To the Graduate Council:

I am submitting herewith a thesis written by Richard S. Paulus entitled "Development and Evaluation of an Improved Positioning Scheme for a Cyclotron Ion Source." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mechanical Engineering.

Frank Speckhart, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Requests for permission for extensive quotation from or reproduction of this thesis in whole or in parts may be granted by the copyright holder.

Signature 

Date November 29, 1994

DEVELOPMENT AND EVALUATION OF AN IMPROVED POSITIONING SCHEME FOR A CYCLOTRON ION SOURCE

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Richard Semmes Paulus

December 1994

This thesis is dedicated to the memory of Albert and Ella Paulus who, through their lives, helped teach me the importance of family and the power of education. Thank you for our time at Southwood.

ACKNOWLEDGMENTS

First, I would like to thank my major professor, Dr. Frank Speckhart, for his guidance and insight during the course of this project. It was a pleasure to benefit from his experience and acumen. I would like to thank Dr. Clement Wilson for his contributions throughout the project, most especially during the early conceptual evaluation stage and mechanism selection. I also appreciate the financial support that Dr. Wilson facilitated. Thank you to Dr. Robert Bodenheimer who, among other things, offered valuable encouragement and perspective during the difficult times of the project.

Second, I would like to express thanks to Kelly Milam and CTI for their financial, technical, and moral support. Thank you, especially, to Joe Matteo for his guidance and camaraderie during this project. I am pleased to have benefited from his experience and insight. Thank you, also, to Jesse Garcia for his CAD wizardry.

Third, thank you to my fellow graduate students, especially Jeremy Davis and Mike Kennedy, who offered generous support and encouragement during my tenure.

Finally, I would like to thank my family for their tremendous support and encouragement to "take the leap" and tackle graduate school. Thanks to my brother, Mike, for his unselfish mentoring and endless encouragement and for his help with optimizing the Pain-versus-Time Curve. It was a pleasure to work with him. Thanks to my parents, T.J. and Sue Paulus, for their classic Paulus mix of encouragement, enthusiasm, and expectation. God Bless PEC. Thanks, most especially, to my wife, Audrey, for her constant support and encouragement during the dark hours of Spring and Summer 1994 and for her willingness to be hungry. I appreciate her example of diligence and determination in achieving one's potential. I also appreciate her tireless energy and commitment in caring for our daughter, Caroline.

ABSTRACT

Cyclotrons are particle accelerators that are used to create radioactive drugs for nuclear medicine. Cyclotron efficiency is affected by the initial positioning of a plasma ion source within a cyclotron. CTI manufactures cyclotrons that use a kinematically inexact positioning scheme which relies on thread clearances to pivot an ion source.

This project analyzes CTI's current scheme, proposes design goals for an improved method, describes analysis and development of the improved mechanism, and presents experimental results and conclusions. The following design goals were defined:

- Range of Motion:
  x & y axes: ± 3 mm; z axis: ± 2 mm .
- Accuracy of Motion:
  x, y, & z axes: ± 0.2 mm .
- Minimum Required Increment of Movement:
  x, y, & z axes: 0.1 mm.

The design solution is a kinematically exact, non-redundant mechanism which positions the ion source with an accuracy of ± 0.2 mm from a remote actuation point 425 mm away from the ion source. The design uses stepper motors and a differential lead-screw drive system controlled by a personal computer.

A mathematical model describing the three degree-of-freedom system is developed and evaluated. The resultant closed-form solution is incorporated into a C program which operates on an Intel 80286-based personal computer.

The prototype system met the accuracy design goal, minimum increment of motion goal, and the x-axis range of motion goal. Y-axis range of motion was 95% of the design goal while x-axis range of motion was 85% of the design goal. Minor hardware modifications to the prototype hardware are proposed to address x-axis range of motion.

