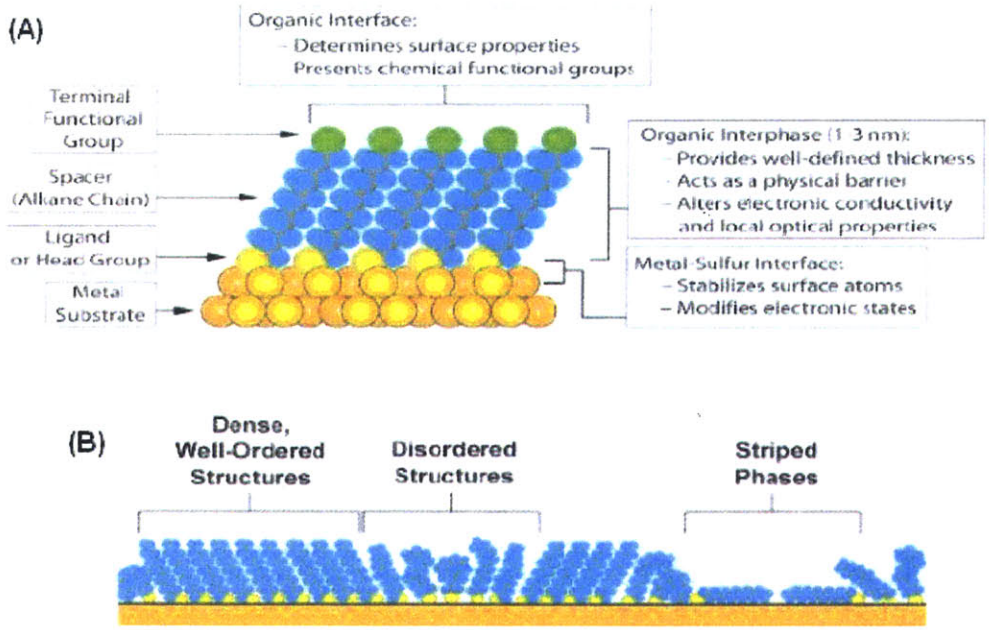


Figure 3.2.1. Self assembled monolayers. (A) Schematic diagram of an ideal, single-crystalline SAM of alkanethiolates supported on a gold surfaces (B) The variety of structural arrangements found in SAMs prepared by μCP when the stamp is wetted with a 1-10 mM solution and applied to the substrate for 1-10 seconds [15].

3.3 Micro contact printing Procedure.

Micro contact Printing can be thought of as a two stage process: Inking and Stamping. Inking consists of wetting with the ink and drying. Ink that forms SAMs is transferred through a solvent such as ethanol, so solvent needs to be removed after transferring ink to stamp. The wetting step is applying ink solution to a PDMS stamp and drying step is removing solvent from PDMS. Solvent can be dried in the air, but a steam of nitrogen gas helps to reduce time for drying. Inking and drying times depends on factors such as ink concentration, printing area, and printing method. Typically, inking step takes 30 ~ 60 seconds and drying step require 10 ~ 60 seconds.

The stamping process comprises 3 sub-steps: Initial contact & propagation, full contact, and separation. Achieving full contact through gradual contact is important in order to minimize air trapping between stamp and substrate. After achieving full contact, contact time is required to transport inks from a stamp to a substrate and form SAMs. The required length of contact time can be different mainly according to ink type, concentration of ink or target thickness of SAMs.

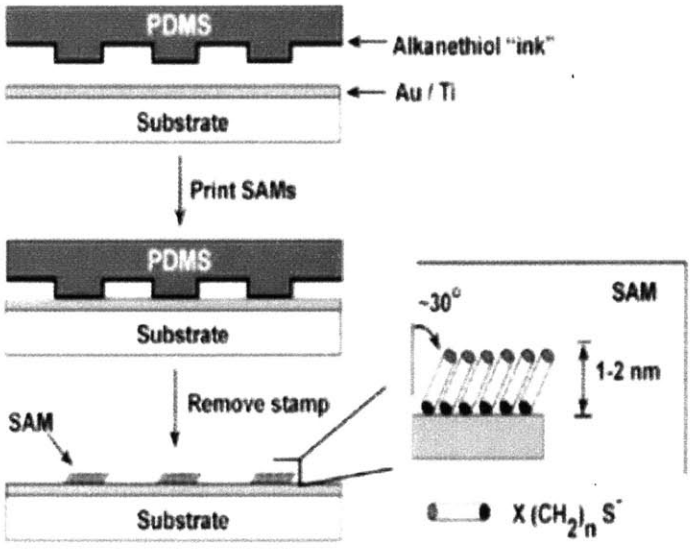


Figure 3.3: Micro contact printing procedure [16].

Table 3.3 summarizes the process, the required time and typical failure modes of Micro contact printing reported through the research so far.

Process	**Inking**		**Stamping**		
Step	Immersion	Drying	Initial contact and propagation	Self assembly full contact	Peeling off
Time	30 ~ 60 sec	10 ~ 60 sec	1 ~ 5 sec	0.3 ~ 60 sec	1~5 sec
Failure Modes	Swelling	Distortion (by capillary force)	Air trapping	- Diffusion of ink - Deformation of stamp	
Control factors	- Inking time - Concentration	- Drying Time - N_2 flow	-Propagation method -Propagation velocity	- Contact time - Pressure - Temperature	- Velocity

Table 3.3: Micro contact Printing Process and characteristics [16].

3.4 Critical Factors for *μCP*

The critical factors for high quality printing for a given pattern are explained next.

1. The Elasticity of the PDMS Stamp

The use of flexible stamps allows micro contact printing to achieve conformal contact for ink transfer. PDMS stamps with low young's Modulus (1~3 MPa) are widely used because they provide near perfect conformal contact. Young's modulus is determined by the mixing ratios between the prepolymer precursor and curing agent, and the preparation conditions, such as curing time and temperature.

In the past, the critical dimension of soft lithography was limited by the choice of commercial siloxane with a young's modulus of 3 MPa as stamp material. This material proved to be too soft to define features smaller than 500 nm. Harder stamp materials had to be developed to allow printing with critical dimensions below 100 nm. An IBM research group demonstrated 80 nm feature size Micro contact printing using a harder

stamp with a young's modulus of 9.7MPa [17]. Therefore, young's modulus should be carefully determined in accordance with the feature size of the product

2. Contact Time of the Stamp with the Substrate

The thickness of the printed SAMs is proportional to contact time (Figure 3.1). Also, the longer the contact time, the lower is the defect rate but the greater the pattern width (Figure 3.2). Generally, it has been shown that contact time is less than 30 seconds, but longer contact time (>30s) usually results in an undesired pattern transfer due to vapor transport of thiols from the stamp to the surface in non-contact regions.

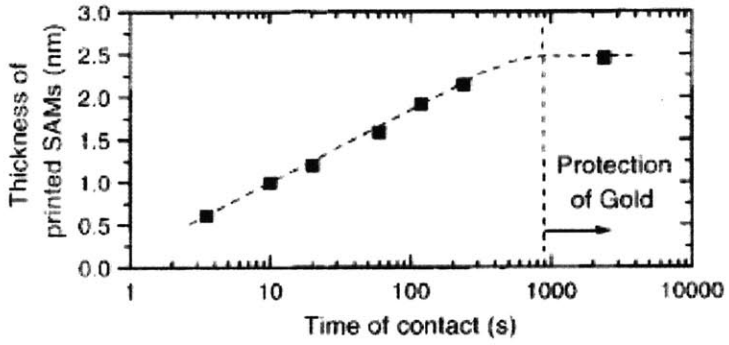


Figure3.4.1: Relationship between thicknesses printed with 0:2 milli Mole solution of ECT ink [18].

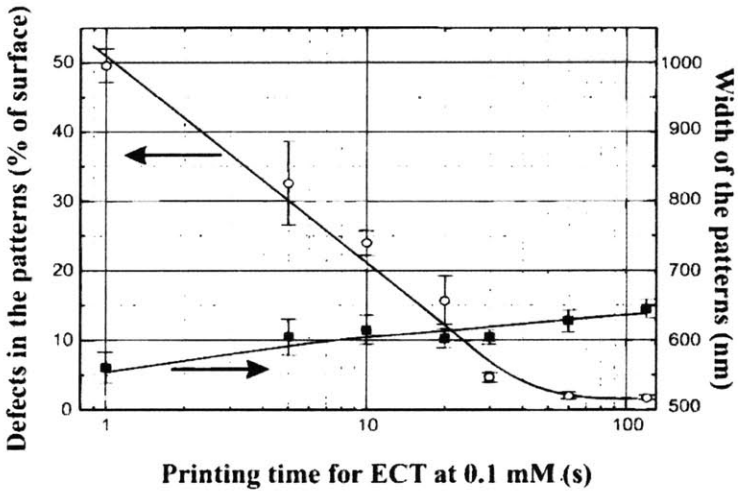


Figure 3.4.2: Relationship between printing time and the number of defects in the pattern [19].

3. Concentration of the Ink Solution

The relative proportion of thiol and ethanol in the solution affects not only the width of the pattern transferred but also the defect rate as shown in figure below.

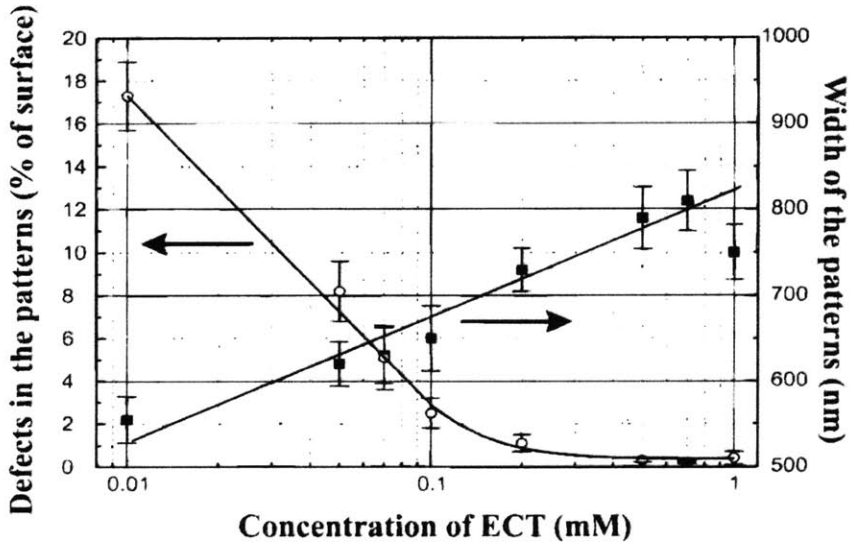


Figure 3.4.3: Quality of Micro contact printed gold structures performed in the case of an ECT contact-inked stamp. Note that defects in patterns are decreased when concentration of ECT gets higher [19].

In addition, concentration and printing time are also inversely related: High concentrations of thiols takes less printing time.

4. Contact Pressure

The required pressure is based on the pressure that can initiate and control conformal contact. Excessive pressure causes the relief features of the stamp to collapse. In order to prevent unexpected printing caused by collapse of stamp from excessive force, we need to consider the basic deformation mechanism. Figure 3.4.4 shows a basic geometry of a stamp that consists of periodic relief line features with height of h, feature width of $2w$, and trench width of 2a. When pressure is applied to the stamp, several deformations

occur. These deformations include in-plane lateral expansion, "sagging" of the trench and relief feature from compression by the external stress [21].

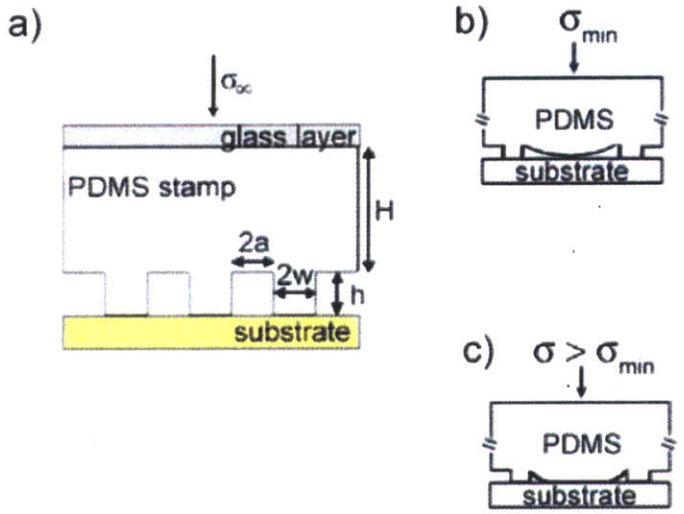


Figure 3.4.4: Illustrations of (a) basic geometry of a PDMS stamp and stamps deformed into contact with substrates under (b) required minimum and (c) excess pressures. [30] showed that the height-to-width ratios, the aspect ratios, of the relief structures on PDMS stamps need to be between about 0.2 and 2 in order to obtain defect-free printing. If the aspect ratio of the PDMS feature is too high, the roof of the feature may come into contact with a substrate under its own weight or under an external pressure. When the aspect ratios are too low, the relief structures are not able to withstand the stamp weight [22].

Specifically, the model for promoted contact between roof and substrates under external stress is suggested by Hui [13] as follows.

$$V_{\max} = \frac{4\sigma_{\min}}{\pi E^*}(w+a)\cosh^{-1}\left[\sec\left(\frac{w\pi}{2(w+a)}\right)\right]$$

where V_{max} is the maximum displacement of the roof by an applied minimum external stress, and the conformability (i.e. the ratio of Young's modulus divided by the work of adhesion (or E/w) of the material) was found to be a measure of the spontaneous occurrence of conformal contact as well as of spreading collapse [28].

Another driving force that causes stamp to collapse or driving the conformal contact is the adhesion force. According to the analysis performed by Bietsch and Michel [28] it takes 400 kPa to conform a Sylgard 184 stamp to a surface with 50 nm rms roughness.

However, conformal contact can be achieved under a small force such as the self weight of the stamp. Moreover, K. J. Hsia et al. [23] observed collapse of groove regardless of self-weight or normal force. According to their study, interfacial adhesion force is the main driving force of groove collapse, and they developed a parameter, which is a function of feature width, height, surface energy, and Young's modulus, that determines the adhesion force.

5. The Choice of Inking Method

Inking can be carried out in three different ways – Immersion inking, Pen-Type inking, and Contact inking.

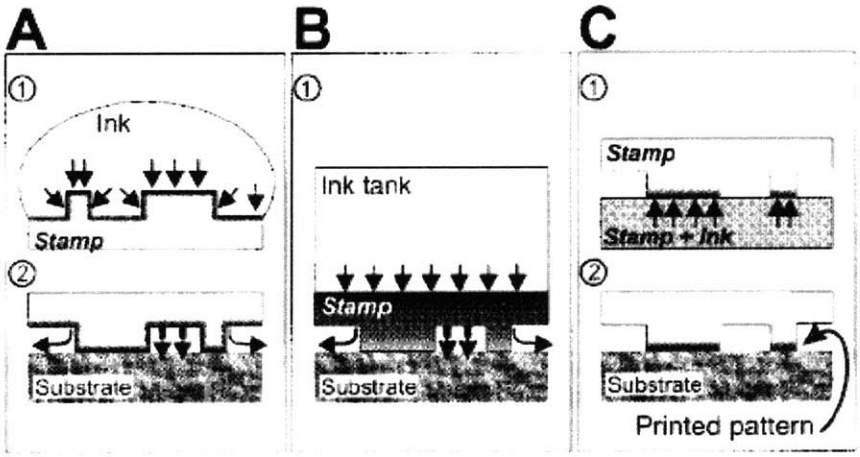


Figure 3.4.5.1: Three different types of inking for Micro contact printing. The patterned stamp is consequently inked only via the contact zones where molecules will be needed in step 2. Concentration and immersion time are inversely related: low concentrations of thiols in solution require long immersion times. (A) A liquid inking technique allows impregnation of the entire surface of the patterned stamp with a drop a dilute solution of thiols in ethanol. The possible consequence of this inking method is the interference during printing of thiols on the stamp adjacent to the regions of contact. (B) Inking a stamp by transferring thiols from a liquid reserve of alkanethiols in ethanol through the PDMS allows the stamp to be inked permanently and reused readily, but does not localize this stamp impregnation only where needed (C) Direct inking of the patterned stamp follows its contact with a flat inker pad previously impregnated by immersion in a dilute solution of thiols. [7].

6. Propagation methods.

Air bubbles trapped between the stamp and substrate easily occur when large areas are printed. Several methods, such as Micro contact printing under the low pressure environment [23] and forcing air out using flexible backings have been studied, but contact initiation and propagation is simple and generally used method.

The contact propagation can be categorized into three methods: radial contact propagation, linear contact propagation, and rolling contact propagation.

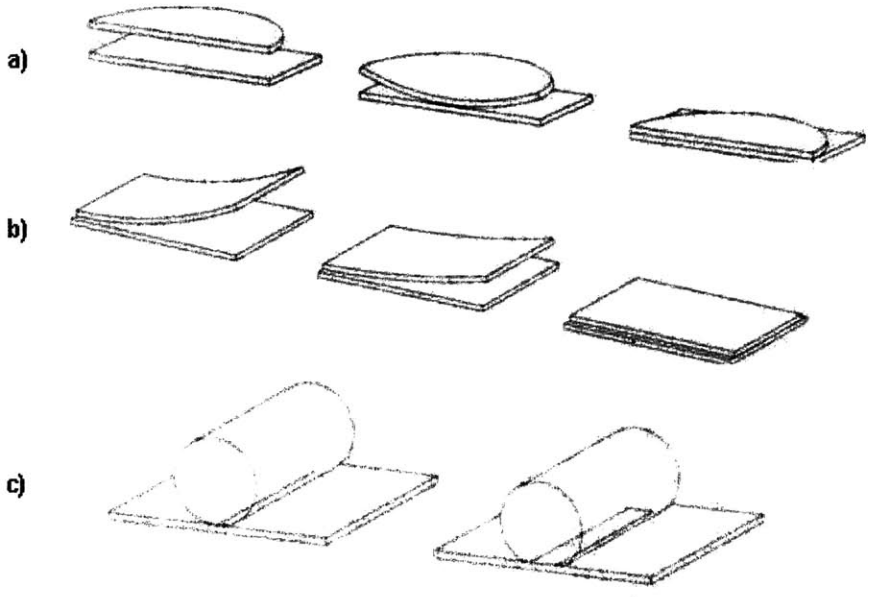


Figure 3.4.5.2: Three contact propagation methods (a) Radial contact propagation (b) Linear contact propagation (c) Rolling contact propagation [23].

Radial contact propagation is done by making a stamp convex. The contact initiation starts from the center, and convexity is decreased gradually along with spreading contact region. [16]. Linear contact propagation starts from line contact instead of point contact. The end of a bent stamp creates line contact initiation and contact area gradually increases by dropping the stamp gently. A thin bendable layer of metal or polymer can be used for backing the soft stamp. Third contact propagation method is rolling contact,

using cylindrical stamp. In terms of automation, rolling propagation has benefits in that it does not require another mechanism or process for separation of stamps after printing. Moreover, rolling propagation can simply scalable to mass production process of Micro contact printing such as reel to reel process. However, the deformation of stamp is difficult to expect during preparing cylindrical stamp and printing.

7. Temperature

It has been known that forming SAMs at temperatures above 25℃ can improve the kinetics of formation and reduce the number of defects in them [9].Also, the effect of temperature is particularly relevant during the first few minutes of the formation of a SAM when most of the adsorption and reorganization of the SAM is taking place.

3.5 Summary of manufacturing considerations.

Rate, quality, cost, and flexibility of manufacturing process provide systematic and analytical view not only in evaluation but also in designing the process. Therefore, critical factors related to physics and automation of the process should be considered based on the four factors.

Rate:

In designing an automated Micro contact printing process, it is important to decouple the printing and inking processes such that two processes operate independently or individual processing times.

As discussed in chapter 3.2, different inking methods require different inking time, and printing time mainly depends on two factors, concentration of ink solution and target thickness of SAMs. Because micro contact printing is a serial process involving inking and stamping, the time to complete inking and printing needs to be simulated by deciding key factors such as inking method and concentration of ink before designing a production line.

Quality:

There are a lot of factors that affect printing quality, but the fundamental problems of Micro contact printing relate to the properties of the stamp material. Xia et al [22] proposed three main concerns when Micro contact printing is implemented as one of the micro fabrication process.

1. The shrinkage of PDMS during curing and the swelling of PDMS by a number of nonpolar organic solvents such as toluene and hexane.
2. The elasticity and thermal expansion of PDMS makes it difficult to get high accuracy in registration across a large area.
3. The softness of an elastomer limits the aspect ratio *(height of feature / length of feature)* of microstructures in PDMS. When the aspect ratio is too high, two posts can easily stick together (pairing). If the aspect ration is too low, space between two posts will collapse (sagging)

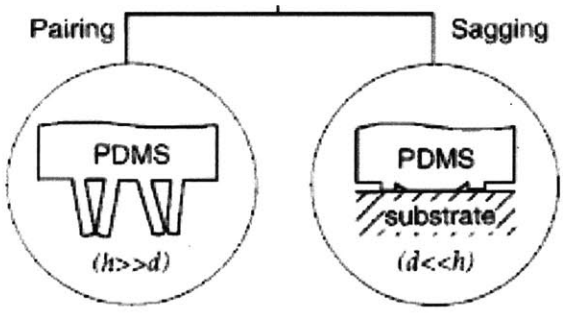


Figure 3.4.5.3: Schematic illustration of possible deformations of microstructures in the surfaces of elastomers [26].

Cost:

The expected cost of tool, ink solution, change over time should be considered in designing & evaluating Micro contact equipment. Usually, tool cost depends on wear of the tool (PDMS stamp) but wear has not been reported so far, so it is very difficult to predict the total tooling cost. However, we can minimize the tool cost by minimizing the number of tools in the buffer between inking and printing station.

Flexibility:

In a high volume manufacturing process, flexibility of tooling and tool change - over time are important, so Micro contact printing machine should be designed such that it provides a fast tool changing mechanism. In addition, if Micro contact printing machine is used for multilayer micro fabrication, the tool changing mechanism should also provide accurate registration capability.

3.6 Industrial design efforts for Micro contact printing

- ***Rolling***

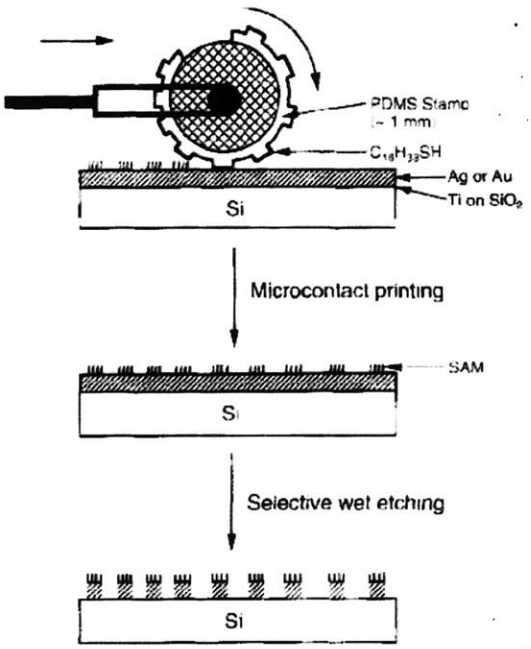


Figure 3.6.1: Schematic procedure for conducting μCP with a rolling PDMS stamp [12].

Micro contact printing by rolling is one way to form patterned SAMs at a high speed and, therefore, appropriate for mass production. Younan Xia et al [57] demonstrated that a minimum feature size of ~300nm is achievable using a cylindrical stamp of which diameter is approximately ~ 4 cm. 3 inch gold coated wafer (an area of 50 cm2) was used for a substrate and printed at 2cm/sec. (It take 15 seconds to print 3 inch wafer) A single inking can print 4~5 times of printing without showing any difference in terms of defects

in SAMs. The main disadvantage of rolling is that feature on stamp should be deformed to make a cylindrical stamp, which limits the minimum resolution or feature width of pattern.