The design is evaluated and experimental data are discussed. Improvements for transition of the first-generation hardware to production use are discussed.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Background

Cyclotron Operation

Positron Emission Tomography (PET) is a medical diagnostic imaging technique. Unlike magnetic resonance imaging (MRI) or radiographic computed tomography (CT) which look at anatomy or body form, PET studies metabolic activity or body function.[1] PET scanners track the flow of injected radiopharmaceuticals through the body and measure the intensity of the radiation as an indication of the metabolic activity of the body part under investigation.[2]

The mechanism for tracing the flow of the injected radiopharmaceutical is the emission of positrons. As emitted positrons collide with electrons in the human body, two gamma rays are emitted in opposite directions as a result of the annihilation of the two elements. As large numbers of positrons collide with electrons, many pairs of gamma rays are propagated. Figure 1 shows a schematic arrangement of a PET scanner. Note the rings of detectors which sense incident gamma rays. As many lines of gamma rays are sensed by the detectors an image is generated by the scanner. This image represents a high density of metabolic activity of the radiopharmaceutical.

Cyclotrons are particle accelerators which bring about nuclear reactions that create radionuclides which can be synthesized into short half-life radiopharmaceuticals. Figure 2 shows a schematic representation of a cyclotron. Wolf and Jones describe a cyclotron as "an electromagnetic device in which a magnetic field is imposed on an evacuated chamber located perpendicular to an electric field. The ions which move in a spiral in increasing energy increments can be eventually deflected and brought outside the vacuum chamber

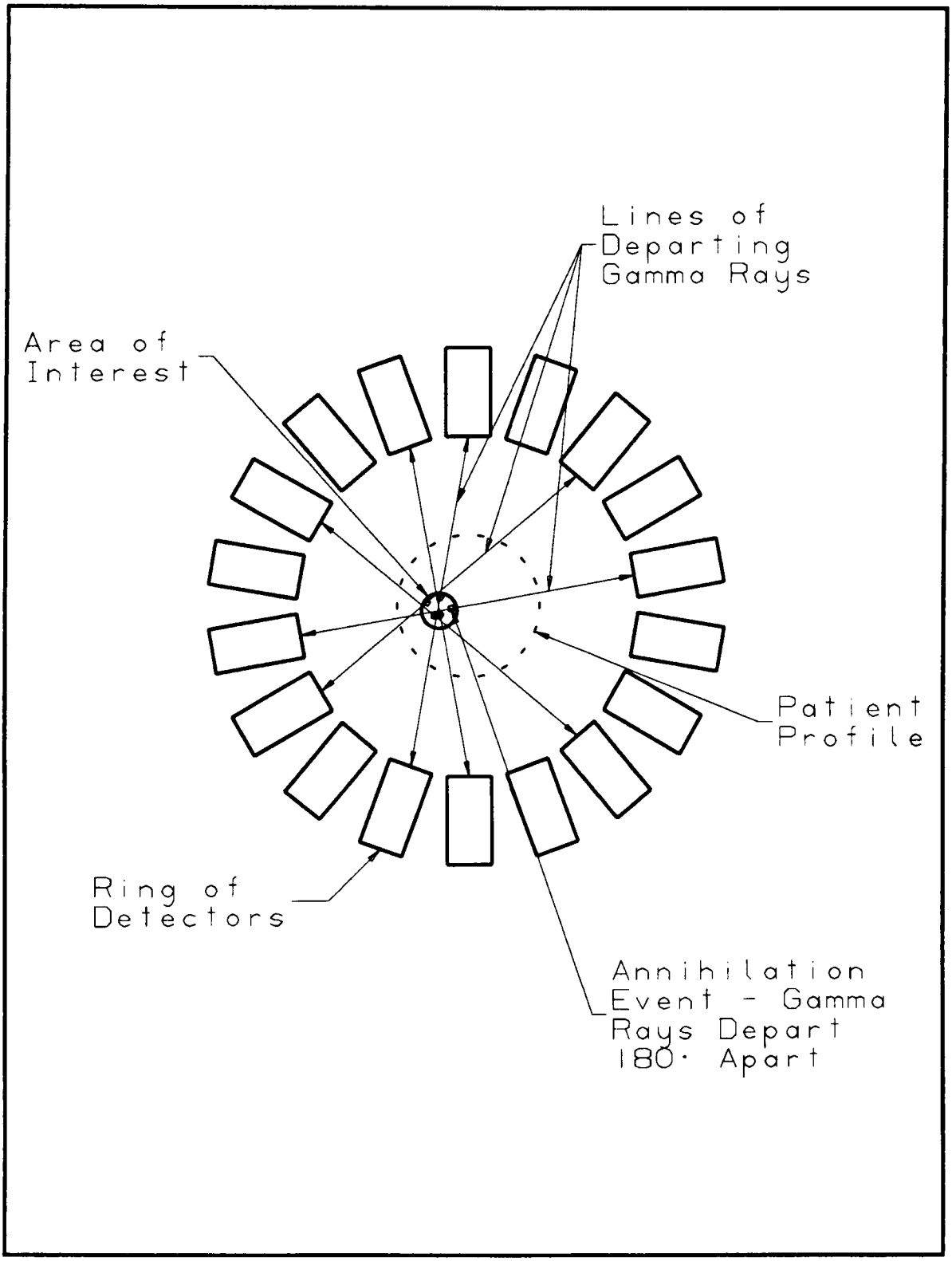


Figure 1
Schematic Representation of a Positron Emission Tomography (PET) Scanner