■ ***Wave printing***

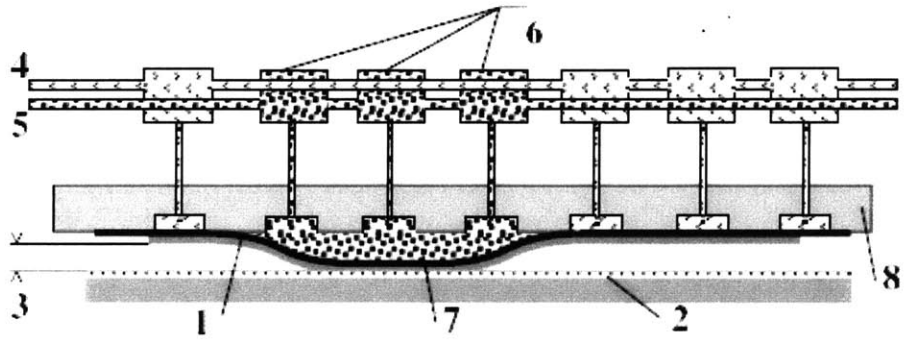


Figure 3.6.2: A vertical Cross-section of the wave printing prototype [58].
1: Stamp-backplane assembly; 2: Substrate; 3: Working gap (≈ 100μm); 4:Vaccum supply; 5:Pressure supply (≈2kPa); 6:Valves switched to pressure supply, thereby creating the wave. 7, 8: Grooves-plate [15].

The concept of wave printing [58] was proposed in order to minimize the deformation and distortion of patterned SAMs printed by the stamp which has both large and small feature and space in a single stamp. Slender posts (height to width ration h/w 2.5) can be easily buckled and sagging will occur if the height of posts is relatively small to space between relief features by a small normal force. In wave printing, a glass backplane as a flexible glass mount is attached to elastomeric stamp to prevent distortion of relief features and this stamp-backplane creates wave by the pressured air mechanism, generating line contact between stamp and backplane. It took twenty minutes for inking, one hour for drying, and 15 seconds for printing as small as 0.75 μm, with h_{post} =2.3 μm. Micro contact wave printing demonstrated single layer capabilities with very low distortions.

3.7 Conclusions

In this chapter, we discussed the process physics and principles behind micro contact printing. The critical factors that determine manufacturability using the process were also discussed besides a few industrial design efforts to automate the process. In the next chapter, we will discuss a manufacturing system that we have designed for commercializing production using micro contact printing. We will also present the design of a prototype to automate the inking and stamping steps.

Chapter 4

Concept Design and Realization

4.1 Introduction

The objectives of this project included designing a manufacturing system for micro contact printing and prototyping an automated system for inking and stamping that could achieve economically viable process. To achieve these goals, a thorough understanding of the process physics involved in micro contact printing was required besides identifying the critical factors that could contribute to output quality and production rate. The final manufacturing system design and the prototype design were derived after several design iterations. The key breakthrough in the design process was achieved on the realization that the target substrate was a flexible sheet of gold coated plastic film which allowed keeping the stamp stationary (to avoid stamp deformation) and apply the substrate onto the stamp while achieving linear propagation (to avoid air bubbles).

This chapter takes the reader through some of the most important design iterations towards the final design. We discuss designs for both the inking and stamping steps. Though some of the designs do not involve the use of a flexible substrate, not only are they are significant milestones in this project but also and can be considered for hard substrates. Finally, we propose a complete manufacturing system for high volume production.

4.2 The Inking Process

Three inking methods are explored and discussed in the academia [5].

1. **Wet Inking** - Ink is uniformly applied to the PDMS stamp to cover the entire exposed surface area. This is done by fully submerging the patterned surface of the stamp in an ink tub.
2. **Pen Stamp** - Ink is stored in an ink tank behind the PDMS stamp and diffuses onto the surface of the stamp for stamping. The method derives its name due of its functional similarity to a writing ink pen.
3. **Contact Inking** - Ink is only applied only to the relief features of the stamp. This is achieved by bringing the stamp's relief feature in contact with a pad soaked with ink.

Table 4.2 gives a functional comparison for each method.

	Wet Inking	**Pen Stamp**	**Contact Inking**
Distortion of stamps	Capillary effects, slight swelling	Strong swelling	No distortion
Adversary diffusion of the ink (HDT)	Strong	NA	Strongly minimized
Geometric effects	Some	NA	Strongly minimized
Surface crystallization	Strong (>=0.5mM)	Strong (>=0.01mM)	Strong (>=0.5mM)
Ease for repeated inking/printing	Bad	Very good	Fair
Scalability	Unknown	NA	Plausible
Economy of reagent	Bad	Very Good	Fair

Table 4.2 Comparison of Three Methods of Inking Stamps for μCP.

Stamp distortion is important to us since a single stamp may have to go through multiple rounds of printing in the industrial production. Even slight distortion would be dramatically reduces the productivity and product quality collectively.

The scalability is also important to us as the goal is to design a machine good for large scale production. For output quality of the printed patterns, it is important that adversary

diffusion (due to vapor transport) is minimized. With the factors above and Table 4.2, contact inking was chosen be the most appropriate inking method for this project.

4.2.1 Designs for Inking

Having decided upon contact inking for the reasons outlined in section 4.2, several mechanisms to achieve it were considered. The most important ones are described next.

1. Roll Inking

A hard roller is inked by rolling it over an ink-soaked pad (contact inking) or by having an ink tank inside (Pen Inking). Then the roller rolls over a fresh stamp to transfer the ink. The advantage of this design is that it could generate linear propagation (to the ink) during the inking process to apply the ink uniformly over the stamp. Also the inking time between the roller and the stamp could be well controlled by the forward motion of the roller. Thirdly, it's convenient to shift between the stamping inking stage and the inking roller inking stage.

However, there are also several challenges with this design.

1. The vertical position of the inking roller determines the applied pressure. Excessively high pressure may deform the stamp while insufficient pressure may lead to insufficient ink transfer or worse, slippage causing some relief features to miss ink application completely.
2. The weight balance of the roller. As the weight of the roller itself is big enough to cause the deformation of the stamp, the weight must be well balanced with the external mechanism.
3. If applying the wet ink to the stamp, it takes longer to dry the ink. After the stamp is inked, it has to wait for another 10-60 seconds before it can be shifted to the printing. This could be the bottleneck for the process.

2. Polygonal Inking

Instead of a cylindrical inking roller, a regular polygonal rolling tool could be used. We consider the case using a hexagonal tool. The design is easily understood with the help of figure 4.2.1.2. A major advantage is that the inking and stamping steps are readily integrated.

However, the challenge is that the inking pad and stamp have to be lifted up or down each time the hexagonal tool rotates. As figure 4.2.1.2 shows, when the hexagonal tool rotates, the ink pad and the stamping stage have to be moved vertically by ½ of the edge length to allow the pinnacles of the hexagon rolling through. This requires high precision position control system, particularly given that the stamps will have to be changed regularly.

The other disadvantage, which is more critical than merely the mechanical inconvenience, is air trapping. As the hexagon surface rests over the stamps, there will be no linear contact propagation.

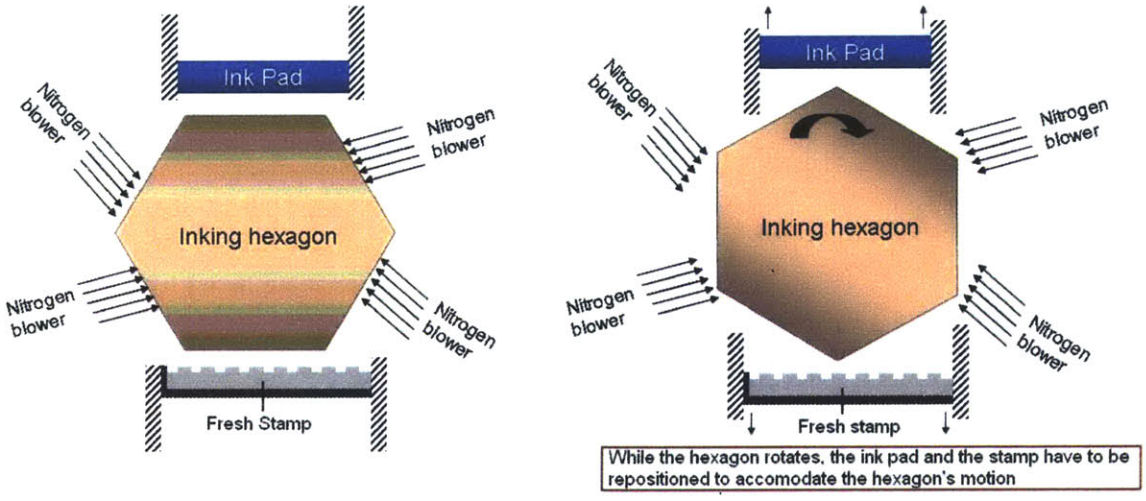


Figure 4.2.1.2: The basic concept of polygonal inking.

3. The Finalized Inking Design

The final inking design is shown step by step in figure 4.2.1.3. This design is based on the inking roller idea and takes into consideration the need for weight balance, pressure adjustment and vertical position control.

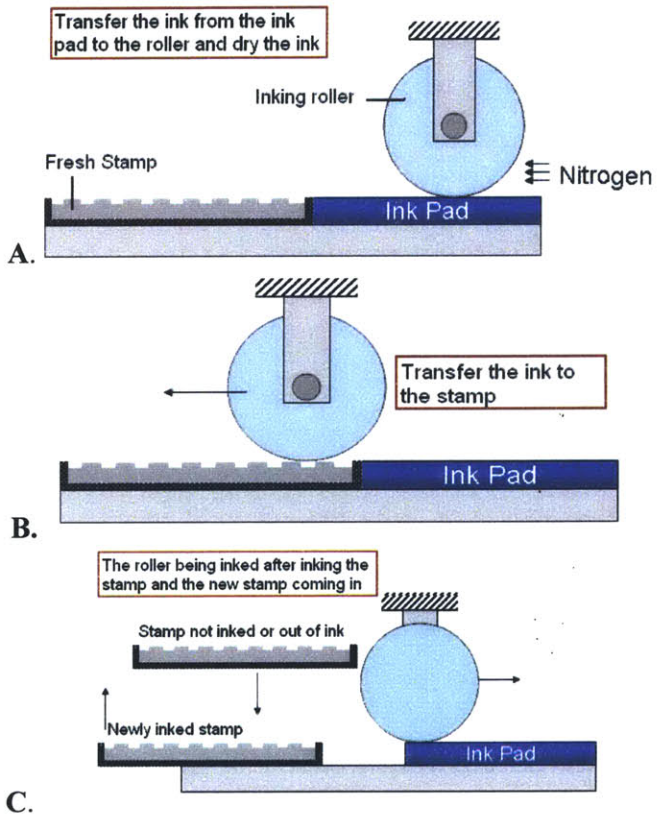


Figure 4.2.1.3: The finalized inking design. (A) The inking roller runs over an ink pad to collect ink while a stream of Nitrogen gas dries the ink solution. (B) The inked roller transfers the dried ink onto a fresh PDMS stamp (C) The inked stamp is transferred onto the stamping station and is replaced by a new stamp [33].

The inking roller is supported by two arms and an arm connector that is driven by a lead screw actuator. The roller is rigidly fixed to a steel shaft that is connected to the arms. The roller is supported on either end by timing belt pulleys which run on a urethane timing belt track. This ensures smooth and slip free motion. The ink pad is set up under the rails. The junction between the roller shaft and the arms is slotted to allow free vertical motion of the roller.

A nitrogen stream will be applied to the roller to dry the ink while the roller moves on the rails. The ink will then be almost dry before being transferred to the stamp. With this dry inking design, the actual drying and inking time to the stamp will be dramatically shortened and yet the quality will be assured.

A fresh stamp will be positioned on the other side of the inking stage. After ink has been collected from the ink pad and dried by the nitrogen stream, the roller will start transferring the ink onto the stamp. After the inking completed, the newly inked stamp will be transferred to the stamping station through two sequential transfer mechanisms- transfer of the stamp from the inking station to the stamping station and transfer of the stamp onto the stamping stage for stamping.

With this design, the two challenges to the previous roller design have been overcome. The weight of the roller will be supported by the rail together with the frame system. The vertical position can be adjusted along the slots on the two roller arms. More importantly, with the dry inking design, the inking and dry time for the PDMS stamp will be shortened, which will greatly improve the production rate.

At the same time, the advantages of the roller design remains. The linear propagation effect will reduce the air trapping during inking and automating the shift between the inking pad and stamps is straightforward.

4.3 The Stamping Process

In our design a flexible sheet of PEN is used as the substrate, which is applied onto a stationary PDMS stamp, and this played a dominant role in finalizing the design for the stamping process. However, several designs for stamping with hard substrates were also considered, which helped us in arriving at the finalized design. Regardless, the following are the key considerations for achieving good stamping outcome:

1. The ability to bring the stamp and the substrate in contact at an appropriate contact pressure.
2. Spreading of the substrate onto the stamp or vice versa in a way such that there is no air trapping.
3. The ability to peel off with force just enough to overcome the adhesion force. Excessive force might damage the printed features due to excess strain induced.

4.3.1 Designs for Stamping

In this section, we will discuss the most important stamping designs which led us to the final design. We shall first describe a couple of designs for hard substrates and then present the final design applicable to flexible substrates.

1. Roll Stamping

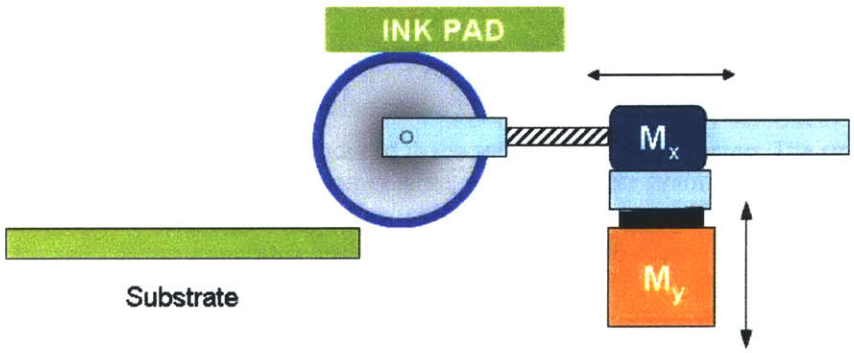


Figure 4.3.1.1.A: Roll Stamping. The motor Mx moves the roll forward and the motor My can control the vertical position of the roll and hence the contact pressure either actively or in open loop.

In this design a hard roller wrapped with a PDMS stamp with the desired patterns is run over an ink pad. The ink solution is dried with a stream of nitrogen and transferred onto a fresh PDMS stamp. While the process is extremely simple and involves only a horizontal motion, it has the following drawbacks:

1. Wrapping of the stamp over a roller can cause deformation of the features on the stamp (figure 4.3.1.2). It can be easily shown that the strain in the stamp will be a

function of the diameter of the roll. This will severely affect the resolution achievable with this process.

2. The vertical position of the roller will have to be very accurately controlled in real time to achieve the right contact pressure.
3. Any slippage due to mismatch between the angular and linear motions would be detrimental to output quality.
4. It would have been hard to control the contact time.

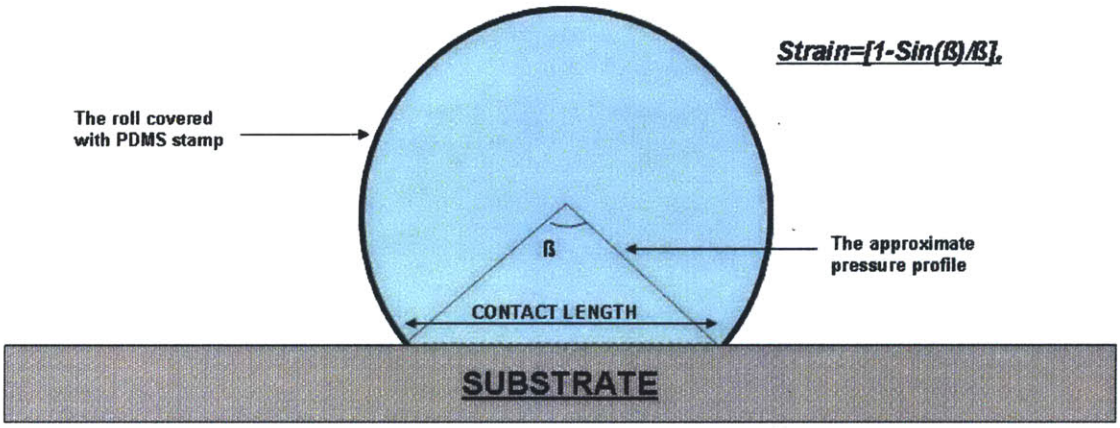


Figure 4.3.1.1.B: Elastic strain induced in the stamp with Roll Stamping. The strain is [1-Sin(ß)/ß], So, as contact length becomes large, we have greater strain which would severely distort the stamp given the high feature-size/resolution ratio. However, if the radius of the roll is large, the angle could still be small.

2. Polygonal Stamping

An improved variation to roll stamping, polygonal stamping overcomes the stamp deformation due to wrapping of the stamp on a roller. Each face of the polygonal tool is covered by a PDMS stamp (but not rigidly attached). The tool can rotate about a hinge (the instantaneous edge of the polygonal tool in contact with the substrate. See figure 4.3.1.4) and applies the stamp onto the substrate. Linear propagation is achieved without wrapping the stamp onto a cylindrical roller and thus stamp deformation is avoided. The design is explained step by step through figure 4.3.1.2.

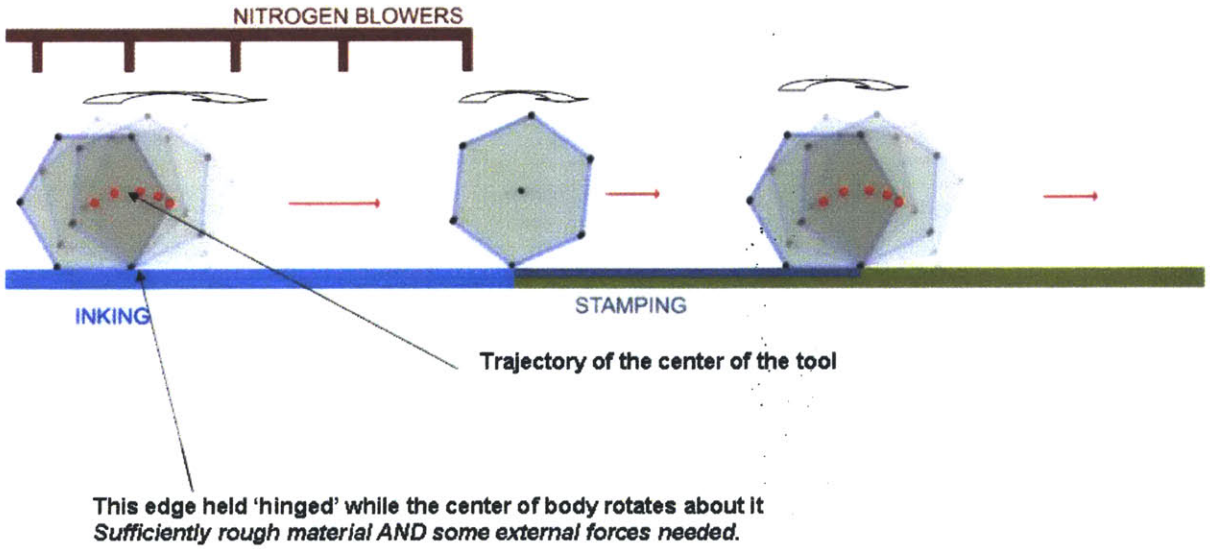


Figure 4.3.1.2.A: The basic concept of polygonal stamping.

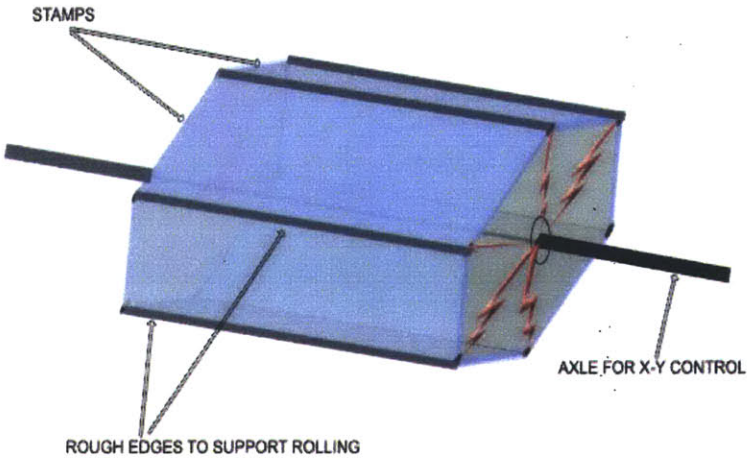


Figure 4.3.1.2.B: The stamping tool proposed for Polygonal Stamping.

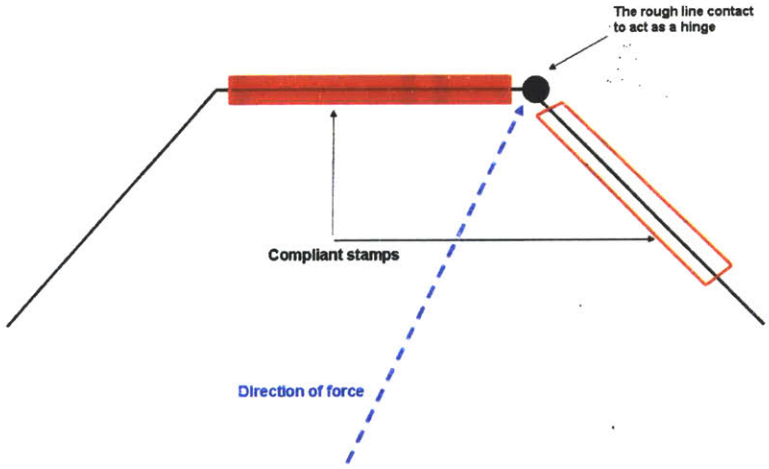
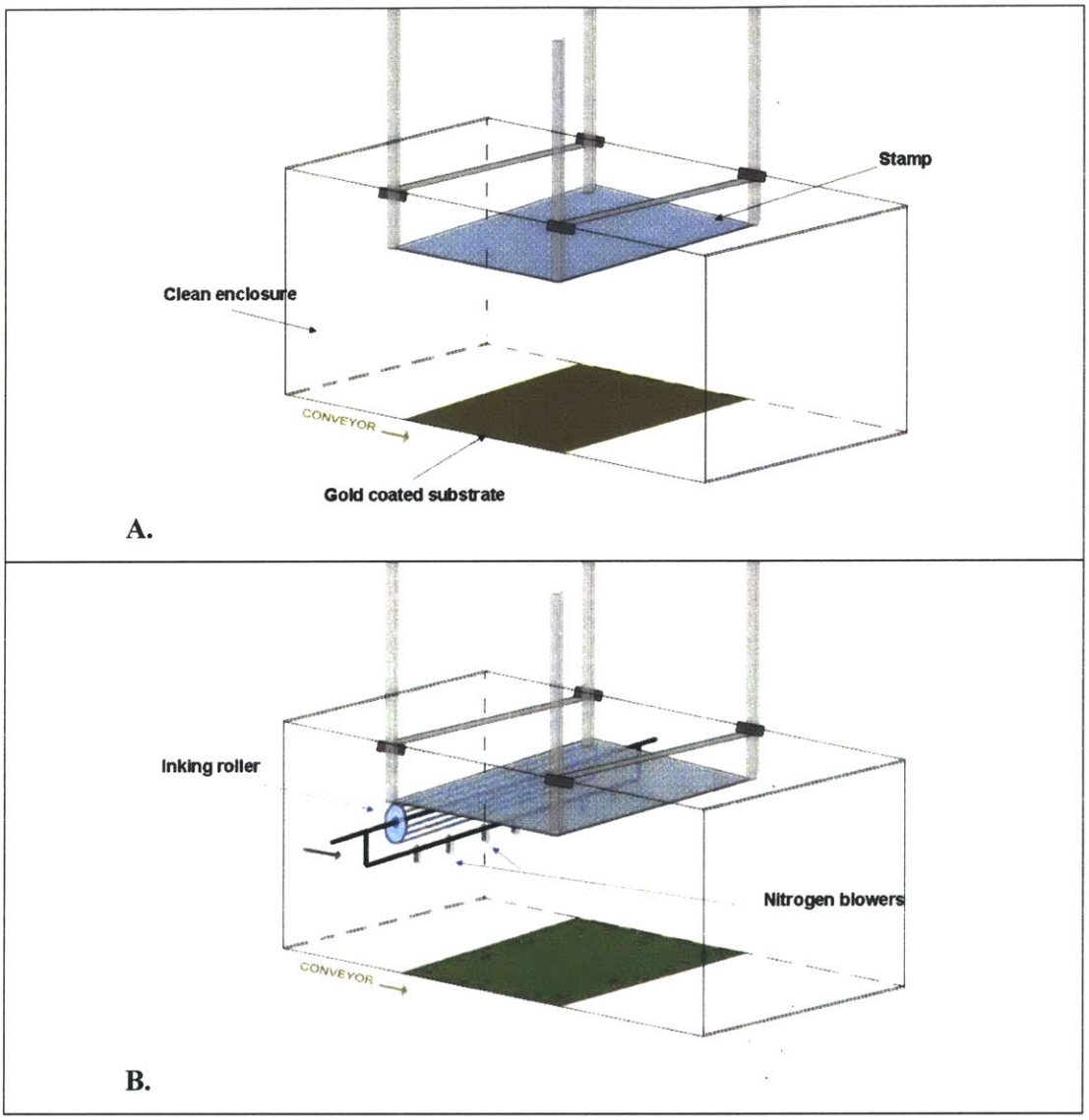


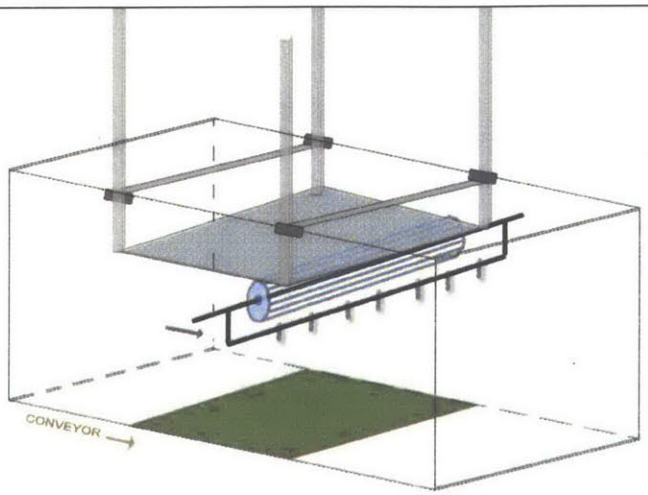
Figure 4.3.1.2.C: The hinge effect of the polygonal stamping tool.