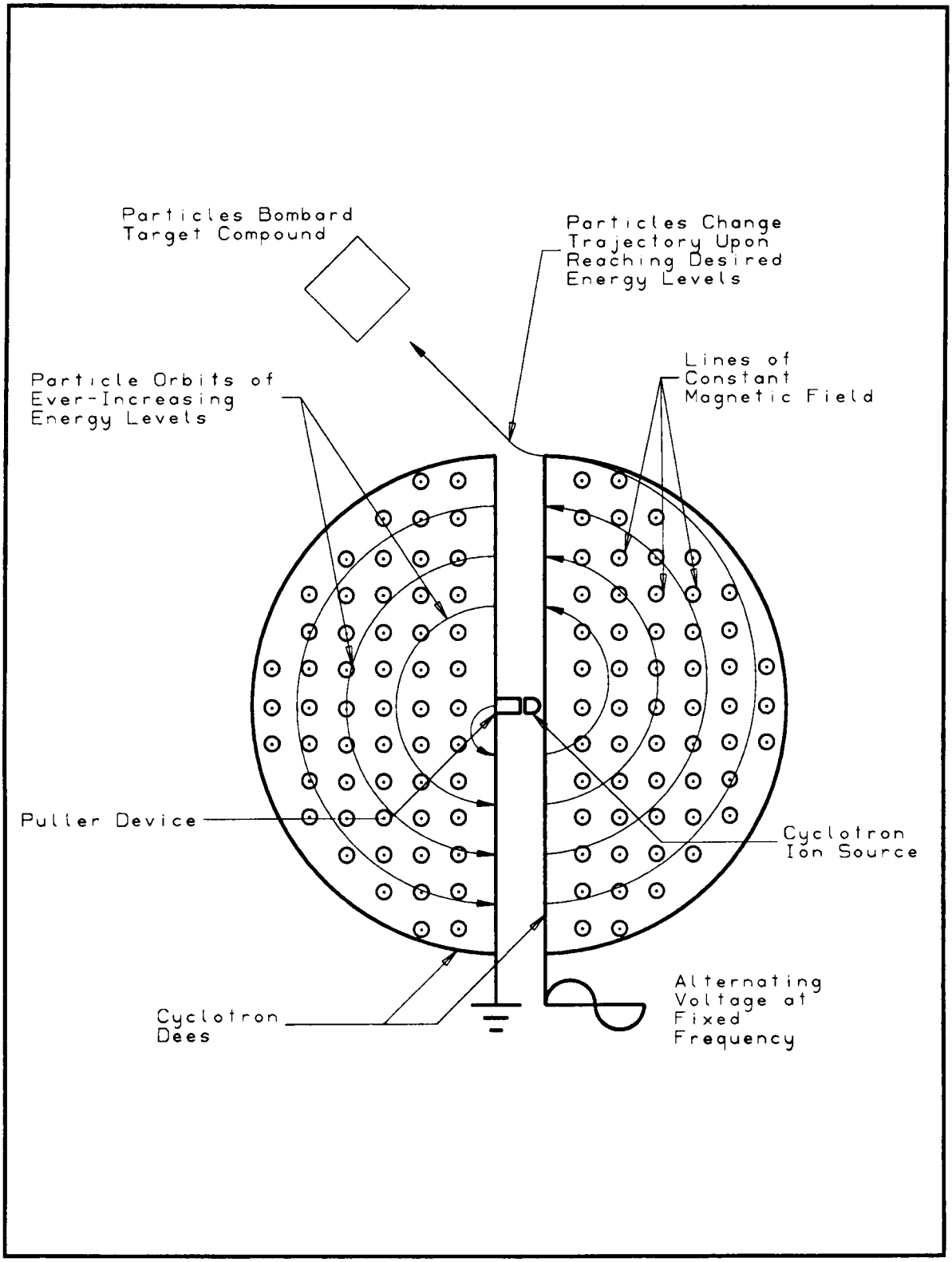


Figure 2
Schematic Representation of a Cyclotron

after passing through a thin foil window. The ion source which produces the ions to be accelerated is located in the center of the cyclotron. [The source] may be introduced either radially (from the side wall of the vacuum chamber) or axially through a hole in the center of the magnet yoke."[3]

The electric field generated in the center of the cyclotron is maintained by semicircular electrodes, called dees (due to their resemblance to the capital letter "D"), that are driven by an alternating voltage of fixed frequency. With the ion source inserted into the cyclotron between the dees, ions are pulled from the anode of the source plasma by the resident electric field. A puller device assists in drawing ions from the plasma out into the region between the dees where the ions experience effects from both the constant magnetic field and the time varying electric field.[4] As a consequence, initial positioning of the anode relative to the puller significantly affects the speed and trajectory of the emerging ions, which affect the overall ion beam. Figure 3 shows the orientation of the *xyz* axes for the critical interface between the anode and puller.

Ion source position may be fine-tuned to improve overall performance of the cyclotron. CTI Cyclotron Systems Inc. (CCS), a division of CTI, Inc., designs and manufactures negative-ion cyclotrons to provide radionuclides for use with PET scanners manufactured by CTI PET Systems, Inc.

Current Positioning System

CCS currently positions an ion source in a cyclotron by displacing the ion source mounting plate using three parallel lead screws. The three lead screws are turned by three alternating current motors. A top view of this scheme is depicted in Figure 4.

This method alternately holds two of the three drive motors fixed while the third motor drives one of the three lead screws. The resulting motion pivots the ion source mounting plate about an axis through the fixed lead screws which, in turn, induces x, y,