In this design, it is important that the PDMS stamps are not attached to the faces of the tool as then no linear propagation could be achieved when the stamp is applied to the substrate. If they are indeed made to hang freely between the two fixed edges, some deformation of the stamp (near the second edge) might take place while the stamp face is parallel to the ground.

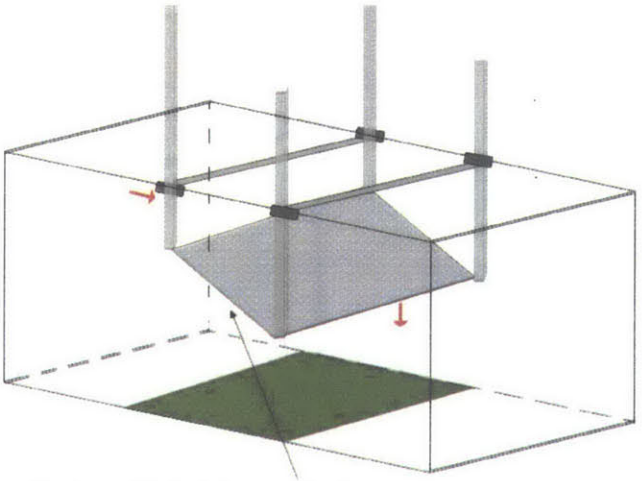
3. Pseudo Roll Stamping

The design derives it name from its similarity in concept to roll stamping. The stamp is applied to the substrate through a mechanism that resembles roll stamping with a roll of infinite diameter (which gives almost no deformation). The process is explained through Figure 4.3.1.3.

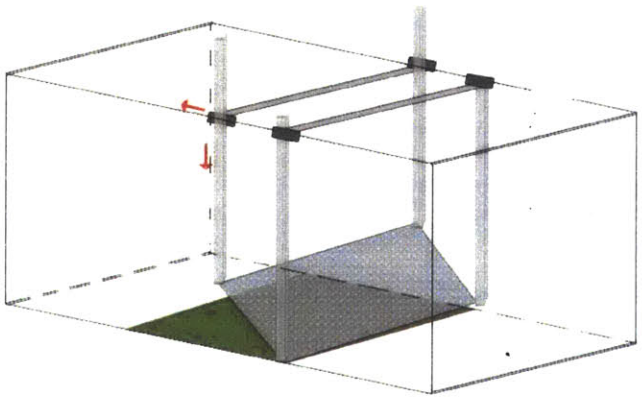

Stamp
Clean enclosure
CONVEYOR
Gold coated substrate
A.
Inking roller
Nitrogen blowers
CONVEYOR
B.



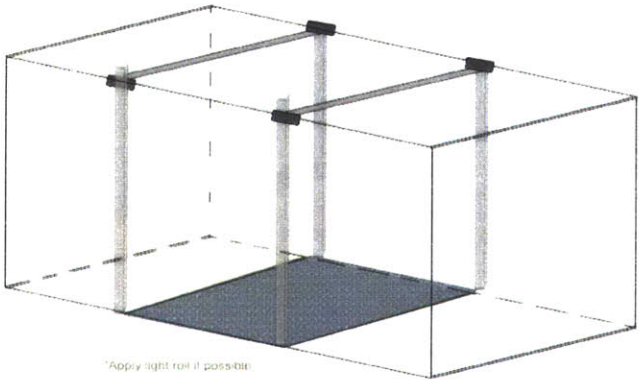
C.



D.



E.



F.

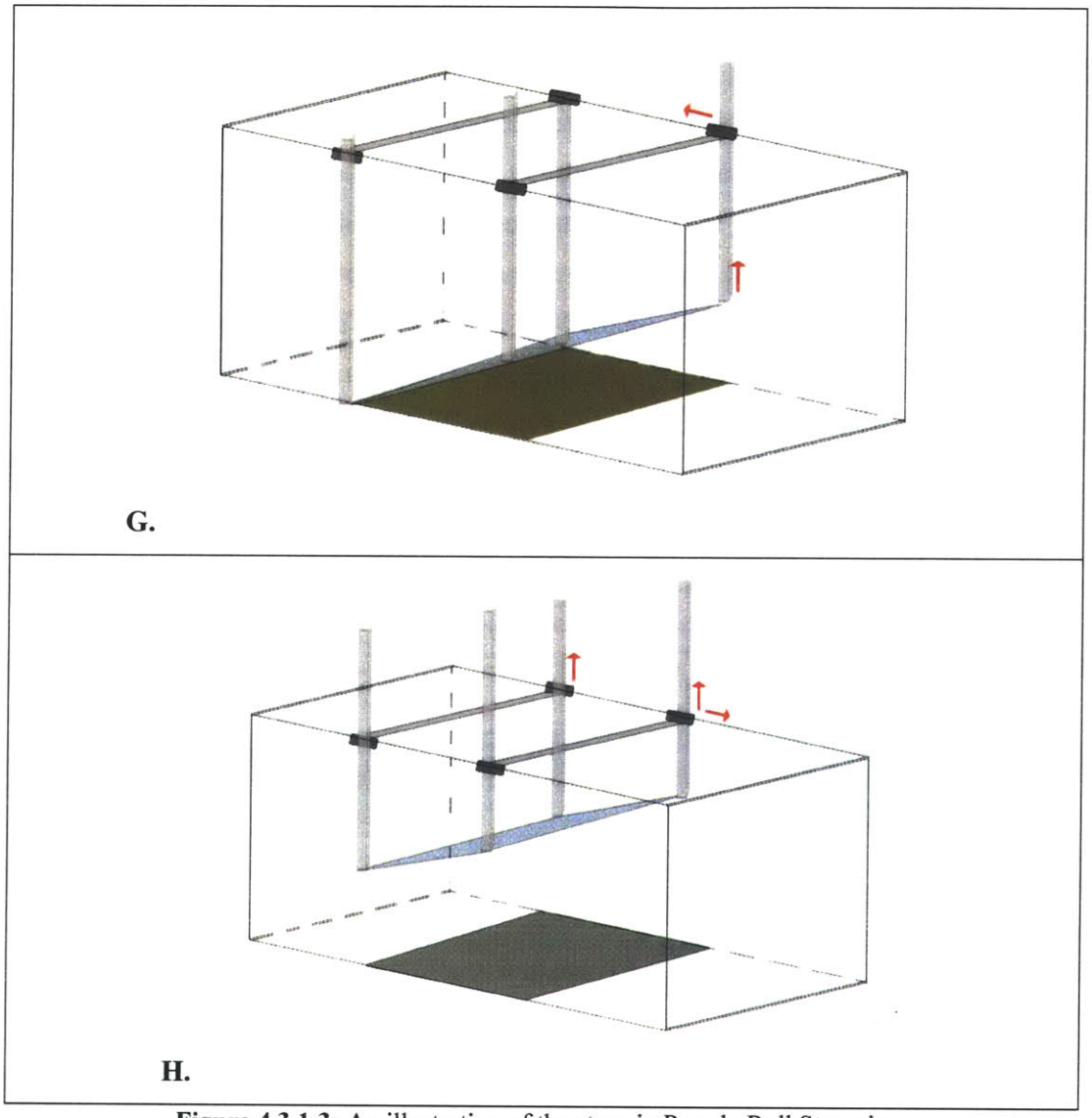


Figure 4.3.1.3: An illustration of the steps in Pseudo Roll Stamping.

The process discussed does not address any contact pressure adjustment means. In roll stamping this could be achieved by balancing the weight of the roller with counter-weights. In this design, this could be achieved by either using a thicker stamp or through a uniform stream of air.

4. The Finalized Stamping Design

We developed a reel to reel stamping design that gives linear propagation for substrate application as well as peeling off. The design utilizes the fact that the substrate (gold coated PEN film) is flexible and can be applied onto the PDMS stamp. The substrate application is achieved through two foam covered wipers whose motion in the vertical plane is controlled through two separate XY-Positioners. Referring to figure 4.3.1.4, the use of foam cover on the wiper core prevents any excessive pressure to be transmitted onto the stamp-substrate interface. This allowed us, for the sake of this prototype, to use a fully open loop based vertical position control system.

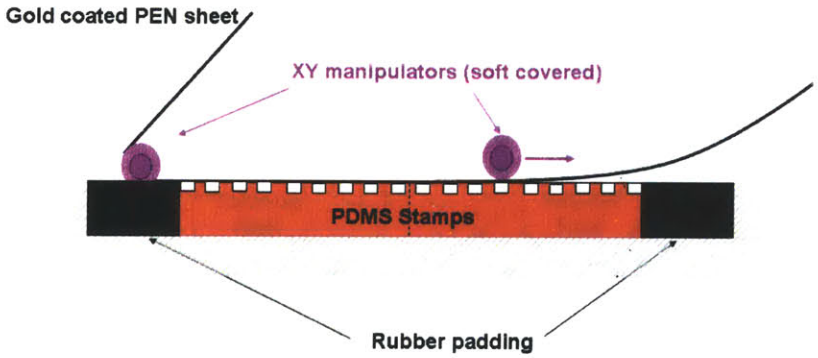
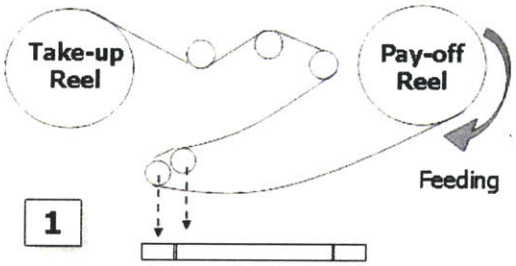
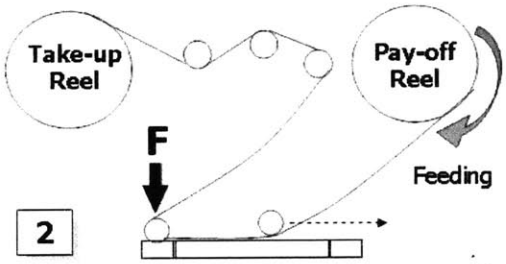


Figure 4.3.1.4: Two foam covered cylindrical wipers apply the gold coated PEN sheet onto the PDMS stamp.

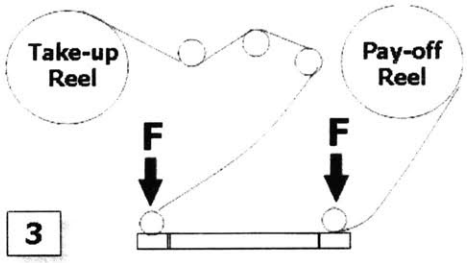
The process is explained, step by step, through figure 4.3.1.5.



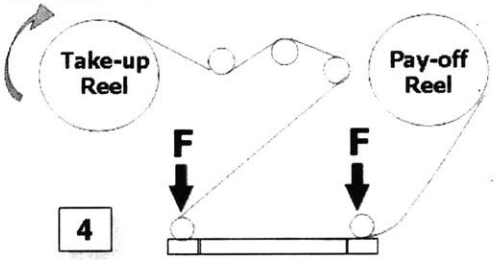
The two wipers move down to touch the stamping stage while the pay-off reel supplies the fresh substrate.



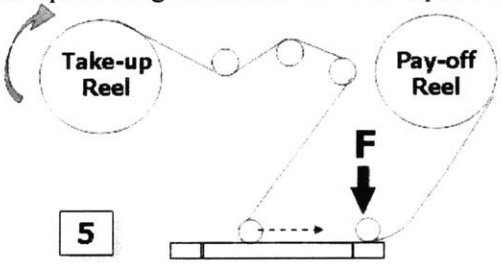
One wiper moves towards the pay-off-reel, applying the substrate onto the stamp in the process. The other wiper and the take-up reel are held fixed so as not to allow any substrate movement.



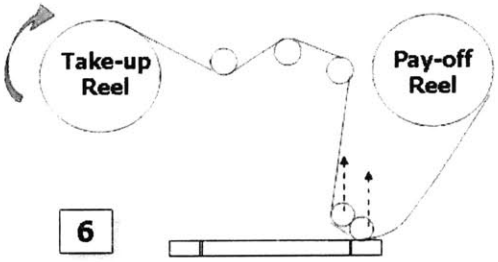
Both the wipers are held fixed for a period of time long enough to enable ink transfer between the stamp and the substrate.



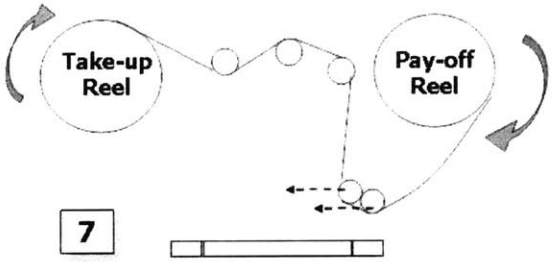
The take-up reel begins to collect the stamped substrate.



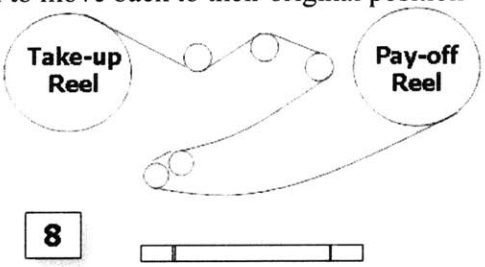
The second wiper moves towards the first wiper while the take up reel collects. This is the peel off step.



Both wipers begin to move upwards, away from the stamping stage. The take-up reel collects the substrate



Both the wipers begin to move back to their original position



Stamping cycle is complete. The used stamp is transferred onto the inking station and is quickly replaced by a freshly inked stamp.

Figure 4.3.1.5: A step by step illustration of the stamping design developed. [56]

4.4 A Manufacturing System for High Volume Production

For micro contact printing to succeed commercially, it is very important that the process be capable of producing products at production rates which cost less than the existing pattern transfer techniques. The most important goal while designing a manufacturing system for micro contact printing was to achieve the highest possible quality at a rate which was limited only by the chemistry of the process.

An integrated design of the inking and stamping processes above could be considered as the miniature model of the high-volume production machine without the etching step. We

will show later that inking station could be integrated with the stamping station so that the production throughput from the inking-stamping steps depends only on the ink transfer time in the stamping station. After being stamped, the substrate would be etched in an etch-bath as shown in figure 4.4. Because etching time is considerably larger than the inking or stamping times, the length of the substrate inside the etch-tub could be several miles long to allow continuously fast production. The process resembles hot rolling in the way the individual processing times are decoupled.

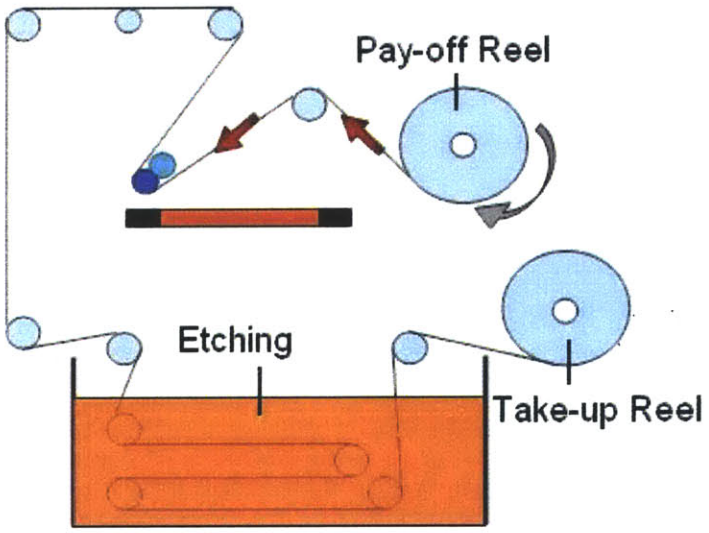


Figure 4.4.1: A schematic illustration of a process centric manufacturing system showing the integration between the inking-stamping steps and etching.

Another version of the manufacturing system showing higher level detail is shown in figure 4.4.2. Here, ink is applied to the stamp by applying an ink pad onto the stamp and leaving it for an appropriate period of time. A stamp with an ink pad residing on its top can still move on the conveyor. This decouples the stamping time and inking time from the production rate. The decoupling of the etching time from the production rate has already been discussed.

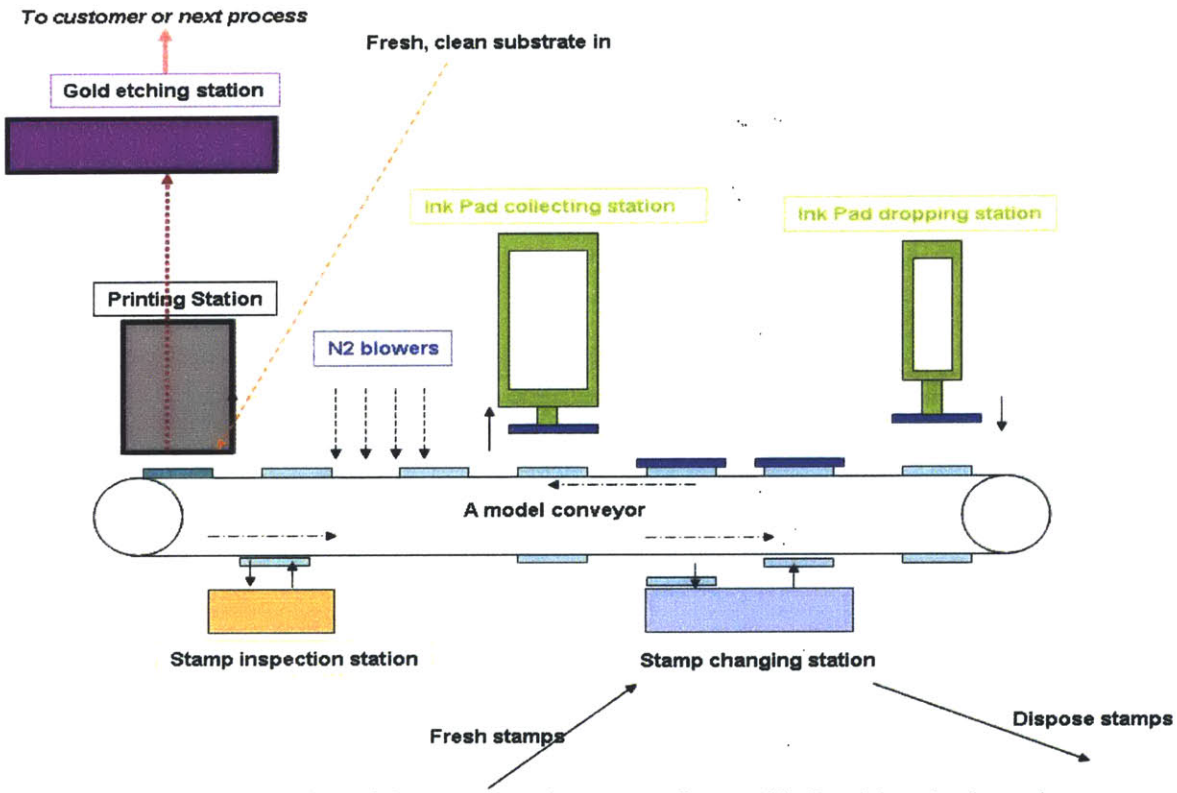


Figure 4.4.2: Another version of the manufacturing system discussed before. Note that here, the inking step is shown is different than the one we finalized. Note that the number of stamps on the belt between the 4 stations can be adjusted based on the time for inking, printing, removing etc.

The CAD models describing the individual components of the inking and stamping stations and their integration mechanism are shown in Appendix 3.

4.5 Conclusions

In this chapter, we described development of our design to achieve inking and stamping. We also discussed how the three steps for micro contact printing – inking, stamping and etching could be integrated for high volume production such that the production rate depends only on the chemistry of stamping. We hope that scientific advancements with a better process optimization or discovery of new materials could reduce this time further. In the next chapter, we shall discuss the automation of the prototype in detail.

Chapter 5

Automation Hardware and Software

5.1 Introduction

This chapter discusses the implementation details of the automation of the prototype. With a large number of coordinated motions to control, it was decided to deploy a PCI[3] based micro controller solution. Using a direct PC parallel or serial port based solution for motion control was ruled out given the need for real time multi-tasking for proper coordination of the motions and collection of the analog signals from the tension sensor and any other sensors to be deployed in the future.

This chapter first gives a high level overview of our automation approach, its justification and the unit elements. The hardware and software architecture of the PCI based solution from National Instruments[4] is briefly discussed, as well as the stepper motor and driver action. The chapter concludes with the kinematic analysis of the motions involved. The source code of the software application can be found in Appendix I.

[3] PCI (Peripheral Component Interconnect) is an interconnection system between a microprocessor and attached devices in which expansion slots are spaced closely for high speed operation. Using PCI, a computer's processor can communicate with the inserted PCI cards through a bus.
[4] NI-Motion.PCI-7344 and PCI-7350

5.2 The Automation Approach

It was desired to integrate the inking and stamping operations seamlessly to demonstrate the ease of integration and continuous production using micro contact printing. Therefore, all the steps in the process were automated and tightly coupled with each other. The major steps in sequence are as follows:

Two holistic approaches to achieve automation were considered:

1. A time based or an "Open loop" approach, in which each motion and the sequence of motions is hard coded into a program, is not monitored while in progress and is independent of other motions (except any electronic gearing). For example, a lead screw driven manipulator would be programmed to go from a point A to point B by simply giving the exact number of revolutions to the driving motor.

2. An event based or a "closed loop" approach, in which sensor signals (like those from encoders or switches) are treated as "events" which coordinate the motions. A motion may also be monitored continuously while in progress. For example, a lead screw actuator would be programmed to go from a point A to point B by simply giving motion signals to the motor until a "stop" signal is received from a limit switch at point B.

While an event based approach obviously ensures better accuracy and repeatability, a time based approach was chosen for the reasons below:

1. Limitation of time to acquire and integrate the sensors in the prototype.
2. The fact that stepper motors are less likely to lose steps and therefore are fairly accurate when running under low loads, allowed us to use a totally open loop based position control.

5.3 The Mechanical Elements for Automation

The mechanical elements to deliver the desired motions for the integrated process were carefully chosen with the following primary considerations:

1. Sufficient mechanical performance in terms of resolution, accuracy and repeatability.
2. Availability in terms of short vendor lead times.
3. Physical size to easily fit with other equipment and the machine structure.

Table 5.3 lists the mechanical equipment acquired along with their respective role in automating the process. Each of these components is discussed in brief in the following sub-sections.

Equipment	Role in Automation
2 belt-pulley driven XY-manipulators	To achieve the desired 2-D motion of each of the two wipers for stamping.
1 Vertical Lead screw slide	To raise and lower stamp retainers to and from the stamping stage respectively.
1 Horizontal Lead screw	To drive the inking roller back and forth.
1 Rotary positioning table	To convey stamps between the inking and stamping station.

Table 5.3: The mechanical equipment sourced for prototype development.

5.3.1 The XY-Manipulators

The motions of the wipers in the vertical plane to achieve substrate application on the stamp, substrate peel off and return to the origin was accomplished through two xy-manipulators, one for each wiper.

The two manipulators were acquired off the shelf from a vendor[5]. The range of motions in either direction was 9 inches. The following changes were made to each of the manipulators to fit into our requirements:

1. The back plane of each was replaced by a transparent polycarbonate sheet for better visibility of the actions taking place inside the stamping station.
2. The flanged bronze bushings, supporting the sliding motion on the unhardened steel shafts, were replaced by nylon bushings which gave much smoother motion with almost no vibration.
3. The stepper motor supplied by the vendor was replaced by one with half the weight (and also easily capable of micro stepping)

The principle of operation of these manipulators is straight forward. A stepper motor drives a timing belt-pulley, which can move the manipulator-head in one direction. Any two dimensional motion, within the travel range in each direction, is achieved by properly coordinating the movements of two stepper motors. For example, to move the manipulator head in a straight line from a point *P1:* (x_1,y_1) in the vertical plane to a point *P2:* (x_2,y_2) is achieved as follows.

If the diameter of the pulley is D inches, then one full revolution of the "X Motor" will result in a horizontal motion of πD inches. Most stepper motors rotate 1.8° per full step, that is 200 steps per revolution. If M is the microstepping index used, then one revolution of the motor is achieved with 200M steps or counts. Thus, it can be easily shown that to move by 'x' inches along the x direction, $\frac{200Mx}{\pi D}$ steps have to be given to the "X motor".

Thus, to move from (x_1,y_1) to (x_2,y_2) is achieved by giving the "X Motor $\frac{200M(x_2-x_1)}{\pi D}$ steps and electronically gearing the "Y Motor" as the slave to the "X Motor", with a

[5] Arrick Robotics, http://www.robotics.com/xytables.htm

constant gear ratio of $\frac{(y_2 - y_1)}{(x_2 - x_1)}$. The step rate to achieve the desired travel velocity can be easily calculated using the above relation.

5.3.2 The Rotary Positioning Table

The rotary table acts as a coupler between the inking and stamping operations by conveying PDMS stamp retainers back and forth between the two stations. The table is timing belt-pulley driven with a 6:1 reduction from the motor axis pulley to the table pulley.

5.3.3 The 2 Lead Screws for Inking and Lifting

1. Lead Screw Actuator for the Inking Station

The inking roller which collects ink on its surface over a pad soaked with alkanthiol (the ink) before applying the ink on the PDMS stamp, rolls on a urethane timing belt fixed to parallel tracks. The forward movement is provided by a precision lead screw that has a pitch of 0.1", straightness error of less than 1 micron per foot of travel and a total travel length of 28.4 inches. This lead screw is rigidly mounted parallel to the

It can be shown by simple calculations that to travel '**x**' inches at a speed of '**v**' inches per second, 200M**x**/p counts at 200M**v**/p counts per second have to be given to the directly coupled driving stepper motor, where p is the pitch of the lead screw (inches/revolution) and M is the microstepping index.

2. Lead Screw Actuator for the Vertical Lift

When a fresh stamp reaches the stamping station from the inking station, it is lifted to the level of the stamping stage by a lead screw actuator. The manipulator is a flat head covered with PVC sheet.

The bottom surface of the stamp retainer is also covered with PVC sheet to allow slipping between the manipulator head and the bottom surface of the retainer so that retainer is free to move in the horizontal plane for alignment while rising upwards along the guide pins. The distance and speed calculations are the same as those for the inking lead screw actuator.

5.4 The Electronic and Software Components for Automation

The electronic and software components to achieve automation are illustrated through figure 5.4. Each of them shall be explained in detail in a bottom to top approach in the following sections.

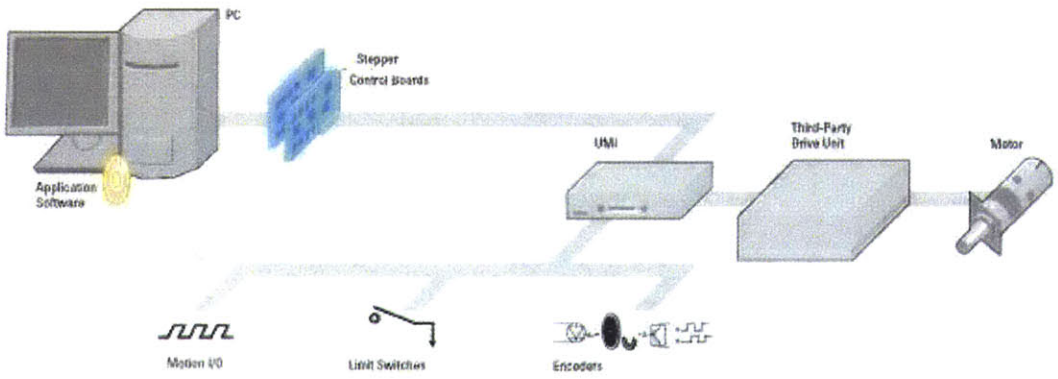


Figure 5.4: A device level schematic of the hardware-software integration for automation [32].

5.4.1 The Motors and Motor Drivers

Stepper motors were used for giving motion to the entire 9 axis. They were preferred over servo motors for the reasons of simplicity, cost and ease of programming. All motors were used in open loop mode (that is, without any optical encoders for position feedback) because low operating loads implied less likelihood of the motor missing steps.

All stepper motors acquired were bipolar stepper motors (with two phases) to allow for flexibility of choice between high speed and low speed operations and better compatibility with the microstepping driver.

Because we were using microstepping up to 4000 micro steps per revolution and wanted high speed operations, all the stepper motors were used in a bipolar parallel mode, that is, both coils of the two phases (Phase A and Phase B) were connected in parallel to ensure minimum torque loss at high speeds. This is explained as follows:

The torque of a stepper motor is directly proportional to the current flowing through the motor coils. The current must ramp up to +100%, then -100% 50 times per revolution (200 quadrature states for a stepper motor of 1.8 degree step angle). At higher speeds the current does not have enough time to ramp up to maximum. Before it reaches the maximum, it must reverse and go in the other direction. Since the maximum rate of current change is inversely proportional to the square of the inductance, high speed with minimum torque loss can be obtained by minimizing the coil inductance. Compared to a bipolar series mode, bipolar parallel mode has one fourth the inductance because both coils of either phase are connected in parallel. Therefore, a stepper motor in bipolar parallel mode will give better torque performance at higher speed compared to a stepper motor operating in bipolar series mode. Bipolar serial is proffered for low speed applications due to lesser consumption of power.

Figure 5.4.1 shows the speed-torque characteristics[6] of the stepper motors used and table 5.4.1 gives the other important specifications.

[6] As supplied by the vendor: USDigital (www.usdigital.com)

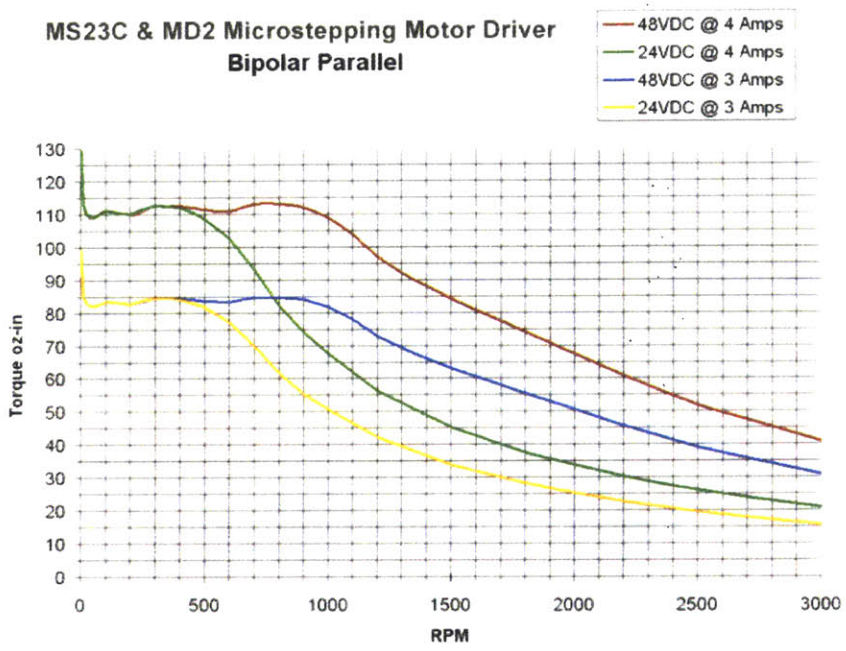


Figure 5.4.1: Speed-torque characteristics for the Motor. Source: USDigital

Parameter	Value
NEMA size	23
Full step angle	1.8 degrees
Full step accuracy	+-5%
Full Step Current	3.00 Amp
Micro Step Current	4.20 Amp
CD Resistance	0.85 ohms
Inductance	1.73 mH
Power Total	15.3 Watts
Holding Torque (static torque)	140 oz-in
Running Torque	110 oz-in

Table 5.4.1: MS23 Stepper motor specifications. Source: USDigital

Besides the stepper motor itself, any stepping motor system requires a Driver (or Amplifier) that converts the controller command signals into the power necessary to energize the motor windings.

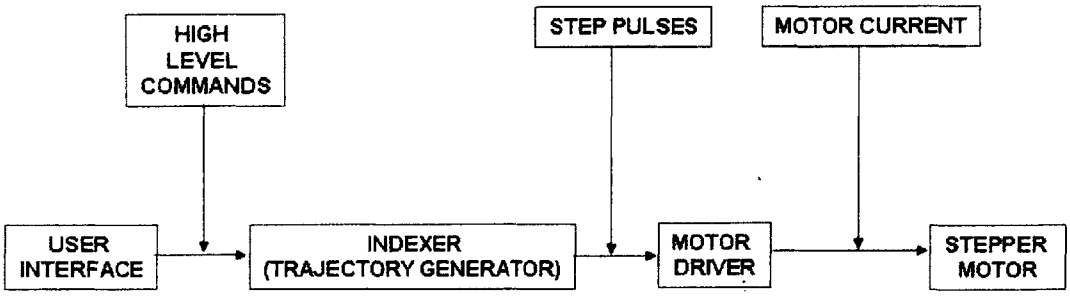


Figure 5.4.2: The basics of a stepper motor system.

The following were the considerations for selecting the drivers:

1. Compatibility with the bipolar stepper motors acquired
2. Provision for easily programmable microstepping
3. Should take step and direction as the inputs to easily integrate with the PCI microcontroller which controls all the motions programmatically.

Figure 5.4.2 shows the layout of the acquired driver with its key features and integration with the motion controller and the motor.

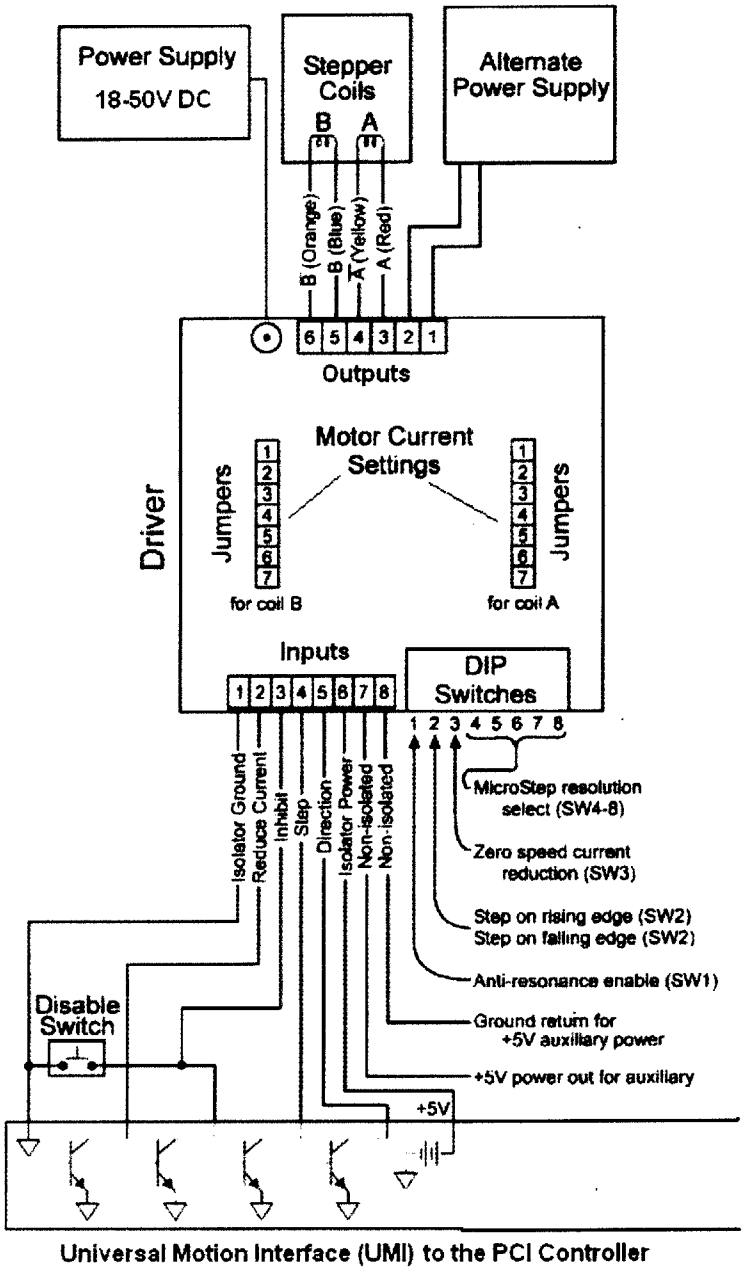


Figure 5.4.2: The layout of the acquired driver MD2S-P. Source: USDigital.

5.4.1.1 Microstepping

Microstepping is a stepper motor technology that controls the current in the motor winding to a degree that further subdivides the number of positions between poles. Microstepping was favored in our application to guarantee smooth motion with higher resolution, particularly given the fact that the XY positioners came with motors and drivers which gave unacceptable vibration levels.

Besides, microstepping can greatly reduce the possibility of resonance in a stepper motor- which could be detrimental to a manufacturing process like micro contact printing. It is known that [13] that the energy transferred to the rotor, when a single step is taken, is in the worst case (no load friction) equal to:

$$\frac{4T_H(1-Cos(\alpha_{electrical})}{n}$$

where T_H is the holding torque, n is the number of steps per revolution and $\alpha_{electrical}$ is the electrical angle[7]. Therefore, by increasing n to a very high value using microstepping, the energy transferred to the rotor becomes almost negligible and hence protecting from resonance.

5.4.2 Motor Driver-Controller Interface

To interface with the PCI Microcontroller and feed the analog inputs (from the tension sensor and other future sensors), a Universal Motion Interface (UMI) from National Instruments Inc were employed. Each UMI could interface up to 4 stepper drives and with the PCI card through a multicore cable. Since 9 motor drivers had to be connected, 3 separate UMI devices were used.

5.4.3 The PCI Microcontroller Architecture

[7] Electrical angle is related to the mechanical angle as $\alpha_{electrical}=(n/4)\ \alpha_{mechanical}$

The NI-Motion driver software architecture is based on the interaction between the PCI based NI motion controllers and a host computer. This interaction includes the hardware and software interface and the physical and functional architecture of the NI motion controllers.

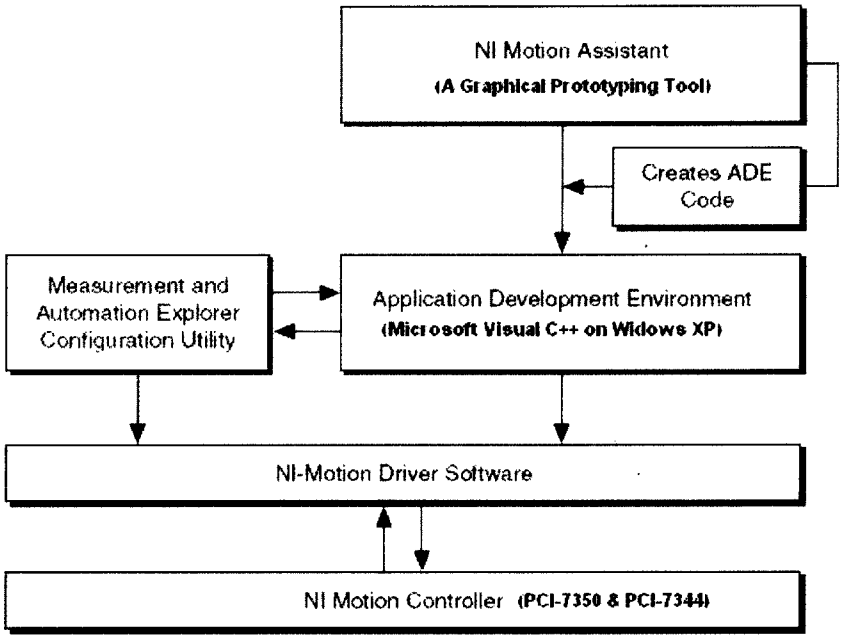


Figure 5.4.3: NI Motion Control Hardware and Software Interaction [32]. A typical motion control application can be prototyped with a GUI based software called "NI Motion Assistant". The GUI software allows one to create Application Development Environment (ADE) code which can be enhanced using an ADE like Microsoft VC++. The Measurements and Automation Explorer (MAX) allows one to configure the boards attached to the PCI slots and update any firmware. Together, MAX and the C++ code use the NI-Motion driver software (which contains the necessary libraries) to control the boards for any motion.

5.4.3.1 NI Motion Controller Architecture

NI 7350 and NI 7330 controllers use a dual-processor architecture. The controller CPU is a 32-bit micro-controller running an embedded real time, multitasking operating system. The CPU performs command execution, host synchronization, I/O reaction, and system supervision. The DSP has the primary responsibility of fast closed-loop control with simultaneous position, velocity, and trajectory maintenance on multiple axes. The

DSP is also responsible for closing the position and velocity loops, and directly commands the torque to the motor drivers.

Motion I/O occurs in hardware on an FPGA (Field Programmable Gate Array) and ensures very low latencies in the range of hundreds of nanoseconds for operations like limit detection. The motion controller processor is monitored by a watchdog timer, which is hardware that can be used to automatically detect software anomalies and reset the processor if any occur. The watchdog timer checks for proper processor operation. If the firmware on the motion controller is unable to process functions within 62 ms, the watchdog timer resets the motion controller and disallows further communications until the motion controller is explicitly reset.

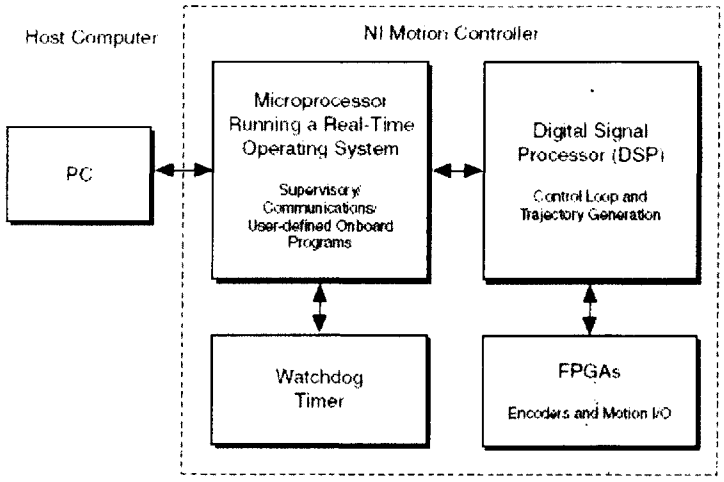


Figure 5.4.3.1: Physical NI Motion Controller Architecture [32].

5.4.3.2 NI Motion Controller Functional Architecture

Functionally, the architecture of the NI 73xx motion controllers is generally divided into four components:

(A) Supervisory control

Performs all the command sequencing and coordination required to carry out the specified operation including:

- *System initialization*, which includes homing operations.
- Event handling, which includes electronic gearing , electronic camming, triggering outputs based on position and so on.
- *Fault Detection*, which includes stopping moves on a limit switch encounter, safe system reaction to emergency stop or drive faults, watchdog, and so on.

(B) Trajectory Generator

Provides path planning based on the profile specified by the user.

(C) Control Loop

Performs closed-loop control with simultaneous position, velocity, and trajectory maintenance on one or more axes. For stepper systems, the control loop is replaced with a step generation component.

(D) Motion I/O

Analog and digital I/O that sends and receives signals from the rest of the motion control system. Typically, the analog output is used as a command signal for the drive, and the digital I/O is used for quadrature encoder signals as feedback from the motor. Also, the supervisory control uses the motion I/O to achieve certain required functionality, such as reacting to limit switches and creating the movement modes needed to initialize the system.

The above mentioned components are shown in figure 5.4.3.2.

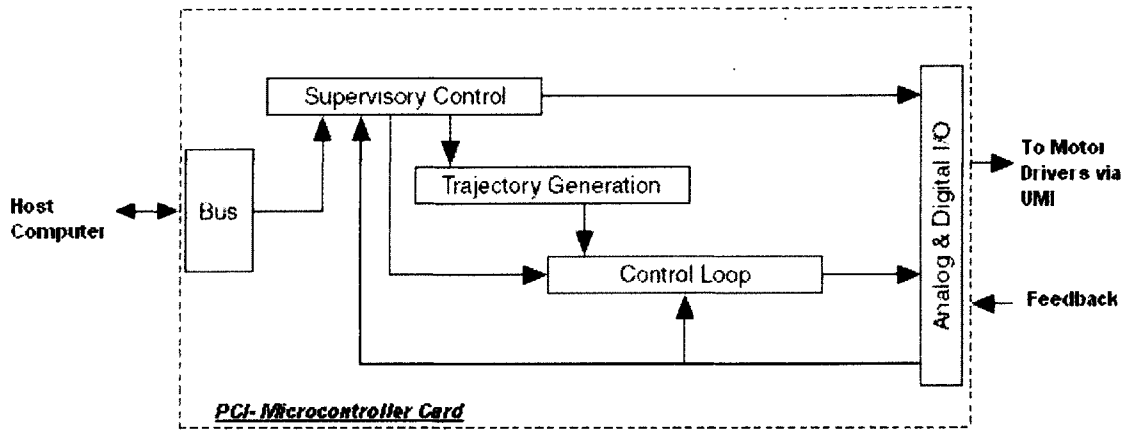


Figure 5.4.3.2: Typical NI 73xx Motion Controller Functional Architecture [32].

5.5 Kinematic Analysis of the Motion

Since we were using a totally open loop approach to motions where each axis was operated independently (besides electronic gearing), in order to ensure that the substrate is neither overstretched nor held too loose during the stamping, it was essential to orchestrate the motions of each axis mathematically with respect to time and code them into the software. In the case of electronic gearing where it was used, it was necessary to calculate the appropriate gear ratio using the kinematics of the coupled motion.

This section will explain each of the unit motion steps with the kinematical analysis in the order that they occur on the prototype machine. We shall analyze motions only for the stamping process because the motions for inking did not involve any electronic gearing.

The following sign conventions have been used:

1. Positive X-direction is the one from the drive spool to the supply spool.
2. Positive Y-direction is the one perpendicular to the face of the floor and upwards.
3. The origin of the XY plane is the point where the first wiper (from XY-Positioner 1) touches on the stamping stage.
4. For the spools, positive direction is clockwise rotation.

5. For the inking lead screw, positive direction is towards the stamping station.
6. XY-Positioner1 is the one closest to the rotary table, the other positioner shall be called XY-Positioner2.
7. The following is the name convention used for each of the 9 axes of motion with direction sense defined as above:

X_XY1	The axis for the x-motion of XY-Positioner1
Y_XY1	The axis for the y-motion of XY-Positioner1
X_XY2	The axis for the x-motion of XY-Positioner2
Y_XY2	The axis for the y-motion of XY-Positioner2
DRIVE	The drive spool
SUPPLY	The supply spool
LIFT	The lead screw for vertical lift motion
ROT	The rotary table
INKING	The inking lead screw

Table 5.5: Name convention used for each axis. Note that XY-Positioner 1 is the one that is closest to the rotary table.

(A) Move 1

Description

The axes Y_XY1 and Y_XY2 travel along the negative Y axis and touch the stamping floor. The axis SUPPLY is electronically geared to Y_XY2 so that the substrate is not stretched during motion. DRIVE is held *locked.* Figure 5.5a gives a view of the xy plane during this motion step and the analysis to determine the gear ratio between SUPPLY and Y_XY2 follows.

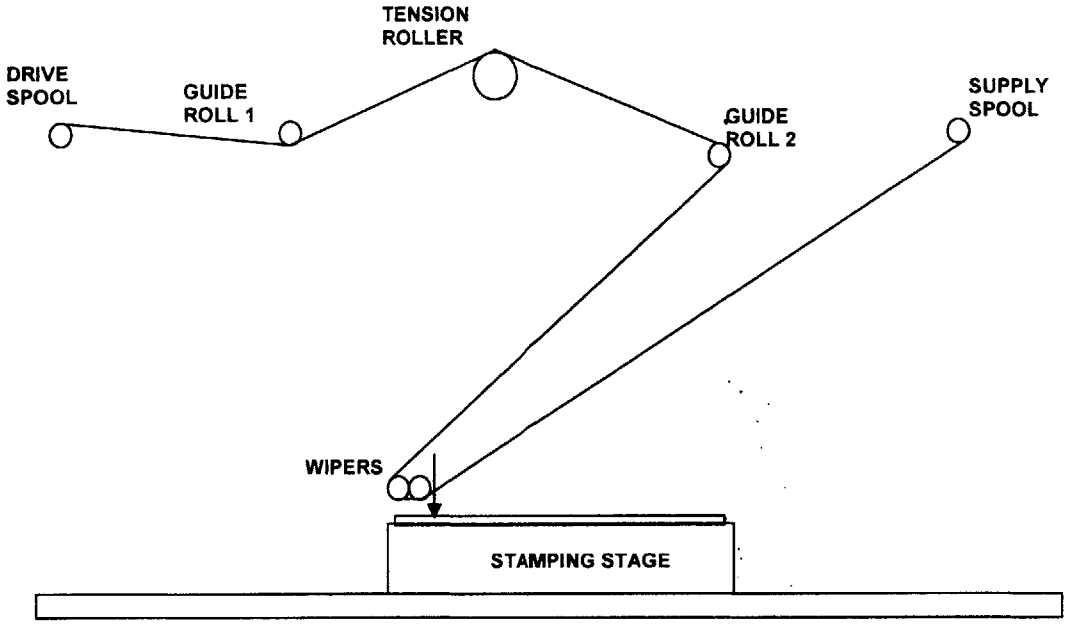


Figure 5.5.A: A simplified illustration of the elements involved in the stamping step.

Kinematic Analysis

We assume the following:

1. The wipers, the supply spool and the guide roll are point objects, i.e, we are ignoring the wrapping of the substrate about them and also the changing diameter of the spool as it supplies the substrate.
2. We assume that the wiper moves at a constant speed v and ignore the acceleration and deceleration of the wiper motion for simplicity.

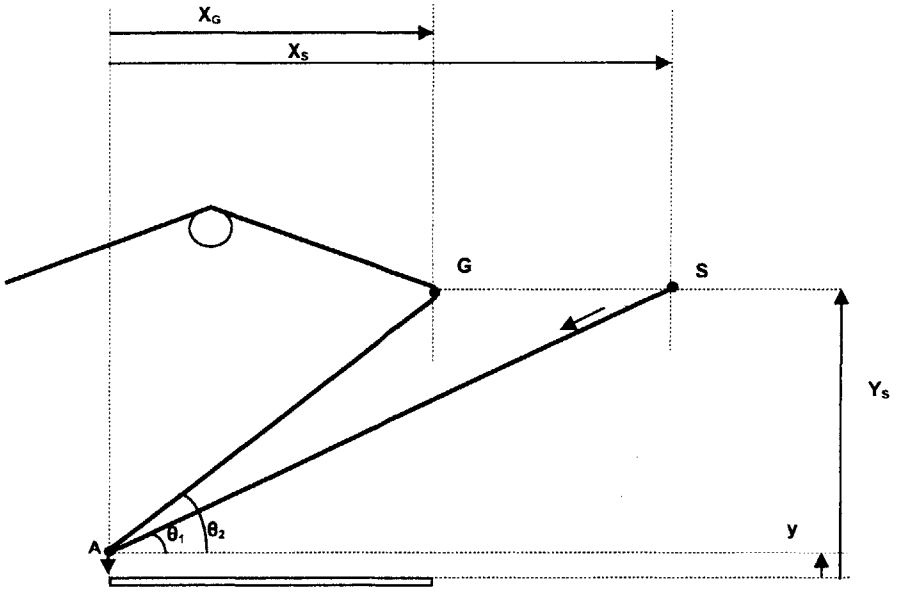


Figure 5.5.B: Kinematic analysis for Move 1.

With reference to figure 5.5.B, consider the length SA. Because the total length of the substrate is to be constant and with our assumptions declared above, we have:

$$\frac{dGA}{dt} = \frac{dSA}{dt} - r\omega(t) \qquad \ldots\ldots\ldots\ldots \quad (1)$$

where $\omega(t)$ is the angular speed and r is the radius of the supply spool.

From geometry,

$$SA = \sqrt{(Y_S - Y)^2 + X_S{}^2} = \frac{(Y_s - Y)}{Sin\theta_1} = \frac{X_S}{Cos\theta_1}$$

and $GA = \sqrt{(Y_S - Y)^2 + X_G{}^2} = \frac{(Y_s - Y)}{Sin\theta_2} = \frac{X_G}{Cos\theta_2}$, where $Y = vt$

$$\Rightarrow \frac{dSA}{dt} = \frac{X_S Sin\theta_1}{Cos^2\theta_1}\frac{d\theta_1}{dt} \quad \& \quad \frac{dGA}{dt} = \frac{X_G Sin\theta_2}{Cos^2\theta_2}\frac{d\theta_2}{dt} \qquad \ldots\ldots\ldots\ldots \quad (2)$$

Also, $\tan\theta_1 = \frac{(Y_S - vt)}{X_s}$ & $\tan\theta_2 = \frac{(Y_S - vt)}{X_G}$

$$\Rightarrow \frac{d\theta_1}{dt} = \frac{vCos^2\theta_1}{X_S} \quad \& \quad \frac{d\theta_2}{dt} = \frac{vCos^2\theta_2}{X_G} \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

Using (1), (2) and (3), we get:

$$\omega(t) = \frac{v}{r}\left[X_S Sin\theta_1 - X_G Sin\theta_2\right]$$

For a speed of v inches/sec, the `Y_XY2` axis must rotate at $\frac{v}{r}$ rad/sec where r is the radius of the pulley of the XY Positioner. The electronic gear ratio, thus, is:

$$G_E = \frac{r\omega(t)}{v}$$

Figures 5.5.1.1 and 5.5.1.2 show the actually required gear ratio and the one used.

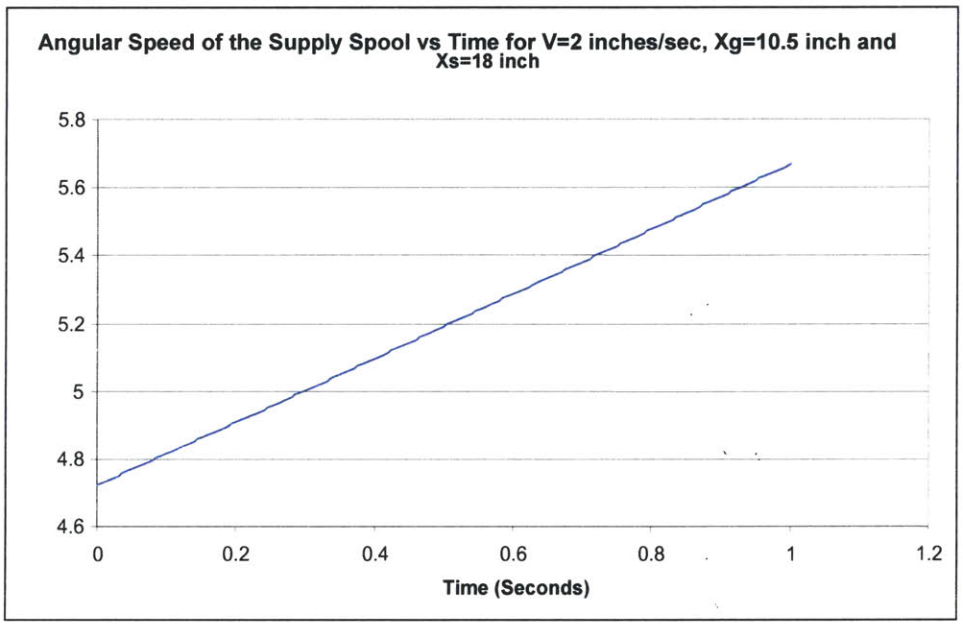


Figure 5.5.1.1: The calculated angular speed of the supply spool vs time.

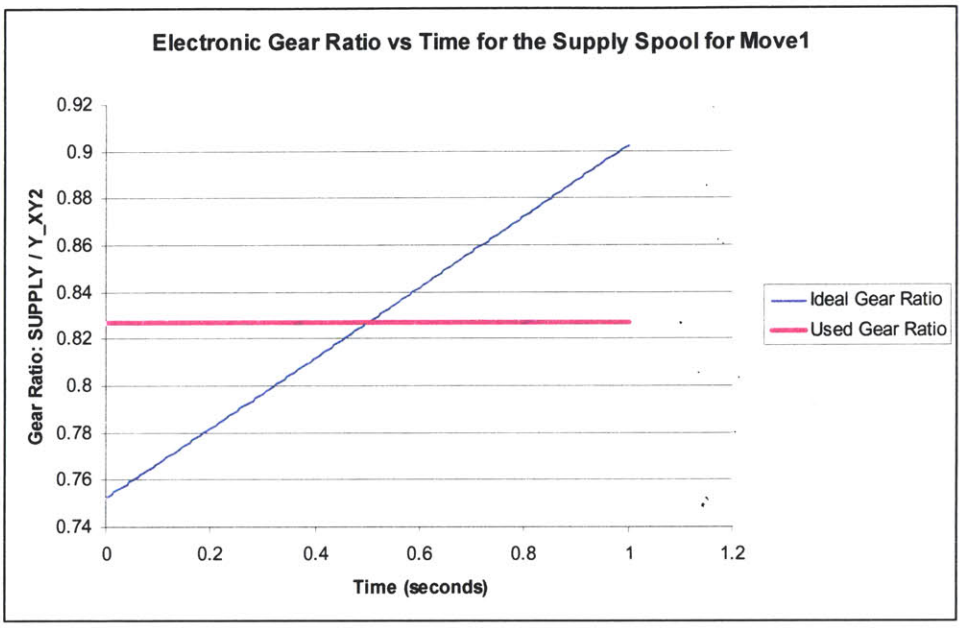
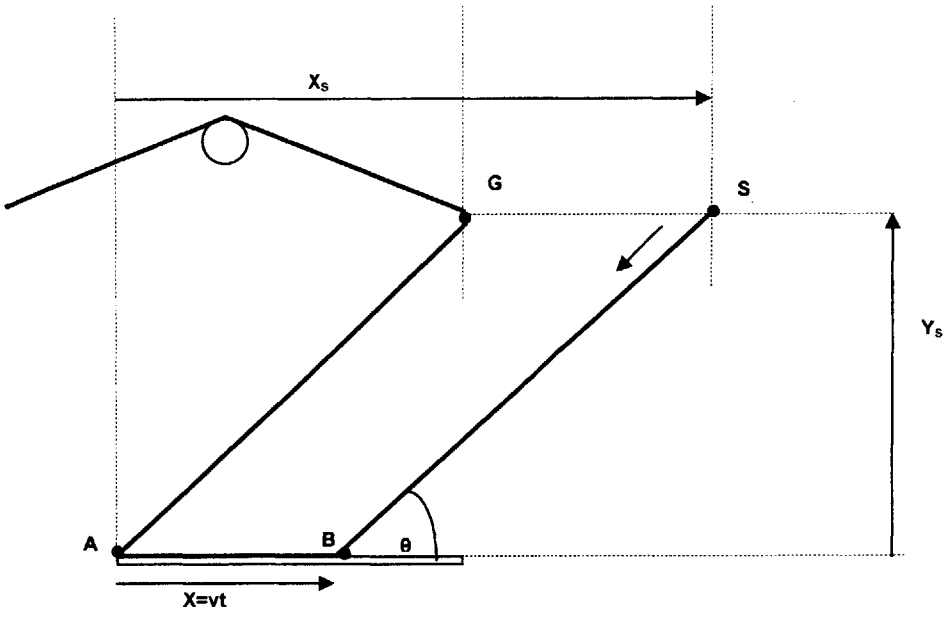


Figure 5.5.1.2: The computed and used electronic gear ratios. Note that a liberal gearing ratio was used so that the substrate is not overstretched. A compensation move will absorb all the excess substrate released.

(B) Move 2

Description

The wiper corresponding to the axis `X_XY2` sweeps across the stamping stage while spreading the substrate on the PDMS stamp. The axis `DRIVE` is still locked and the substrate is supplied by the supply spool. The axis `SUPPLY` is electronically geared to the axis `X_XY2`.



Kinematic Analysis

Again, $v - \frac{dSA}{dt} = r\omega(t)$ ..... (1)

From geometry, $SA = \sqrt{(Y_S{}^2 + (X_S - vt)^2}$

$$\Rightarrow \frac{dSA}{dt} = \frac{-v(X_S - vt)}{\sqrt{Y_S{}^2 + (X_S - vt)^2}} \quad \text{..... (2)}$$

From (1) and (2),

$$\Rightarrow \omega(t) = \frac{v}{r}\left[1 + \frac{(X_S - vt)}{\sqrt{(Y_S{}^2 + (X_S - vt)^2}}\right]$$

$\Rightarrow$ Gear ratio: $G_E = \frac{r\omega(t)}{v}$

Figures 5.5.2.1 and 5.5.2.2 show the ideal gear ratio and the one used.

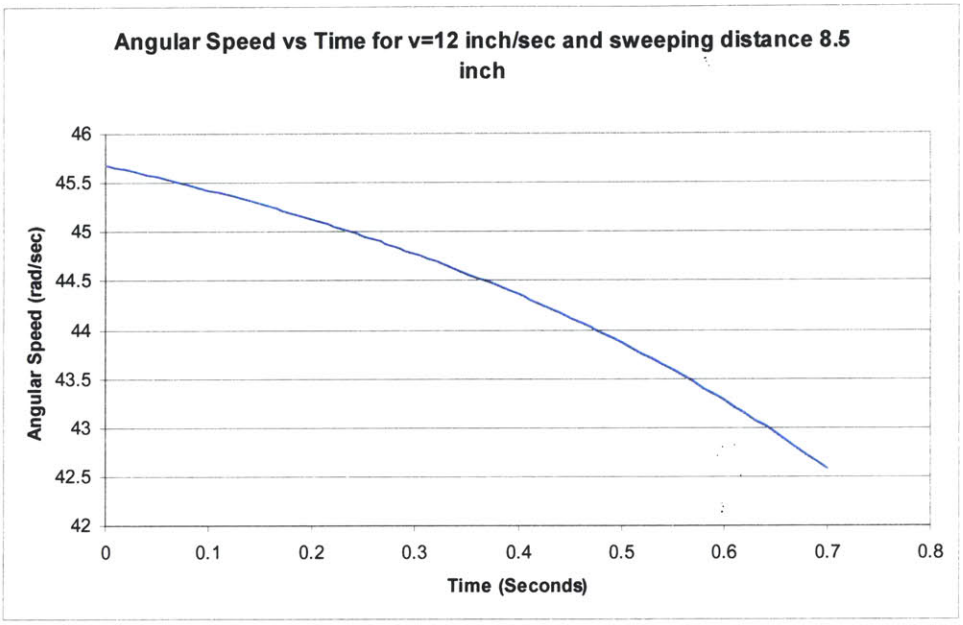


Figure 5.5.2.2: The calculated angular speed vs time for the SUPPLY axis for Move 2.

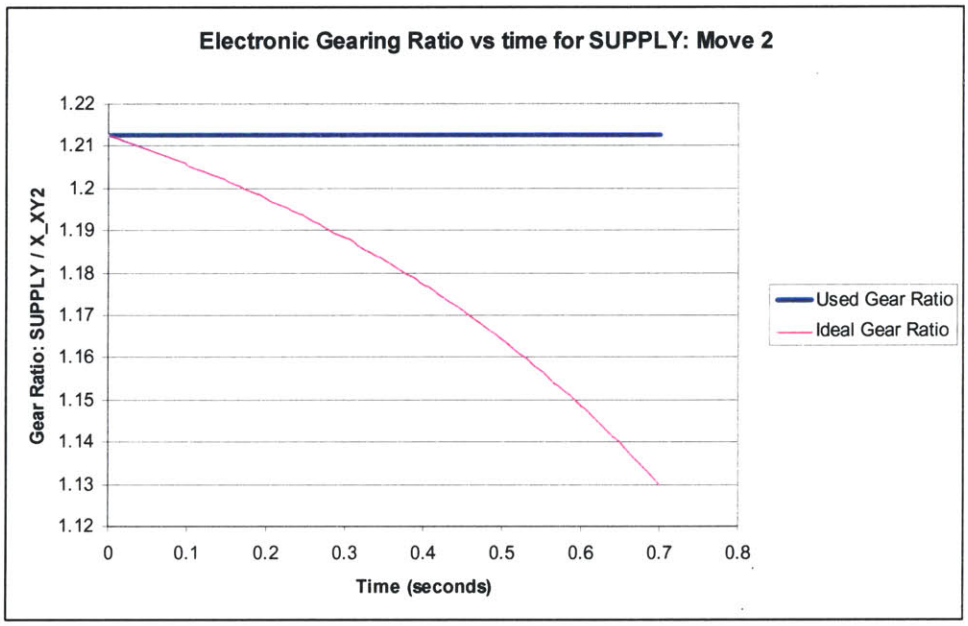


Figure 5.5.2.2: The computed and used electronic gear ratios. Note that a liberal gearing ratio was used so that the substrate is not overstretched. A compensation move will absorb all the excess substrate released.

(C) Move 3

Description

Move 3 is an angular compensation move because of the liberal gear ratio employed by Move 2. The angular compensation is easily calculated as:

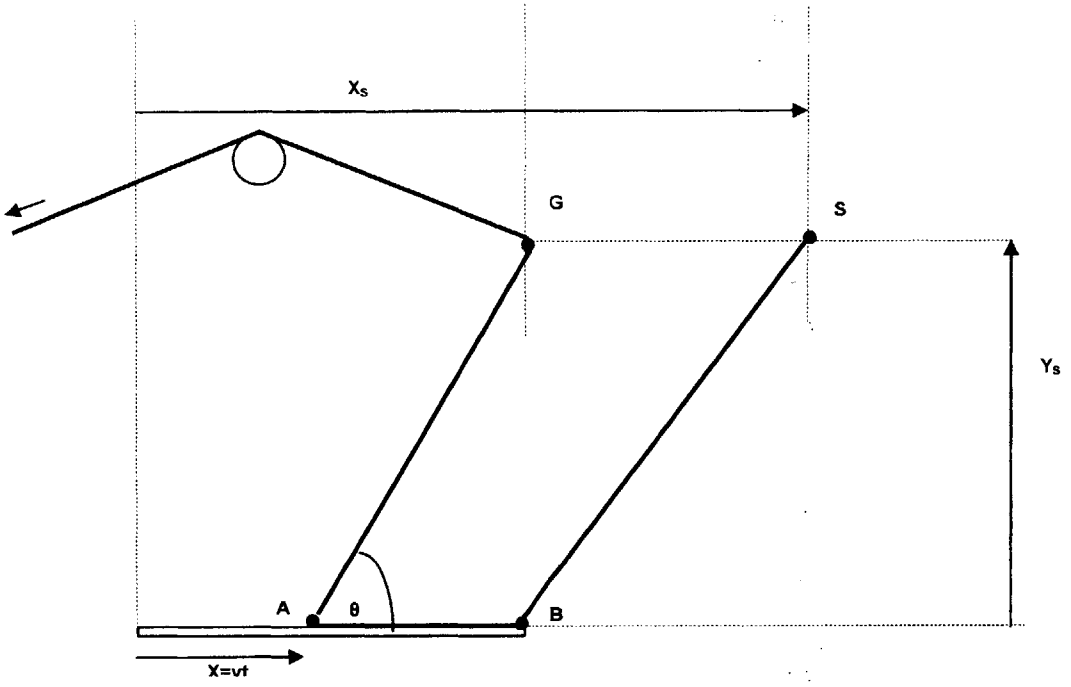
$$\Delta\theta = \int \omega_{USED}(t)dt - \int \omega_{IDEAL}(t)dt$$

Where ω_{USED} and ω_{IDEAL} correspond to the actually used and the ideal gear ratio respectively for Move 2.

(D) Move 4

Description

The wiper corresponding to the axis `X_XY1` sweeps across the stamping stage while peeling off the substrate from the PDMS stamp. The axis SUPPLY is locked and the substrate is absorbed by the DRIVE axis spool. The axis DRIVE is electronically geared to the axis X_XY1.



Kinematic Analysis

$$v - r\omega(t) = \frac{dGA}{dt} \qquad \ldots\ldots\ldots\ldots\ldots \qquad (1)$$

From Geometry,

$$Sin\theta = \frac{Y_S}{GA} = \frac{Y_S}{\sqrt{Y_S^{\,2} + (X_G - vt)^2}}$$

$$\Rightarrow \frac{dGA}{dt} = -Y_S \frac{Cos\theta}{Sin^2\theta} \frac{d\theta}{dt} \qquad \ldots\ldots\ldots\ldots\ldots \qquad (2)$$

where $Cos\theta = \dfrac{(X_G - vt)}{\sqrt{Y_S^{\,2} + (X_G - vt)^2}}$

Also, $\tan\theta = \dfrac{Y_S}{(X_G - vt)}$

$$\Rightarrow \frac{d\theta}{dt} = vY_S \frac{Cos^2\theta}{(X_G - vt)} \qquad \ldots\ldots\ldots\ldots\ldots \qquad (3)$$

From (1), (2) and (3),

$$\omega(t) = \frac{v}{R}\left[1 + \frac{(X_G - vt)}{\sqrt{Y_S{}^2 + (X_G - vt)^2}}\right]$$

$\Rightarrow$ Gear ratio: $G_E = \dfrac{r\omega(t)}{v}$

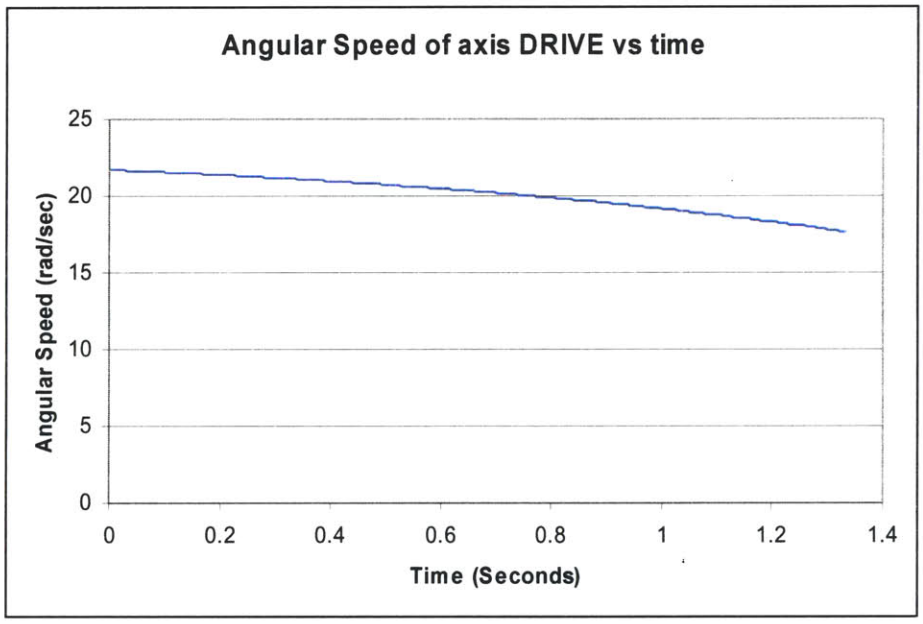


Figure 5.5.3.1: The calculated angular speed of the DRIVE axis vc time for Move 4.

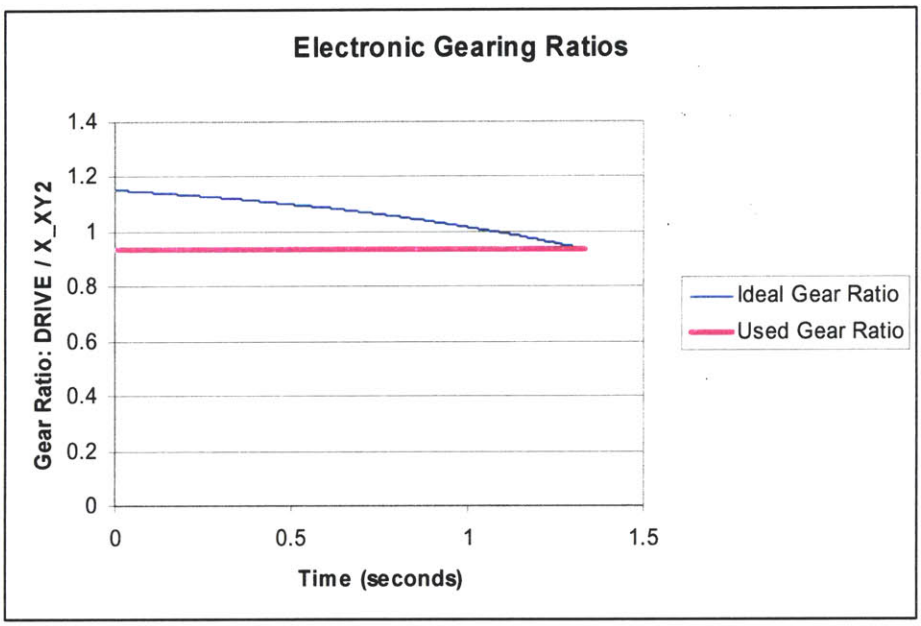


Figure 5.5.3.2: The computed and used electronic gear ratios. Note that a liberal gearing ratio was used so that the substrate is not overstretched. A compensation move will absorb all the excess substrate released.

(E) Move 5

Description

Move 5 is an angular compensation move because of the liberal gear ratio employed by Move 4. The angular compensation is easily calculated as:

$$\Delta\theta = \int \omega_{IDEAL}(t)dt - \int \omega_{USED}(t)dt$$

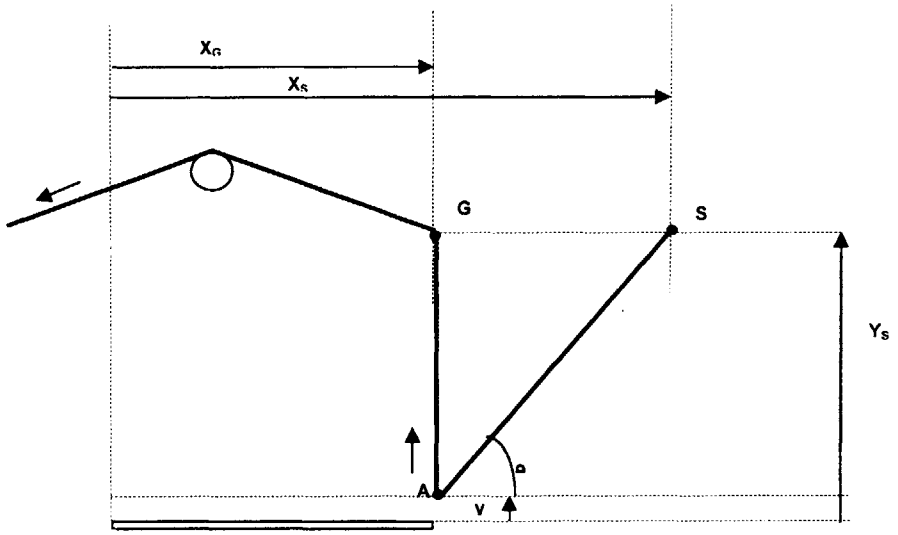
Where ω_{USED} and ω_{IDEAL} correspond to the actually used and the ideal gear ratio respectively for Move 4.

(F) Move 6

Description

The two wipers lift off the stamping stage. The axis DRIVE is electronically geared to the axis Y_XY2 .

.



Kinematic Analysis

$$\frac{dGA}{dt} - r\omega(t) = \frac{dSA}{dt}$$

$$\Rightarrow v - r\omega(t) = \frac{dSA}{dt} \qquad \ldots\ldots\ldots\ldots\ldots \qquad (1)$$

From Geometry,

$$Cos\theta = \frac{(X_G - vt)}{SA}$$

$$\Rightarrow \frac{dSA}{dt} = (X_S - X_G)\frac{Sin\theta}{Cos^2\theta}\frac{d\theta}{dt} \qquad \ldots\ldots\ldots\ldots\ldots \qquad (2)$$

Also, $\tan\theta = \dfrac{(Y_S - vt)}{(X_S - X_G)}$

$$\Rightarrow \frac{d\theta}{dt} = -v\frac{Cos^2\theta}{(X_S - X_G)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots \qquad (3)$$

Using (1), (2) and (3),

$$\omega(t) = \frac{v}{R}\left[1 + \frac{(Y_S - vt)}{\sqrt{(Y_S - vt)^2 + (X_G - X_S)^2}}\right]$$

The gear ratio is $G_E = \dfrac{r\omega(t)}{v}$

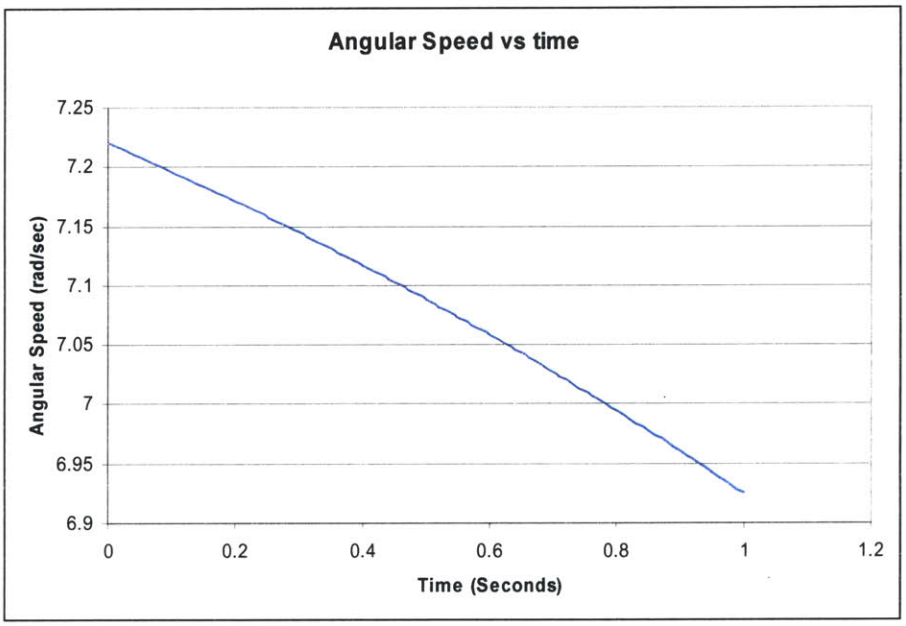


Figure 5.5.4.1: The calculated angular speed vs time for the DRIVE axis for Move 6.

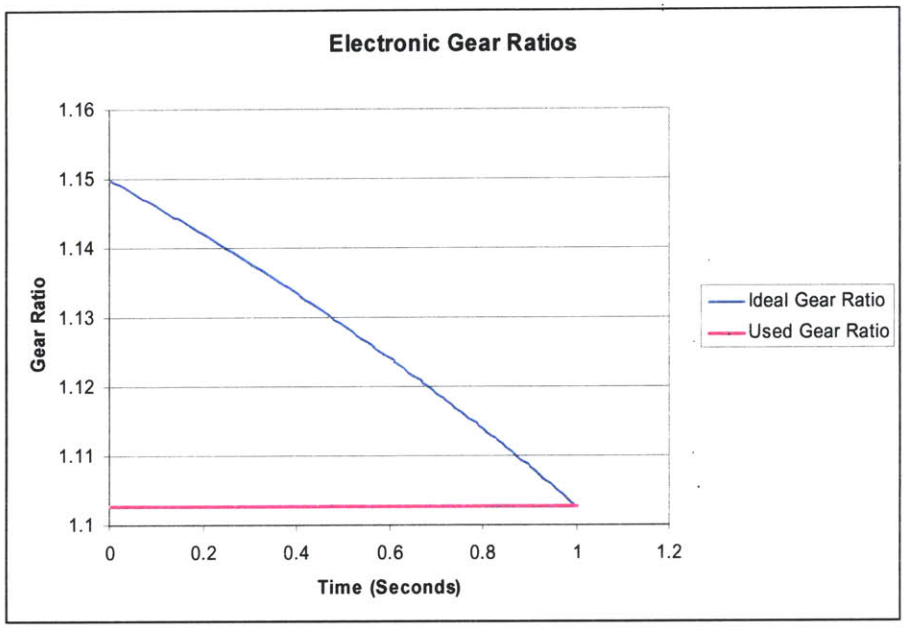


Figure 5.5.4.2: The computed and used electronic gear ratios. Note that a liberal gearing ratio was used so that the substrate is not overstretched. A compensation move will absorb all the excess substrate released.

(G) Move 7

Description

Move 7 is an angular compensation move because of the liberal gear ratio employed by Move 6. The angular compensation is easily calculated as:

$$\Delta\theta = \int \omega_{IDEAL}(t)dt - \int \omega_{USED}(t)dt$$

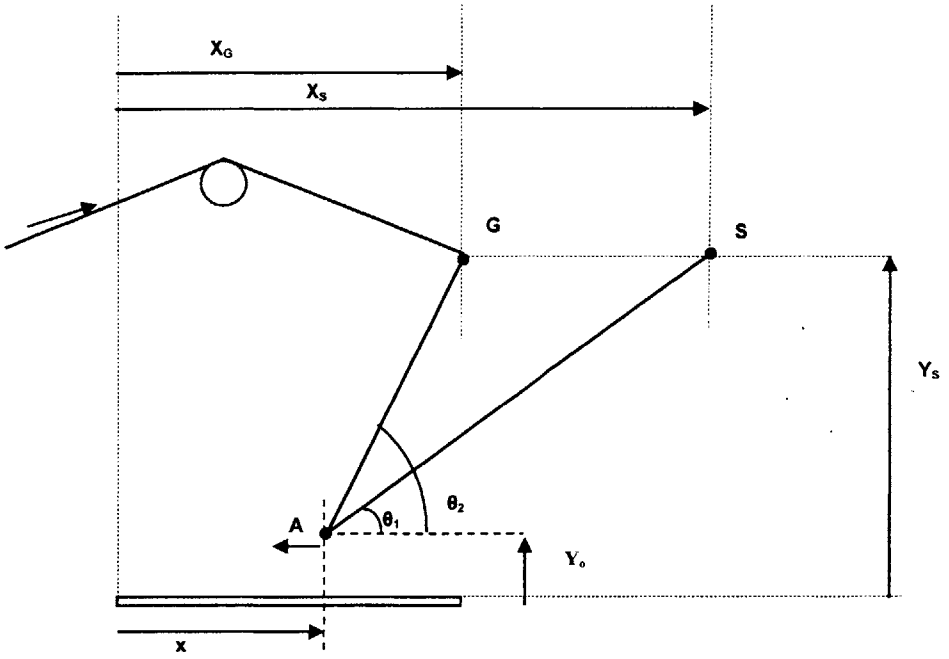
where ω_{USED} and ω_{IDEAL} correspond to the actually used and the ideal gear ratio respectively for Move 6.

(H) Move 8

Description

The two wipers, i.e the axes Y_XY1, Y_XY2, X_XY1 and X_XY2 come back to their original position. The axis DRIVE is electronically geared to X_XY1

.



Kinematic Analysis

Again, $-\frac{dGA}{dt} + r\omega(t) = \frac{dSA}{dt}$ ..... (1)

From Geometry,

$$Sin\theta_1 = \frac{(Y_S - Y_o)}{GA} \quad \& \quad Sin\theta_2 = \frac{(Y_S - Y_o)}{SA}$$

$$\Rightarrow \frac{dGA}{dt} = -(Y_S - Y_o)\frac{Cos\theta_1}{Sin^2\theta_1}\frac{d\theta_1}{dt} \quad \& \quad \frac{dSA}{dt} = -(Y_S - Y_o)\frac{Cos\theta_2}{Sin^2\theta_2}\frac{d\theta_2}{dt} \quad \ldots\ldots\ldots \quad (2)$$

(2)

Also, $\tan\theta_1 = \frac{(Y_S - Y_O)}{vt}$ & $\tan\theta_2 = \frac{(Y_S - Y_O)}{(X_S - (X_G - vt))}$

$$\Rightarrow \frac{d\theta_1}{dt} = -v(Y_S - Y_O)\frac{Cos^2\theta_1}{(X_G - vt)^2} \quad \& \Rightarrow \frac{d\theta_2}{dt} = -v(Y_S - Y_O)\frac{Cos^2\theta_2}{(X_S - (X_G - vt))^2} \quad \ldots\ldots$$

(3)

Using (1), (2) and (3),

$$\frac{dGA}{dt} = -v\frac{(X_G - vt)}{\sqrt{(Y_S - Y_O)^2 + (X_G - vt)^2}} \quad \& \quad \frac{dSA}{dt} = -v\frac{(X_S - vt)}{\sqrt{(Y_S - Y_O)^2 + (X_S - vt)^2}}$$

$$\Rightarrow \omega(t) = \frac{v}{R}\left[\frac{(X_G - vt)}{\sqrt{(Y_S - Y_O)^2 + (X_G - vt)^2}} + \frac{(X_S - vt)}{\sqrt{(Y_S - Y_O)^2 + (X_S - vt)^2}}\right]$$

The gear ratio is $G_E = \frac{r\omega(t)}{v}$

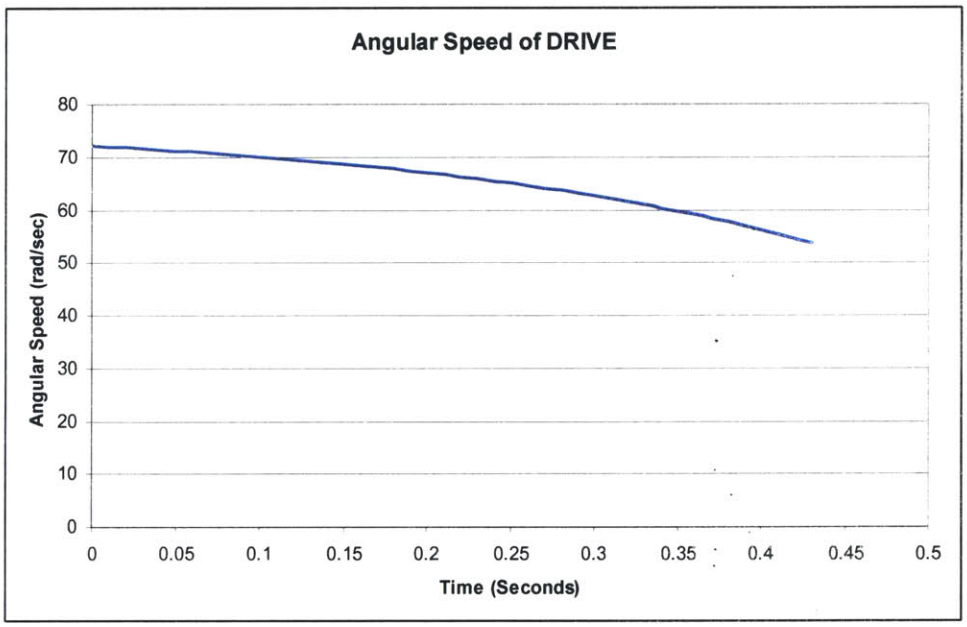


Figure 5.5.5.1: The calculated angular speed vs time for the DRIVE axis for Move 8.

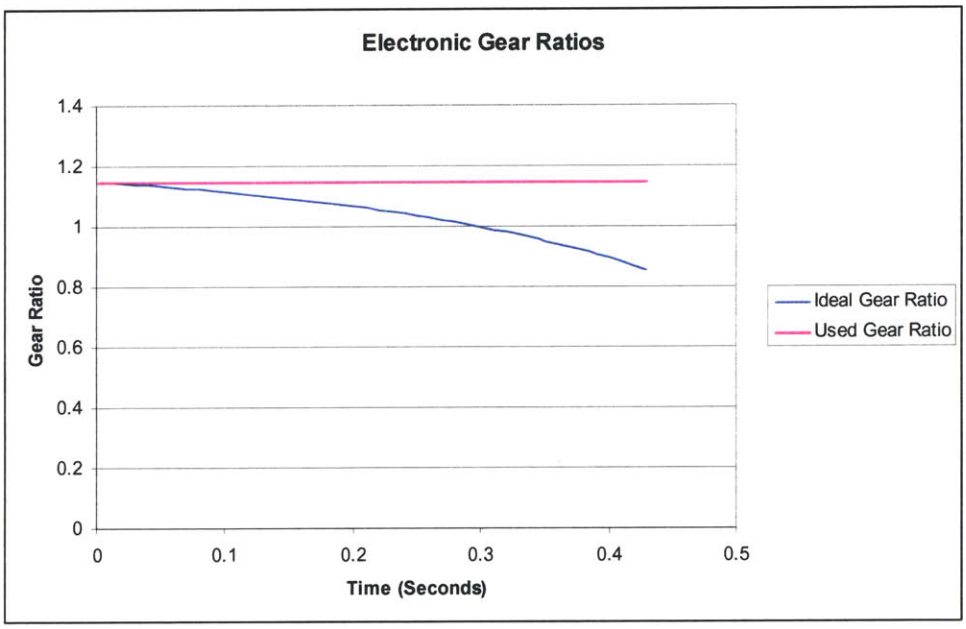


Figure 5.5.5.2: The computed and used electronic gear ratios. Note that a liberal gearing ratio was used so that the substrate is not overstretched. A compensation move will absorb all the excess substrate released.

(I) Move 9

Description

Move 9 is an angular compensation move because of the liberal gear ratio employed by Move 8. The angular compensation is easily calculated as:

$$\Delta\theta = \int \omega_{USED}(t)dt - \int \omega_{IDEAL}(t)dt$$

where ω_{USED} and ω_{IDEAL} correspond to the actually used and the ideal gear ratio respectively for Move 8.

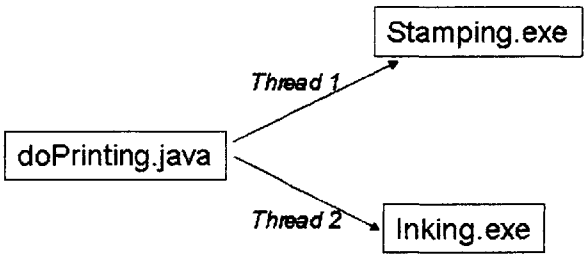
(J) Move 10

Description

This is an indexing move to bring out the fresh substrate from the supply spool. The drive and supply spool are electronically geared in the ratio 1:1.

5.5 Execution of the Application

The inking and stamping processes were controlled by two separate C/C++ programs called `Stamping.cpp` and `Inking.cpp`. Both the programs were pre-compiled and the executable files were started as parallel threads by a simple java program. The following is the pseudo code for Stamping.exe.



5.6 Conclusions

In this chapter, we discussed the mechanical and electronic elements of automation for the prototype and the software integration to accomplish the desired motions. The use of high quality stepper motors working under low loads compensated sufficiently for the lack of closed loop control to result in highly accurate and repeatable motions. The software development was helped partially by the well documented function libraries supplied with the microcontroller's vendor. The powerful libraries allowed for sophisticated functions like electronic gearing and the appropriate gearing ratios were determined using kinematic analysis.

Chapter 6

Conclusions and Future Work

A working prototype demonstrating the integration of inking and stamping for micro contact printing was developed. The design of the prototype was based on a manufacturing system which could achieve high volume production without sacrificing quality. The prototype demonstrates that high quality, high flexibility, high production rate and low cost manufacturing using micro contact printing is indeed possible. The literature review in the process of the design suggests that advancements in chemistry will further improve the case for taking this process towards commercialization.

Limitation of time meant that a closed loop approach to automation could not be tried. It is recommended that in the future versions of the prototype, at least simple enhancements like addition of limit switches and optical encoders for at least the rotary table is considered. It is also recommended to replace the wipers' motion actuators by a simpler mechanism to reduce the complexity of motion.

The motions in the prototype were tested for a large range of motions and it was clear that the production rate will not depend on the mechanical motions. The production rate will be dictated by the chemistries of the materials used and therefore further research is recommended. As a suggestion, designed experiments could be conducted on either this prototype or a simple manual device to optimize the processing times for stamping and inking by identifying the optimal process parameters. It is expected that not only will this

lead to better knowledge about the process but may also play an important role in creating new chemistries for better outcomes.

List of References

[1] Zhao, X.-M., et al. 1996. Fabrication of single-mode polymeric waveguides using micromolding in capillaries. *Adv. Mater.* 8: 420-24.

[2] Kumar, A.; Whitesides, G.M.,"Features of Gold having Micrometer to Centimeter Dimensions Can Be Formed Through a Combination of Stamping with an Elastomeric Stamp and an Alkanethiol 'Ink' Followed by Chemical Etching," *App. Phys. Lett.,* 1993, Vol. 63(14), pp.2002-2004

[3] Xia, Y.; Jabez, J.; McClelland; Gupta, R.;Qin, D.; Zhao, X-M. et al., "Replica molding using polymeric materials: A practical step toward nano-manufacturing," *Adv. Mater*. 1997, Vol.9(2), pp.147-149

[4] Xia, Y.; Whitesides, G.M.,"Soft Lithography," *Annu. Rev. Mater. Sci.*1998, Vol. 28,pp.153-184

[5] Kim, E.; Xia, Y.; Whitesides, G.M ., "Polymer microstructures formed by moulding in capillaries," *Nature *, 1995, Vol.376, pp.581-584

[6] Zhao, X.-M.; Xia, Y.; Whitesides, G.M., "Fabrication of three-dimensional micro-structures: Microtransfer molding,", *Adv. Mater.*, 1996, Vol.8(10), pp.837-840

[7] Kim, E.; Xia, Y.; Zhao, X.-M.; Whitesides, G.M.,"Solvent-assisted microcontact molding: A convenient method for fabricating three-dimensional structures on surfaces of polymers," *Adv. Mater. *, 1997, Vol.9 (8), pp. 651-65

[8] Love, J. C.; Estroff, L. A.; Kriebel, J. K.; Nuzzo, R. G.; Whitesides, G. M.,"Self-Assembled Monolayers of Thiolates on Metals as a Form of Nanotechnology," *Chem. Rev.,2005 Vol.**105* (4),pp.1103-1170

[9] Kumar, A.; Abbott, N.L.; Kim, E.; H. Biebuyck; Whitesides G.M., "Patterned Self-Assembled Monolayers and Meso-Scale Phenomena," *Acc. Chem. Res.,*1995, Vol.28, pp.219

[10] Li, X.-M.; Huskens, J.; Reinhoudt, D.N. "Reactive self-assembled monolayers on flat and nanoparticle surfaces, and their application in soft and scanning probe lithographic nanofabrication technologies," *J. Mater. Chem*., 2004, Vol.14, pp.2954-2971

[11] Xia, Y.;Kim, E.; Whitesides, G.M., "Microcontact Printing of Alkanethiols on Silver and Its Application in Microfabrication," *J. Electrochem. Soc., *1996, Vol.143(3), pp.1070-1079

[12] Xia, Y; Kim, E.; Mrksich, M.; Whitesides, G.M. "Microcontact Printing of Alkanethiols on Copper and Its Application in Microfabrication," *Chem. Mater.* 1996.Vol.8(3), pp.601-603

[13] Clarson, S. J.;Semlyen, J.A. Siloxane polymers; Prentice Hall:Englewood Cliffs, NJ, 1993.

[14] J. A. Greenwood and J. B. P. Williamson, *Proc. R. Soc. London, Ser. A, 1996* Vol.295, pp.300

[15] Rogers, J.A.;Nuzzo, R.G.,"Recent Progress in soft lithography," *Materialstoday, *2005 pp.50

[16] Xia, Y.; Qin, D.; Whitesides, G.M., "Microcontact Printing with a Cylindrical Rolling Stamp: A Practical Step Toward Automatic Manufacturing of Patterns with Submicrometer Sized Features," *Adv. mater.*, 1996, Vol.8(12), pp.1015-1017

[17] M.M.J. Decre, Schneider, R.; Burdinski, D; Schellekens, J.; Saalmink, M.; Dona, R.,"Wave Printing(I): Towards Large-Area, Multilayer Microcontact Printing," *Mat. Res. Soc. Symp. Proc.* 2004, Vol. EXS-2

[18] Delamarche, E.; Geissler, M.; Vichiconti, J.; Graham, W. S.; Andry, P. A.; Flake, J. C. et al., "Electroless Deposition of NiB on 15 Inch Glass Substrates for the Fabrication of Transistor Gates for Liquid Crystal Displays," ***Langmuir,2003,*** Vol.19 (14), pp.5923-5935

[19] Rogers, J.A.,"Toward Paperlike Displays," ***Science,*** 2001, Vol.291, pp.1502-1503

[20] Guo, Q; Teng, X.; Yang, H.,"Overpressure Contact Printing," *Nano Letters,* 2004, Vol.4(9), pp.1657-1662 **

*[21] *Odom, T. W.; Love, J. C.; Wolfe, D.B.; Paul, K.E.; Whitesides, G.M.,"Improved Pattern Transfer in Soft Lithography Using Composite Stamps," *Langmuir, *2002, Vol.18(13), pp.5314-5420

[22] Xia, Y.; Zhao, X-M.; Whitesides, G.M. *Microelectron. **Eng.,* 1996, Vol.32, pp.255.

[23] Delamarche, E.; Schmid, H.; Bietsch, A.; Larsen, N. B.; Rothuizen, H.; Michel, B.; Biebuyck, H.,"Transport Mechanisms of Alkanethiols during Microcontact Printing on

Gold," *J. Phys. Chem. B.,*1998, Vol. 102(18), pp.3324-3334

[24] Libioulle, L.; Bietsch, A.; Schmid, H.; Michel, B.; Delamarche, E. ,"Contact-Inking Stamps for Microcontact Printing of Alkanethiols on Gold," *Langmuir, *1999, Vol.15(2) pp.300-304

[25] Guo, Q; Teng, X.; Yang, H.,"Overpressure Contact Printing," *Nano Letters,* 2004, Vol.4(9), pp.1657-1662 .

[26] Delamarche, E; Schmid, H.; Michel, B; Biebuyck, H. "Stability of molded polydimethylsiloxane microstructures," *Adv. Mater*., 1997, Vol.9(9), pp.741-746

[27] Hui, C. Y.; Jagota, A; Lin, Y. Y.; Kramer, E.J.,"Constraints on Microcontact Printing Imposed by Stamp Deformation *Langmuir*," 2002, Vol.18, pp.1394-1407

[28] B. Michel, A.; Bernard, A.; Bietsch, E.; Delamarche, M.; Geissler, D.; Juncker, H. et al., "Printing Meets Lithography," *IBM J. RES. & DEV.*, 2001, Vol 45(5), pp.5697-719

[29] Masuda, T.;Isobe, E.;Higashimura, T.;Tagada., K.," Poly[1-(trimethylsilyl)-1-propyne]: a new high polymer synthesized with transition-metal catalysts and characterized by extremely high gas permeability," *J. Am. Chem. Soc.,* 1983, Vol.105, pp.7473

[30] A.M. Kendale, Automation of soft lithographic microcontact printing. S.M. Thesis, Massachusetts Institute of Technology, 2001

[31] Rogers, J.A.; Bao, Z.; Baldwin, K.;Dodabalapur, A.; Crone,B.; Raju, V.R. et al., "Paper-like electronic displays: Large-area rubber-stamped plastic sheets of electronics and microencapsulated electrophoretic inks," 2001, vol. 98(9), pp. 4835-4840

[32] National Instruments product manual for PCI 7350. ww.ni.com/pdf/manuals/371060a.pdf.

[33] M.Eng Thesis, Massachusetts Institute of Technology, "Development of a prototype for Micro Contact Printing" Hyung Jun Kim.

APPENDIX 1

Source code for the stamping software (Stamping.cpp)

```
#include "stdafx.h"
#include "flexmotn.h"
#include "stdio.h"
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <math.h>

//Addresses for the axes
#define AXES_246 0x54
#define AXES_16  0x42
#define AXES_35  0x8
#define AXES_245 0x14
#define AXES_135  0xA

#define M 10          //microstepping index used.
# define NUM_CYCLES 2 //The number of cycles

//Fixed parameters representing the structure of the machine
# define Y_S 9.5
# define X_S 20
# define X_G 13

//definitions for the motions, velocities, accelerations and jerks

//MOVE1: Both the wiper touch down on the stamping stage
#define DIS_Y_XY1_MOVE1  -2.0
#define DIS_Y_XY2_MOVE1  -2.0
#define VEL_Y_XY1_MOVE1   2.0
#define VEL_Y_XY2_MOVE1   2.0
#define ACLN_Y_XY1_MOVE1  2.0
#define ACLN_Y_XY2_MOVE1  2.0
#define JERK_Y_XY1_MOVE1  2.0
#define JERK_Y_XY2_MOVE1  2.0

//MOVE2:The wiper corresponding to X_XY2 moves towards the supply spool,
applying the substrate
#define DIS_X_XY2_MOVE2  8.2
#define VEL_X_XY2_MOVE2   12.0
#define ACLN_X_XY2_MOVE2  2.0
#define JERK_X_XY2_MOVE2  2.0

#define STAMPING_SLEEP 1000 //Stamping time in milliseconds

//Move4:The wiper corresponding to X_XY1 moves towards the supply spool,
peeling off the substrate
```

```
// note: may need to change gearing ratio in "DoMove4" if you
drastically change spool diameter of substrate
#define DIS_X_XY1_MOVE4  -8.2
#define VEL_X_XY1_MOVE4   6.0
#define ACLN_X_XY1_MOVE4  2.0
#define JERK_X_XY1_MOVE4  2.0

//Move6: Both the wipers lift off the stamping stage.

#define DIS_Y_XY1_MOVE6   2.0
#define DIS_Y_XY2_MOVE6   2.0
#define VEL_Y_XY1_MOVE6   2.0
#define VEL_Y_XY2_MOVE6   2.0
#define ACLN_Y_XY1_MOVE6  2.0
#define ACLN_Y_XY2_MOVE6  2.0
#define JERK_Y_XY1_MOVE6  2.0
#define JERK_Y_XY2_MOVE6  2.0

//Move 8: Both wipers return to their original positions. End of cycle.
#define DIS_X_XY1_MOVE8   8.2
#define DIS_X_XY2_MOVE8  -8.2
#define VEL_X_XY1_MOVE8   12.0
#define VEL_X_XY2_MOVE8   12.0
#define ACLN_X_XY1_MOVE8  2.0
#define ACLN_X_XY2_MOVE8  2.0
#define JERK_X_XY1_MOVE8  2.0
#define JERK_X_XY2_MOVE8  2.0

/////Prototyes of functions used////
void ErrorHandler(u16 errorCode, u16 commandID, u16 resourceID);
i32 loadTrajectory(u8& bID,u8& axis,u16 jerk, u32 acln, i32 dis,i32
vel);
i32 configureGearing(u8& bID, u8& Master, u8& Slave, i16  numerator,
u16 denominator);
i32 checkModalErrors(u8& bID);
i32 doPreHoming(u8& bID);
i32 doMove1(u8& bID);
i32 doMove2(u8& bID);
i32 doMove3(u8& bID);
i32 doMove4(u8& bID);
i32 doMove5(u8& bID);
i32 doMove6(u8& bID);
i32 doMove7(u8& bID);
i32 doMove8(u8& bID);
i32 doMove9(u8& bID);
i32 doEndHoming(u8 &biD);
//End Prototyes of functions used////

// Main routine
void __cdecl main(void){

i32 status=0;//The return status of the function (for non-modal errors)
u8 boardID=1;            //The board identification parameter
u16 csr=0;
int i;
```

```
      //Check if the board is at power up reset condition
      status = flex_read_csr_rtn(boardID, &csr);

      if(status == NIMC_noError){
            if (csr & NIMC_POWER_UP_RESET ){
                  printf("\nThe FlexMotion board is in the reset
condition. Please initialize the board before running this example.
The Initialize.c example shows you how to do this.\n");
                  exit(0);
            }
      }
if (status == NIMC_noError){
         status=doPreHoming(boardID);
        if (status)ErrorHandler((u16)status, 0, 0);
   }

for(i=0;i<NUM_CYCLES;i++){ //this is the for loop for stamping
if (status == NIMC_noError){

//Do the sequential moves
status = doMove1(boardID);
       if (status)ErrorHandler((u16)status, 0, 0);
status = doMove2(boardID);
Sleep(STAMPING_SLEEP);     //Stamping time
       if (status)ErrorHandler((u16)status, 0, 0);
status = doMove3(boardID);//Angular Compensation.SUPPLY absorbs by
rotating clockwise
       if (status)ErrorHandler((u16)status, 0, 0);
status = doMove4(boardID);//move X_XY1 to right, DRIVE is
electronically geared to X_XY1
       if (status)ErrorHandler((u16)status, 0, 0);
status = doMove5(boardID);//Angular Compensation.DRIVE absorbs by
rotating anticlockwise
       if (status)ErrorHandler((u16)status, 0, 0);
status = doMove6(boardID);
       if (status)ErrorHandler((u16)status, 0, 0);
status = doMove7(boardID);
       if (status)ErrorHandler((u16)status, 0, 0);
status = doMove8(boardID);
       if (status) ErrorHandler((u16)status, 0, 0);
status = doMove9(boardID);
       if (status)ErrorHandler((u16)status, 0, 0);
      }
   }//end for loop

      if (status == NIMC_noError){
         status=doEndHoming(boardID);
        if (status)ErrorHandler((u16)status, 0, 0);
   }

      printf("\n\nFinished.\n");

      exit(0);

}//end main
```

```
i32 doMove1(u8& bID) {
    u8 boardID=bID;
      i32 status=0;                  //return status
      u8 Y_XY2 = NIMC_AXIS2;  // The y axis of XY-positioner2
      u8 Y_XY1 = NIMC_AXIS4;  // The y axis of XY-positioner2.
      u8 SUPPLY = NIMC_AXIS6; // The supply spool electronically geared
to Y_XY2
      u16 axisStatusOf_Y_XY2, axisStatusOf_Y_XY1; //Status of the axes
      u8 inputVector = 0xFF;  //input vector

    i16 numerator;  //-3;
      u16 denominator;//2;      //for the gearing ratio.
    u16 csr=0;
      //Variables for modal error handling
      u16 commandID;                 //The commandID of the function
      u16 resourceID;                //The resource ID
      u16 errorCode;                 //The error generated

      //Load trajectory parameters for Y_XY2 and Y_XY1
      if(status == NIMC_noError){
      status=loadTrajectory( boardID, Y_XY2,JERK_Y_XY2_MOVE1,
ACLN_Y_XY2_MOVE1, DIS_Y_XY2_MOVE1, VEL_Y_XY2_MOVE1);
      }
      if(status == NIMC_noError){
      status=loadTrajectory( boardID, Y_XY1,JERK_Y_XY1_MOVE1,
ACLN_Y_XY1_MOVE1, DIS_Y_XY1_MOVE1, VEL_Y_XY1_MOVE1);
      }

      //Calculate the electronic gearing ratio
    //numerator=(Xs*(Ys-abs(DIS_Y_XY2_MOVE1)/2)/sqrt(pow((Ys-
abs(DIS_Y_XY2_MOVE1)/2),2)+pow(Xs,2))-Xg*(Ys-
abs(DIS_Y_XY2_MOVE1)/2)/sqrt(pow((Ys-
abs(DIS_Y_XY2_MOVE1)/2),2)+pow(Xg,2)))*100;

      numerator=83;//YOU WILL HAVE TO MAKE THIS -83 IN CASE THE SUPPLY
MOTOR STARTS TO TURN CLOCKWISE FOR POSITIVE SIGNAL
      denominator=100;

      //End calculating the electronic gearing ratio

     //ELECTRONIC GEARING: SUPPLY IS SLAVE TO Y_XY2
      if(status == NIMC_noError){
       status=configureGearing(boardID, Y_XY2,SUPPLY,
numerator,denominator);
      }
      //Start the motions
      if(status == NIMC_noError){status = flex_start(boardID,
NIMC_AXIS_CTRL, AXES_246);}
      printf("\n Running the Move1 now\n The two wipers are touching
the stamping stage\n");
      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, Y_XY1,
&axisStatusOf_Y_XY1);
            if(status) break;
```

```
                    status = flex_read_axis_status_rtn(boardID, Y_XY2,
&axisStatusOf_Y_XY2);
            if(status) break;

      }while ( ! ( (axisStatusOf_Y_XY1 &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) &&
                    (axisStatusOf_Y_XY2 & (NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) )); //Test against the move complete bit or the
axis off bit
      }

      //Bring all axes to halt
      if(status == NIMC_noError){ status = flex_stop_motion(boardID,
NIMC_AXIS_CTRL, NIMC_HALT_STOP, AXES_246);      }

      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
    }

        if(status == NIMC_noError){
                status = flex_enable_gearing(boardID, 0); //disable the
gearing----IS IT REALLY NEEDED???
        }
        if(status){ErrorHandler((u16)status, 0, 0);}

            return(status);

}

i32 doMove2(u8& bID) {
      u8 boardID=bID;
      i32 status=0;                                  //return status
      u8 X_XY2 = NIMC_AXIS1;  //i.e, The x axis of XY-positioner2
      u8 SUPPLY = NIMC_AXIS6; //i.e, The SUPPLY spool
      u16 axisStatusOf_X_XY2;
      u8 inputVector = 0xFF;
    i16 numerator=-121;//-23;
      u16 denominator=100;//20;
      u16 commandID;
      u16 resourceID;
      u16 errorCode;
      u16 csr=0;

      //Load trajectory parameters for X_XY2, i.e the X motor of the
XY-positioner2
      if(status == NIMC_noError){
            status=loadTrajectory( boardID, X_XY2,JERK_X_XY2_MOVE2,
ACLN_X_XY2_MOVE2, DIS_X_XY2_MOVE2, VEL_X_XY2_MOVE2);
      }
    //Electronically gear the SUPPLY axis to X_XY2
      if(status == NIMC_noError){
       status=configureGearing(boardID, X_XY2,SUPPLY,
numerator,denominator);
      }
      //Start the motions
      status = flex_start(boardID, NIMC_AXIS_CTRL, AXES_16);
```

```
      printf("\n Running the Move2 now\nWiper of X_XY2");
      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, X_XY2,
&axisStatusOf_X_XY2);
            if(status) break;
            }while ( ! ( (axisStatusOf_X_XY2 &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT ))));
        }
      //Bring all axes to halt
      if(status == NIMC_noError){
     status = flex_stop_motion(boardID, NIMC_AXIS_CTRL, NIMC_HALT_STOP,
AXES_16);
      }

      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
    }
        if(status == NIMC_noError){
           status = flex_enable_gearing(boardID, 0);
        }
        if(status){ErrorHandler((u16)status, 0, 0);}
            return(status);

}
i32 doMove3(u8& bID){
      u8 boardID=bID;
      i32 status=0;
      printf("\n Running the Move3 now\n");
      //calculate the angualr compensation..and move the supply
clockwise
      return(status);
}
i32 doMove4(u8& bID) {
    u8 boardID=bID;
      i32 status=0;                    //return status
      u8 X_XY1 = NIMC_AXIS3;  //i.e, The X axis of XYPositioner 1
      u8 DRIVE = NIMC_AXIS5;  //i.e, The Drive spool
      u16 axisStatusOf_X_XY1; //axis status of axis2,i.e, the master
axis
      u8 inputVector = 0xFF;  //input vector
    i16 numerator=95;  // this corresponds to the drive spool
      u16 denominator=100; //for the gearing ratio. This corresponds to
the wiper
      u16 commandID;
      u16 resourceID;
      u16 errorCode;
      u16 csr=0;

      //Load trajectory parameters for X_XY1
    if(status == NIMC_noError){
      status=loadTrajectory( boardID, X_XY1,JERK_X_XY1_MOVE4,
ACLN_X_XY1_MOVE4, DIS_X_XY1_MOVE4, VEL_X_XY1_MOVE4);
      }
```

```
      //ELECTRONIC GEARING--AXIS5 IS SLAVE TO AXIS3
    //if(status == NIMC_noError){
      //status=configureGearing(boardID, X_XY1,DRIVE,
numerator,denominator);
      //}
      if(status == NIMC_noError){status =
flex_config_gear_master(boardID, DRIVE, X_XY1); }
      if(status == NIMC_noError){status = flex_load_gear_ratio(boardID,
DRIVE, NIMC_RELATIVE_GEARING,numerator, denominator, 0xFF);}
      if(status == NIMC_noError){status = flex_stop_motion(boardID,
DRIVE, NIMC_HALT_STOP, 0);}//need to energize befor enabling gearing
      if(status == NIMC_noError){status = flex_enable_gearing(boardID,
(u16)(1 << DRIVE) ); }
      if(status ==
NIMC_noError){status=flex_configure_stepper_output(boardID,DRIVE,NIMC_S
TEP_AND_DIRECTION,NIMC_ACTIVE_LOW,NIMC_OPEN_COLLECTOR);}



    //Start the motions simultaneously
      status = flex_start(boardID, NIMC_AXIS_CTRL, AXES_35);
            printf("\n Running the Move4 now\n");

      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, X_XY1,
&axisStatusOf_X_XY1);
            if(status) break;

            }while ( ! ( (axisStatusOf_X_XY1 &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )))); //Test against the move complete bit or the
axis off bit
        }
      //Bring all axes to halt
      if(status == NIMC_noError){ status = flex_stop_motion(boardID,
NIMC_AXIS_CTRL, NIMC_HALT_STOP, AXES_35);       }

      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
    }
        if(status == NIMC_noError){
                status = flex_enable_gearing(boardID, 0);
        }
        if(status){ErrorHandler((u16)status, 0, 0);}
            return(status);

}
i32 doMove5(u8& bID){
    u8 boardID=bID;
      i32 status=0;
            printf("\n Running the Move5 now\n");
      //calculate the angualr compensation..and move the supply
clockwise
```

```
      return(status);
}
i32 doMove6(u8& bID) {
    u8 boardID=bID;
      i32 status=0;                                  //return status
      u8 Y_XY2 = NIMC_AXIS2;
      u8 Y_XY1 = NIMC_AXIS4;
      u8 DRIVE = NIMC_AXIS5;
      u16 axisStatusOf_Y_XY1,axisStatusOf_Y_XY2;
      u8 inputVector = 0xFF;  //input vector
    i16 numerator=-111;//-1;
      u16 denominator=100;//1; //for the gearing ratio.
      //Variables for modal error handling
      u16 commandID;                                 //The commandID of the
function
      u16 resourceID;                          //The resource ID
      u16 errorCode;                                 //The error generated
      u16 csr=0;

      //Load trajectory parameters for Y_XY2, i.e the y motor of the
XY-positioner2
    if(status == NIMC_noError){
      status=loadTrajectory( boardID, Y_XY2,JERK_Y_XY2_MOVE6,
ACLN_Y_XY2_MOVE6, DIS_Y_XY2_MOVE6, VEL_Y_XY2_MOVE6);
      }
    if(status == NIMC_noError){
      status=loadTrajectory( boardID, Y_XY1,JERK_Y_XY1_MOVE6,
ACLN_Y_XY1_MOVE6, DIS_Y_XY1_MOVE6, VEL_Y_XY1_MOVE6);
      }

    //ELECTRONIC GEARING--DRIVE TO Y_XY2
    if(status == NIMC_noError){
      status=configureGearing(boardID, Y_XY2,DRIVE,
numerator,denominator);
      }
      //Start the motions simultaneously
      status = flex_start(boardID, NIMC_AXIS_CTRL, AXES_245);
            printf("\n Running the Move6 now\n");

      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, Y_XY1,
&axisStatusOf_Y_XY1);
            if(status) break;
                  status = flex_read_axis_status_rtn(boardID, Y_XY2,
&axisStatusOf_Y_XY2);
            if(status) break;

      }while ( ! ( (axisStatusOf_Y_XY1 &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) &&
                  (axisStatusOf_Y_XY2 & (NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) )); //Test against the move complete bit or the
axis off bit
      }
```

```
      //DE-ENERGIZE ALL THE AXIS TO SAVE POWER
      if(status == NIMC_noError){ status = flex_stop_motion(boardID,
NIMC_AXIS_CTRL, NIMC_KILL_STOP, AXES_245);      }

      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
    }
        if(status == NIMC_noError){
                status = flex_enable_gearing(boardID, 0); //disable the
gearing----IS IT REALLY NEEDED???
        }
        if(status){ErrorHandler((u16)status, 0, 0);}

            return(status);

}
i32 doMove7(u8& bID){
    u8 boardID=bID;
      i32 status=0;
            printf("\n Running the Move7 now\n");
      //calculate the angualr compensation..and move the supply
clockwise
      return(status);
}
i32 doMove8(u8& bID) {
    u8 boardID=bID;
      i32 status=0;
      u8 X_XY2 = NIMC_AXIS1;
      u8 X_XY1 = NIMC_AXIS3;
      u8 DRIVE = NIMC_AXIS5;
      u16 axisStatusOf_X_XY2,axisStatusOf_X_XY1;
      u8 inputVector = 0xFF;  //input vector
    i16 numerator=-80;//-1;
      u16 denominator=100;//1; //for the gearing ratio.
      //Variables for modal error handling
      u16 commandID;
      u16 resourceID;
      u16 errorCode;
      u16 csr=0;

      //Load trajectory parameters
    if(status == NIMC_noError){
      status=loadTrajectory( boardID, X_XY2,JERK_X_XY2_MOVE8,
ACLN_X_XY2_MOVE8, DIS_X_XY2_MOVE8, VEL_X_XY2_MOVE8);
      }
    if(status == NIMC_noError){
      status=loadTrajectory( boardID, X_XY1,JERK_X_XY1_MOVE8,
ACLN_X_XY1_MOVE8, DIS_X_XY1_MOVE8, VEL_X_XY1_MOVE8);
      }
      // ELECTRONIC GEARING--DRIVE is geared to X_XY1
  //  if(status == NIMC_noError){
//    status=configureGearing(boardID, X_XY1,DRIVE,
numerator,denominator);
//    }

      //Start the motions simultaneously
```

```
      status = flex_start(boardID, NIMC_AXIS_CTRL, AXES_135);
            printf("\n Running the Move8 now\n");

      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, X_XY1,
&axisStatusOf_X_XY1);
            if(status) break;
                  status = flex_read_axis_status_rtn(boardID, X_XY2,
&axisStatusOf_X_XY2);
            if(status) break;

      }while ( ! ( (axisStatusOf_X_XY2 &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) &&
                  (axisStatusOf_X_XY1 & (NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) )); //Test against the move complete bit or the
axis off bit
      }

      //DE-ENERGIZE ALL THE AXIS TO SAVE POWER
      if(status == NIMC_noError){ status = flex_stop_motion(boardID,
NIMC_AXIS_CTRL, NIMC_KILL_STOP, AXES_135);      }

      //Checking modal errors here
   if(status == NIMC_noError){
      status=checkModalErrors(boardID);
    }

        if(status == NIMC_noError){
                status = flex_enable_gearing(boardID, 0);
        }
        if(status){ErrorHandler((u16)status, 0, 0);}

            return(status);

}
i32 doMove9(u8& bID){
    u8 boardID=bID;
      i32 status=0;
            printf("\n Running the Move9 now\n");
      //calculate the angualr compensation..and move the supply
clockwise

      if(status == NIMC_noError){
      status=checkModalErrors(boardID);
    }
      return(status);
}


void ErrorHandler(u16 errorCode, u16 commandID, u16 resourceID){

      i8 *errorDescription;         //Pointer to i8's -  to get error
description
      u32 sizeOfArray;                    //Size of error description
```

```
      u16 descriptionType;                //The type of description to
be printed
      i32 status=0;                                    //Error returned
by function

      if(commandID == 0){
            descriptionType = NIMC_ERROR_ONLY;
      }else{
            descriptionType = NIMC_COMBINED_DESCRIPTION;
      }

      //First get the size for the error description
      sizeOfArray = 0;
      errorDescription = NULL;//Setting this to NULL returns the size
required
      status = flex_get_error_description(descriptionType, errorCode,
commandID, resourceID,

      errorDescription, &sizeOfArray );

      //Allocate memory on the heap for the description
      errorDescription = (i8*)malloc(sizeOfArray + 1);

      sizeOfArray++; //So that the sizeOfArray is size of description +
NULL character
      // Get Error Description
      status = flex_get_error_description(descriptionType, errorCode,
commandID, resourceID,

      errorDescription, &sizeOfArray );

      if (errorDescription != NULL){
            printf("\n");
            printf(errorDescription);//Print description to screen
            free(errorDescription);//Free allocated memory
      }else{
            printf("Memory Allocation Error");
      }
}

i32 loadTrajectory(u8& bID,u8& axis,u16 jerk, u32 acln, i32 dis,i32
vel)
{
      u8 boardID=bID;
      i32 status=0;
      u8 inputVector = 0xFF;

      if(status == NIMC_noError){
      status = flex_load_scurve_time(boardID, axis, (u16)jerk,
inputVector);
      }
      if(status == NIMC_noError){
      status = flex_load_acceleration(boardID, axis, NIMC_BOTH,
(u32)(acln*M*99.9403097), inputVector);
      }
      if(status == NIMC_noError){
```

```
      status = flex_load_velocity(boardID, axis,
(i32)(vel*M*99.9403097), inputVector);
      }
      if(status == NIMC_noError){
      status = flex_load_target_pos(boardID, axis,
(i32)(dis*M*99.9403097), inputVector);
      }
      if(status == NIMC_noError){
      status = flex_set_op_mode(boardID, axis, NIMC_RELATIVE_POSITION);
      }
      if(status == NIMC_noError){
      status=flex_configure_stepper_output(boardID,axis,NIMC_STEP_AND_D
IRECTION,NIMC_ACTIVE_LOW,NIMC_OPEN_COLLECTOR);
      }
      return(status);
}
i32 configureGearing(u8& bID, u8& Master, u8& Slave, i16  numerator,
u16 denominator)
{
      u8 boardID=bID;
      i32 status=0;

      if(status == NIMC_noError){status =
flex_config_gear_master(boardID, Slave, Master); }
      if(status == NIMC_noError){status = flex_load_gear_ratio(boardID,
Slave, NIMC_RELATIVE_GEARING,numerator, denominator, 0xFF);}
      if(status == NIMC_noError){status = flex_stop_motion(boardID,
Slave, NIMC_HALT_STOP, 0);}//need to energize befor enabling gearing
      if(status == NIMC_noError){status = flex_enable_gearing(boardID,
(u16)(1 << Slave) ); }
      if(status ==
NIMC_noError){status=flex_configure_stepper_output(boardID,Slave,NIMC_S
TEP_AND_DIRECTION,NIMC_ACTIVE_LOW,NIMC_OPEN_COLLECTOR);}
    return(status);
}
i32 checkModalErrors(u8& bID)
{
    i32 status=0;
      u8 boardID=bID;
      //Variables for modal error handling
      u16 commandID;
      u16 resourceID;
      u16 errorCode;
      u16 csr=0;

      if (status == NIMC_noError){
                //Read the Communication Status Register - check the
modal error bit
                status = flex_read_csr_rtn(boardID, &csr);
        }
        if (status == NIMC_noError){
                while (csr & NIMC_MODAL_ERROR_MSG){
                        //Get the command ID, resource and the error
code of the modal error from the error stack on the board
                        status = flex_read_err_msg_rtn(boardID,
&commandID, &resourceID, &errorCode);
                        if (status == NIMC_noError){
```

```
                                ErrorHandler(errorCode, commandID,
resourceID);
                        }else break;
                        //Read the Communication Status Register -
check the modal error bit
                        status = flex_read_csr_rtn(boardID, &csr);
                        if (status) break;
                    }
                }
                return(status);
}

i32 doPreHoming(u8& bID)//to take the wipers x inches above the
stamping stage. Motions follow next
{
 u8 boardID=bID;
      i32 status=0;                              //return status
      u8 Y_XY2 = NIMC_AXIS2;
      u8 Y_XY1 = NIMC_AXIS4;
      u8 DRIVE = NIMC_AXIS5;
      u16 axisStatusOf_Y_XY1,axisStatusOf_Y_XY2;
      u8 inputVector = 0xFF;  //input vector
    i16 numerator=-111;//-1;
      u16 denominator=100;//1; //for the gearing ratio.
      //Variables for modal error handling
      u16 commandID;                             //The commandID of the
function
      u16 resourceID;                      //The resource ID
      u16 errorCode;                             //The error generated
      u16 csr=0;

      //Load trajectory parameters for Y_XY2, i.e the y motor of the
XY-positioner2
    if(status == NIMC_noError){
      status=loadTrajectory( boardID, Y_XY2,JERK_Y_XY2_MOVE6,
ACLN_Y_XY2_MOVE6, DIS_Y_XY2_MOVE6, VEL_Y_XY2_MOVE6);
      }
    if(status == NIMC_noError){
      status=loadTrajectory( boardID, Y_XY1,JERK_Y_XY1_MOVE6,
ACLN_Y_XY1_MOVE6, DIS_Y_XY1_MOVE6, VEL_Y_XY1_MOVE6);
      }

    //ELECTRONIC GEARING--DRIVE TO Y_XY2
    if(status == NIMC_noError){
      status=configureGearing(boardID, Y_XY2,DRIVE,
numerator,denominator);
      }
      //Start the motions simultaneously
      status = flex_start(boardID, NIMC_AXIS_CTRL, AXES_245);
            printf("\n Running the Move6 now\n");

      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, Y_XY1,
&axisStatusOf_Y_XY1);
```

```
            if(status) break;
                  status = flex_read_axis_status_rtn(boardID, Y_XY2,
&axisStatusOf_Y_XY2);
            if(status) break;

      }while ( ! ( (axisStatusOf_Y_XY1 &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) &&
                  (axisStatusOf_Y_XY2 & (NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) )); //Test against the move complete bit or the
axis off bit
      }

      //DE-ENERGIZE ALL THE AXIS TO SAVE POWER
      if(status == NIMC_noError){ status = flex_stop_motion(boardID,
NIMC_AXIS_CTRL, NIMC_KILL_STOP, AXES_245);      }

      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
    }
        if(status == NIMC_noError){
                status = flex_enable_gearing(boardID, 0); //disable the
gearing----IS IT REALLY NEEDED???
        }
        if(status){ErrorHandler((u16)status, 0, 0);}

            return(status);



}
i32 doEndHoming(u8& bID)
{
 u8 boardID=bID;
      i32 status=0;                  //return status
      u8 Y_XY2 = NIMC_AXIS2;  // The y axis of XY-positioner2
      u8 Y_XY1 = NIMC_AXIS4;  // The y axis of XY-positioner2.
      u8 SUPPLY = NIMC_AXIS6; // The supply spool electronically geared
to Y_XY2
      u16 axisStatusOf_Y_XY2, axisStatusOf_Y_XY1; //Status of the axes
      u8 inputVector = 0xFF;  //input vector

    i16 numerator;  //-3;
      u16 denominator;//2;      //for the gearing ratio.
    u16 csr=0;
      //Variables for modal error handling
      u16 commandID;                //The commandID of the function
      u16 resourceID;               //The resource ID
      u16 errorCode;                //The error generated

      //Load trajectory parameters for Y_XY2 and Y_XY1
      if(status == NIMC_noError){
      status=loadTrajectory( boardID, Y_XY2,JERK_Y_XY2_MOVE1,
ACLN_Y_XY2_MOVE1, DIS_Y_XY2_MOVE1, VEL_Y_XY2_MOVE1);
      }
      if(status == NIMC_noError){
```

```
      status=loadTrajectory( boardID, Y_XY1,JERK_Y_XY1_MOVE1,
ACLN_Y_XY1_MOVE1, DIS_Y_XY1_MOVE1, VEL_Y_XY1_MOVE1);
      }

      //Calculate the electronic gearing ratio
    //numerator=(Xs*(Ys-abs(DIS_Y_XY2_MOVE1)/2)/sqrt(pow((Ys-
abs(DIS_Y_XY2_MOVE1)/2),2)+pow(Xs,2))-Xg*(Ys-
abs(DIS_Y_XY2_MOVE1)/2)/sqrt(pow((Ys-
abs(DIS_Y_XY2_MOVE1)/2),2)+pow(Xg,2)))*100;

      numerator=83;//YOU WILL HAVE TO MAKE THIS -83 IN CASE THE SUPPLY
MOTOR STARTS TO TURN CLOCKWISE FOR POSITIVE SIGNAL
      denominator=100;

      //End calculating the electronic gearing ratio

     //ELECTRONIC GEARING: SUPPLY IS SLAVE TO Y_XY2
      if(status == NIMC_noError){
       status=configureGearing(boardID, Y_XY2,SUPPLY,
numerator,denominator);
      }
      //Start the motions
      if(status == NIMC_noError){status = flex_start(boardID,
NIMC_AXIS_CTRL, AXES_246);}
      printf("\n Running the Move1 now\n The two wipers are touching
the stamping stage\n");
      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, Y_XY1,
&axisStatusOf_Y_XY1);
            if(status) break;
                  status = flex_read_axis_status_rtn(boardID, Y_XY2,
&axisStatusOf_Y_XY2);
            if(status) break;

      }while ( ! ( (axisStatusOf_Y_XY1 &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) &&
                  (axisStatusOf_Y_XY2 & (NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT )) )); //Test against the move complete bit or the
axis off bit
      }

      //Bring all axes to halt
      if(status == NIMC_noError){ status = flex_stop_motion(boardID,
NIMC_AXIS_CTRL, NIMC_HALT_STOP, AXES_246);      }

      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
    }

       if(status == NIMC_noError){
              status = flex_enable_gearing(boardID, 0); //disable the
gearing----IS IT REALLY NEEDED???
       }
```

```
        if(status){ErrorHandler((u16)status, 0, 0);}

            return(status);


}
```

```
// Stamping.cpp : Defines the entry point for the console application.
//THIS IS THE REAL FILE WHICH WILL MAKE EVERYTHING WORK/////////

#include "stdafx.h"
#include "flexmotn.h"
#include "stdio.h"
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define AXES_4    0x10
#define AXES_14   0x12
#define AXES_2   0x4
#define AXES_1    0X2
#define M 10           //microstepping index
# define NUM_CYCLES 5 //The number of cycles for which the printing
takes place

//definitions for the motions, velocities,accelerations and jerks
#define DIS1_INKING_MOVE1  105000
//#define DIS2_INKING_MOVE1  8.0
#define VEL1_INKING_MOVE1  10000
//#define VEL2_INKING_MOVE1    2.0
#define ACLN1_INKING_MOVE1  2.0
//#define ACLN2_INKING_MOVE1  5.0
#define JERK1_INKING_MOVE1  2.0
//#define JERK2_INKING_MOVE1  5.0
#define SLEEP_INKING        2000 //milliseconds between forth and
backward motion of the inking roller

#define DIS_INKING_MOVE2  -105000
#define VEL_INKING_MOVE2    10000
#define ACLN_INKING_MOVE2  2.0
#define JERK_INKING_MOVE2  2.0
#define DIS_Z_MOVE2    -161000 //modify this so that tehre are 160000
microsteps
#define VEL_Z_MOVE2    10000
#define ACLN_Z_MOVE2    2.0
#define JERK_Z_MOVE2    2.0

/*#define DIS_INKING_MOVE3  -18.0
#define VEL_INKING_MOVE3    3.0
#define ACLN_INKING_MOVE3  2.0
#define JERK_INKING_MOVE3  2.0*/
#define VEL_ROT_MOVE3       0.167    //radians per second
#define ACLN_ROT_MOVE3      0.6
#define JERK_ROT_MOVE3      2.0
```

```
#define DIS_Z_MOVE4   161000 //modify this so that tehre are 160000
microsteps
#define VEL_Z_MOVE4    10000
#define ACLN_Z_MOVE4   2.0
#define JERK_Z_MOVE4   2.0


/////Prototyes of functions used////
void ErrorHandler(u16 errorCode, u16 commandID, u16 resourceID);
i32 loadTrajectoryRotaryTable(u8& bID,u8& axis,u16 jerk, u32 acln, i32
vel,double gearRatio);
i32 loadTrajectoryLeadScrew(u8& bID,u8& axis,u16 jerk, u32 acln, i32
dis,i32 vel, double pitch);
i32 checkModalErrors(u8& bID);
i32 doMove1(u8& bID);//inking roller moves forward by 28 inches (in 2
internal steps)
i32 doMove2(u8& bID);//inking roller goes back by 10 inches and the z
lead screw moves down by 11 inches
i32 doMove3(u8& bID);//inking roller goes back by 18 inches & rotary
table rotates by 180 degrees
i32 doMove4(u8& bID);//z lead screw lifts up by 11 inches

// Main routine
void __cdecl main(void){

      i32 status=0;                              //The return status of
the function (for non-modal erros)
      u8 boardID=2;                              //The board
identification paramenter
      u16 csr=0;

    //int i;
      //for(i=0;i<NUM_CYCLES;i++){

      //Check if the board is at power up reset condition
      status = flex_read_csr_rtn(boardID, &csr);

      if(status == NIMC_noError){
            if (csr & NIMC_POWER_UP_RESET ){
                  printf("\nThe FlexMotion board is in the reset
condition. Please initialize the board before running this example.
The Initialize.c example shows you how to do this.\n");
                  exit(0);
            }
      }



      if (status == NIMC_noError){
            //Do the sequential moves
            status = doMove1(boardID); //inking roller moves forward by
28 inches (in 2 internal steps)
                  if (status)ErrorHandler((u16)status, 0, 0);
            Sleep(SLEEP_INKING); // To synchronize the two machines.
            status = doMove2(boardID);//inking roller goes back by 10
inches and the z lead screw moves down by 11 inches
                  if (status)ErrorHandler((u16)status, 0, 0);
```

```
            status = doMove3(boardID);//inking roller goes back by 18
inches & rotary table rotates by 180 degrees
                  if (status)ErrorHandler((u16)status, 0, 0);
            status = doMove4(boardID);//z lead screw lifts up by 11
inches
                    if (status)ErrorHandler((u16)status, 0, 0);
      }

      printf("\n\nFinished.\n");

      exit(0);

}//end main


i32 doMove1(u8& bID) //inking roller moves forward by 28 inches
{
    u8 boardID=bID;
      i32 status=0;                  //return status
      u8 INKING = NIMC_AXIS4; // The axis of the inking roller lead
screw
      u16 axisStatusOf_INKING; //Status of the axes
      u8 inputVector = 0xFF;  //input vector
    u16 csr=0;
      //Variables for modal error handling
      u16 commandID;                 //The commandID of the function
      u16 resourceID;                //The resource ID
      u16 errorCode;                 //The error generated

      //Load trajectory parameters for the first part of the Move1
      if(status == NIMC_noError){
      status=loadTrajectoryLeadScrew(boardID,INKING,JERK1_INKING_MOVE1,
ACLN1_INKING_MOVE1,DIS1_INKING_MOVE1,VEL1_INKING_MOVE1,0.1);
      }
      //Start the first part of the motion
      if(status == NIMC_noError){
            status = flex_start(boardID, NIMC_AXIS4, AXES_4);}
      printf("\n Running the first 20 inches of the inking roller,i.e,
first part of Move1 now\n");
      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, INKING,
&axisStatusOf_INKING);
            if(status) break;
      }while ( ! ( (axisStatusOf_INKING &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT ))));
      }

      //Bring the axes to halt
      if(status == NIMC_noError){ status = flex_stop_motion(boardID,
NIMC_AXIS4, NIMC_HALT_STOP, AXES_4);}
printf("move complete\n");
      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
```

```
    }
        if(status){ErrorHandler((u16)status, 0, 0);}

            return(status);
}


i32 doMove2(u8& bID) //inking roller goes back by 28 inches and the z
lead screw moves down by 11 inches
{
      u8 boardID=bID;
      i32 status=0;
      u8 INKING = NIMC_AXIS4;
      u8 Z = NIMC_AXIS1;
      u16 axisStatusOf_Z,axisStatusOf_INKING;
      u8 inputVector = 0xFF;
      u16 commandID;
      u16 resourceID;
      u16 errorCode;
      u16 csr=0;

      //Load trajectory parameters for INKING
      if(status == NIMC_noError){
      status=loadTrajectoryLeadScrew(boardID,INKING,JERK_INKING_MOVE2,A
CLN_INKING_MOVE2,DIS_INKING_MOVE2,VEL_INKING_MOVE2,0.1);
      }
      //Load trajectory parameters for Z
      if(status == NIMC_noError){
      status=loadTrajectoryLeadScrew(boardID,Z,JERK_Z_MOVE2,ACLN_Z_MOVE
2,DIS_Z_MOVE2,VEL_Z_MOVE2,0.125);
      }
      //Start the motions
      status = flex_start(boardID, NIMC_AXIS_CTRL, AXES_14);
      printf("\n Running the Move2 now\n");
      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, Z,
&axisStatusOf_Z);
            if(status) break;
                  status = flex_read_axis_status_rtn(boardID, INKING,
&axisStatusOf_INKING);
            if(status) break;
            }while ( ! ( (axisStatusOf_Z &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT ))&& (axisStatusOf_INKING &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT ))));
        }
      //Bring all axes to halt
      if(status == NIMC_noError){
     status = flex_stop_motion(boardID, NIMC_AXIS_CTRL, NIMC_HALT_STOP,
AXES_14);
      }
      printf("move complete\n");
      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
```

```
    }
        if(status){ErrorHandler((u16)status, 0, 0);}
            return(status);

}
i32 doMove3(u8& bID)//inking roller goes back by 18 inches & rotary
table rotates by 180 degrees
{
    u8 boardID=bID;
      i32 status=0;
      u8 ROT = NIMC_AXIS2;
      u16 axisStatusOf_ROT;
      u8 inputVector = 0xFF;
      u16 commandID;
      u16 resourceID;
      u16 errorCode;
      u16 csr=0;

      if(status == NIMC_noError){
      status = flex_stop_motion(boardID, NIMC_AXIS2, NIMC_HALT_STOP,
ROT);
      }
      //Load trajectory parameters for ROT
      if(status == NIMC_noError){
      status=loadTrajectoryRotaryTable(boardID,ROT,JERK_ROT_MOVE3,ACLN_
ROT_MOVE3,VEL_ROT_MOVE3,6.0);
      }
      //Start the motions
      status = flex_start(boardID, NIMC_AXIS2, ROT);
      printf("\n Running the Move3 now\n");
      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, ROT,
&axisStatusOf_ROT);
            if(status) break;
            }while ( ! ( (axisStatusOf_ROT &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT ))));
        }
      //Bring all axes to halt
      if(status == NIMC_noError){
      status = flex_stop_motion(boardID, NIMC_AXIS2, NIMC_HALT_STOP,
ROT);
      }
      printf("move complete\n");
      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
    }
        if(status){ErrorHandler((u16)status, 0, 0);}
            return(status);
}
i32 doMove4(u8& bID) {//z lead screw lifts up by 11 inches
    u8 boardID=bID;
      i32 status=0;
      u8 Z = NIMC_AXIS1;
```

```
      u16 axisStatusOf_Z;
      u8 inputVector = 0xFF;
    u16 commandID;
      u16 resourceID;
      u16 errorCode;
      u16 csr=0;

      //Load trajectory parameters for Z
    if(status == NIMC_noError){
      status=loadTrajectoryLeadScrew( boardID, Z,JERK_Z_MOVE4,
ACLN_Z_MOVE4, DIS_Z_MOVE4, VEL_Z_MOVE4,0.125);
      }

    //Start the motions simultaneously
      status = flex_start(boardID, NIMC_AXIS1, Z);
            printf("\n Running the Move4 now\n");

      //Wait for this motion to complete and Test against the move
complete bit
      if(status == NIMC_noError){
            do{
            status = flex_read_axis_status_rtn(boardID, Z,
&axisStatusOf_Z);
            if(status) break;

            }while ( ! ( (axisStatusOf_Z &(NIMC_MOVE_COMPLETE_BIT |
NIMC_AXIS_OFF_BIT ))));
        }
      if(status == NIMC_noError){ status = flex_stop_motion(boardID,
NIMC_AXIS1, NIMC_HALT_STOP, AXES_1);      }
printf("move complete\n");
      //Checking modal errors here
    if(status == NIMC_noError){
       status=checkModalErrors(boardID);
    }
        if(status){ErrorHandler((u16)status, 0, 0);}
            return(status);

}

void ErrorHandler(u16 errorCode, u16 commandID, u16 resourceID){

      i8 *errorDescription;         //Pointer to i8's -  to get error
description
      u32 sizeOfArray;                    //Size of error description
      u16 descriptionType;                //The type of description to
be printed
      i32 status=0;                                    //Error returned
by function

      if(commandID == 0){
            descriptionType = NIMC_ERROR_ONLY;
      }else{
            descriptionType = NIMC_COMBINED_DESCRIPTION;
      }

      //First get the size for the error description
```

```
	sizeOfArray = 0;
	errorDescription = NULL;//Setting this to NULL returns the size
required
	status = flex_get_error_description(descriptionType, errorCode,
commandID, resourceID,

	errorDescription, &sizeOfArray );

	//Allocate memory on the heap for the description
	errorDescription = (i8*)malloc(sizeOfArray + 1);

	sizeOfArray++; //So that the sizeOfArray is size of description +
NULL character
	// Get Error Description
	status = flex_get_error_description(descriptionType, errorCode,
commandID, resourceID,

	errorDescription, &sizeOfArray );

	if (errorDescription != NULL){
		printf("\n");
		printf(errorDescription);//Print description to screen
		free(errorDescription);//Free allocated memory
	}else{
		printf("Memory Allocation Error");
	}
}

i32 loadTrajectoryRotaryTable(u8& bID,u8& axis,u16 jerk, u32 acln, i32
vel,double gearRatio)
{
	u8 boardID=bID;
	i32 status=0;
	u8 inputVector = 0xFF;

	if(status == NIMC_noError){
	status = flex_load_scurve_time(boardID, axis, (u16)jerk,
inputVector);
	}
	if(status == NIMC_noError){
	status = flex_load_acceleration(boardID, axis, NIMC_BOTH,
(u32)(acln*100*M*gearRatio), inputVector);
	}
	if(status == NIMC_noError){
	status = flex_load_velocity(boardID, axis,
(i32)(vel*100*M*gearRatio), inputVector);
	}
	if(status == NIMC_noError){
	status = flex_load_target_pos(boardID, axis,
(i32)(100*M*gearRatio), inputVector);
	}
	if(status == NIMC_noError){
	status = flex_set_op_mode(boardID, axis, NIMC_RELATIVE_POSITION);
	}
	if(status == NIMC_noError){
	status=flex_configure_stepper_output(boardID,axis,NIMC_STEP_AND_D
IRECTION,NIMC_ACTIVE_LOW,NIMC_OPEN_COLLECTOR);
```

```
	}
	return(status);
}
i32 loadTrajectoryLeadScrew(u8& bID,u8& axis,u16 jerk, u32 acln, i32
dis,  i32 vel,double pitch)
{
	u8 boardID=bID;
	i32 status=0;
	u8 inputVector = 0xFF;

	if(status == NIMC_noError){
	status = flex_load_scurve_time(boardID, axis, (u16)jerk,
inputVector);
	}
	if(status == NIMC_noError){
	status = flex_load_acceleration(boardID, axis, NIMC_BOTH,
(u32)(acln*M*200/pitch), inputVector);
	}
	if(status == NIMC_noError){
	status = flex_load_velocity(boardID, axis, (i32)vel,
inputVector);
	}
	if(status == NIMC_noError){
	status = flex_load_target_pos(boardID, axis,  (i32)dis,
inputVector);
	}
	if(status == NIMC_noError){
	status = flex_set_op_mode(boardID, axis, NIMC_RELATIVE_POSITION);
	}
	if(status == NIMC_noError){
	status=flex_configure_stepper_output(boardID,axis,NIMC_STEP_AND_D
IRECTION,NIMC_ACTIVE_LOW,NIMC_OPEN_COLLECTOR);
	}
	return(status);
}
i32 checkModalErrors(u8& bID)
{
    i32 status=0;
	u8 boardID=bID;
	//Variables for modal error handling
	u16 commandID;
	u16 resourceID;
	u16 errorCode;
	u16 csr=0;

	if (status == NIMC_noError){
		//Read the Communication Status Register - check the
modal error bit
		status = flex_read_csr_rtn(boardID, &csr);
	  }
	  if (status == NIMC_noError){
		while (csr & NIMC_MODAL_ERROR_MSG){
			//Get the command ID, resource and the error
code of the modal error from the error stack on the board
			status = flex_read_err_msg_rtn(boardID,
&commandID, &resourceID, &errorCode);
			if (status == NIMC_noError){
```

```
                                ErrorHandler(errorCode, commandID,
resourceID);
                        }else break;
                        //Read the Communication Status Register -
check the modal error bit
                        status = flex_read_csr_rtn(boardID, &csr);
                        if (status) break;
                }
            }
            return(status);
}
```