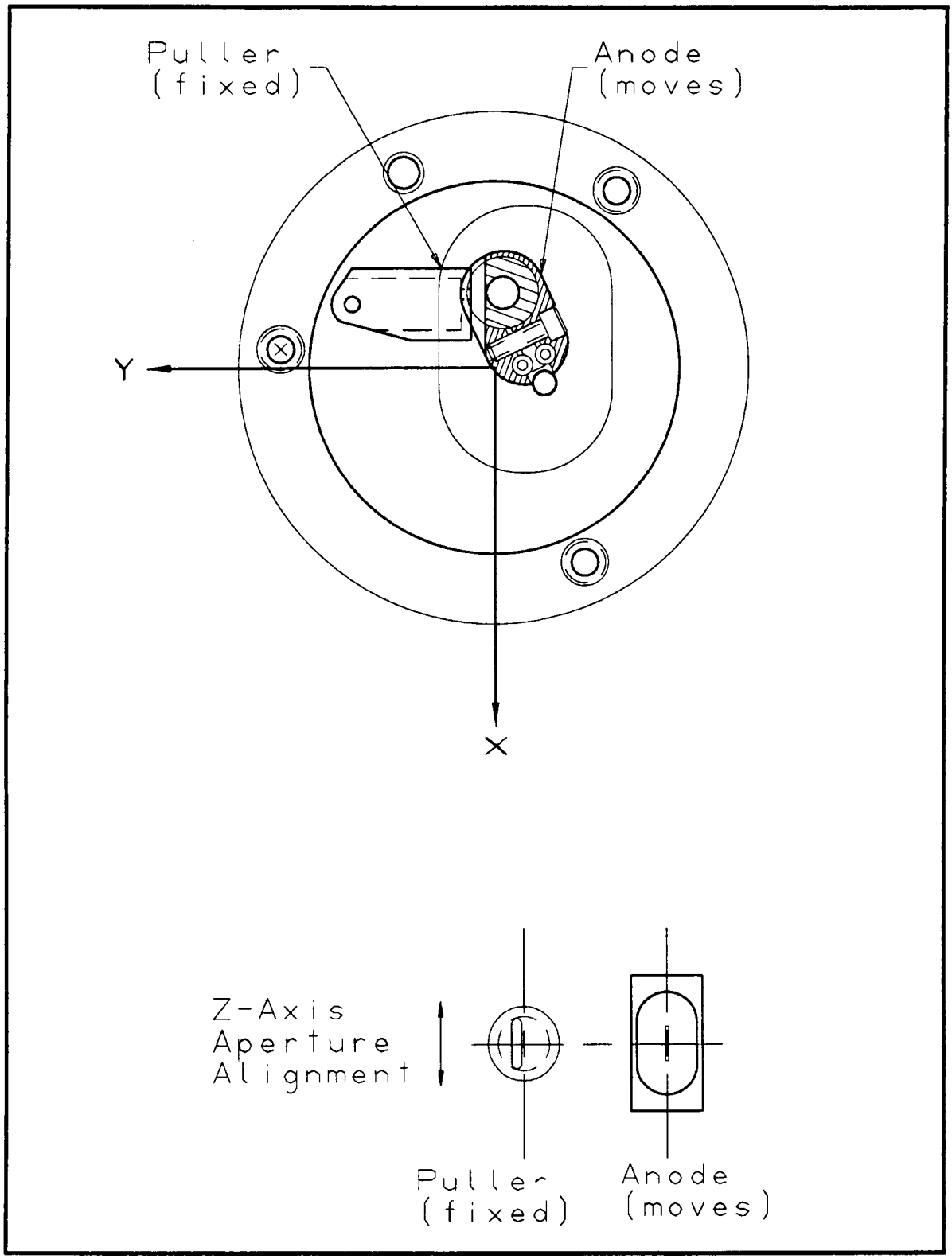


Figure 3
Orientation of X,Y,Z Axes at Anode-Puller Interface

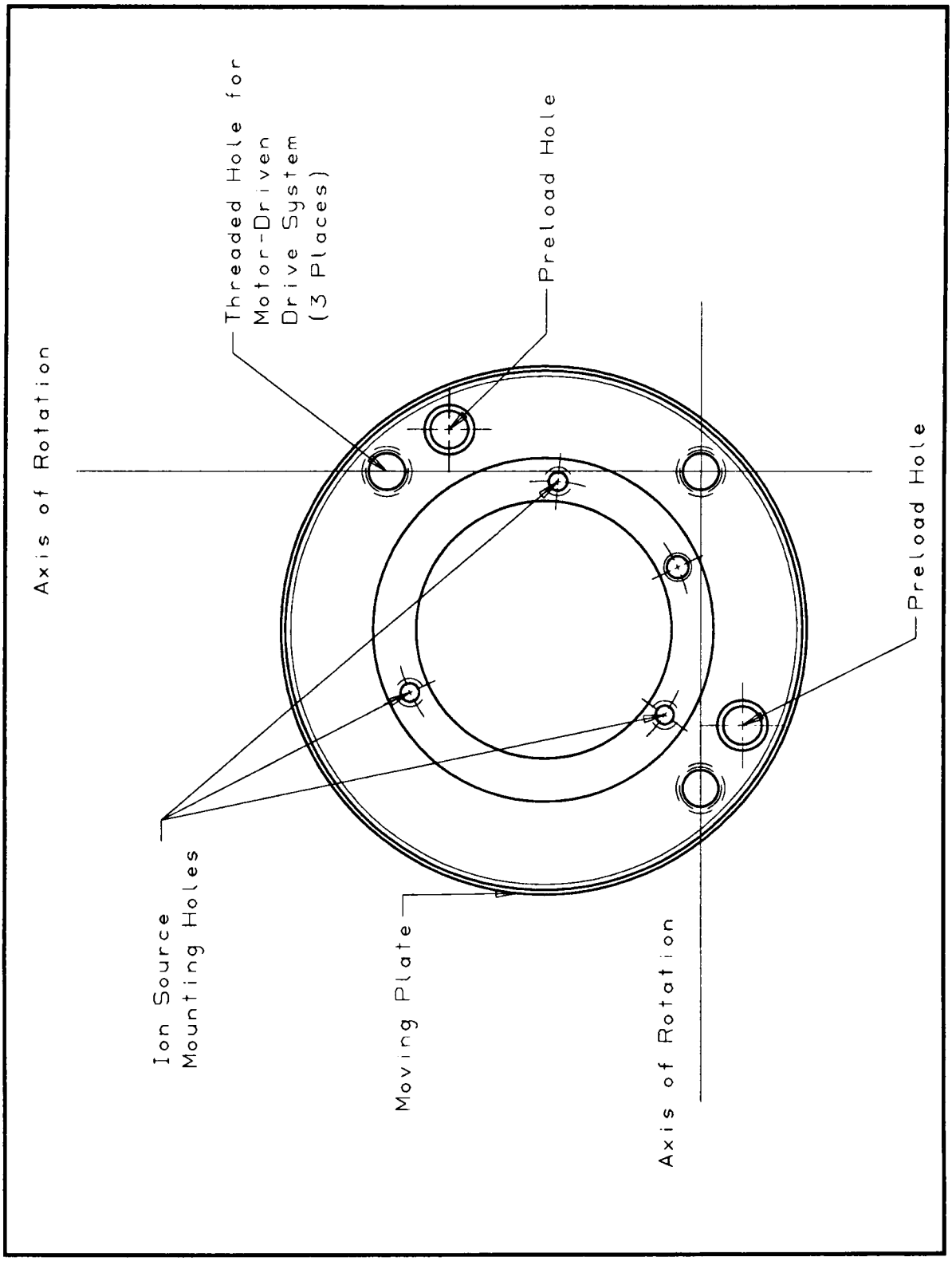


Figure 4
Top View of CTI's Current Positioning Scheme

and z motion at the anode-puller interface in the center of the cyclotron. Pivoting about the orthogonal axis requires actuation of the opposite diagonal lead screw with the remaining two screws held fixed. By driving all three motors in unison, displacement in only the z axis may be accomplished. Figure 5 shows a side view of one of the three motor / screw systems.

Three main issues necessitate redesign of the current system. First, the current system operates in an open-loop mode with no quantitative data describing the position of the source assembly relative to a known starting point. Since the ion source assembly is removed and reinstalled periodically during start-up and later for maintenance, an open-loop positioning scheme makes realignment cumbersome and time-consuming. Second, the current system does not generate kinematically predictable motion. System moves are heavily dependent on friction and clearances in the screw thread profiles. Third, the mechanical drive system experiences binding at the full range of motion of the system. It is hypothesized that binding occurs outside of the operating space of the positioning system after all clearances in the thread system have been consumed. Since the operator is limited to remote actuation of the drive motors without position feedback, the motors are easily driven beyond the intended range of motion and into a binding condition.

Figure 6 shows a photograph of the current drive system used by CCS. Figure 7 shows a photograph of the ion source used during this investigation. Figure 8 shows a close-up view of the anode aperture.

Statement of Problem

The central problem that this thesis addresses is the lack of predictable kinematic positioning of the cyclotron's ion source. Solution to this problem is a system that can move the anode portion of the ion source to a user-specified *xyz* location relative to the

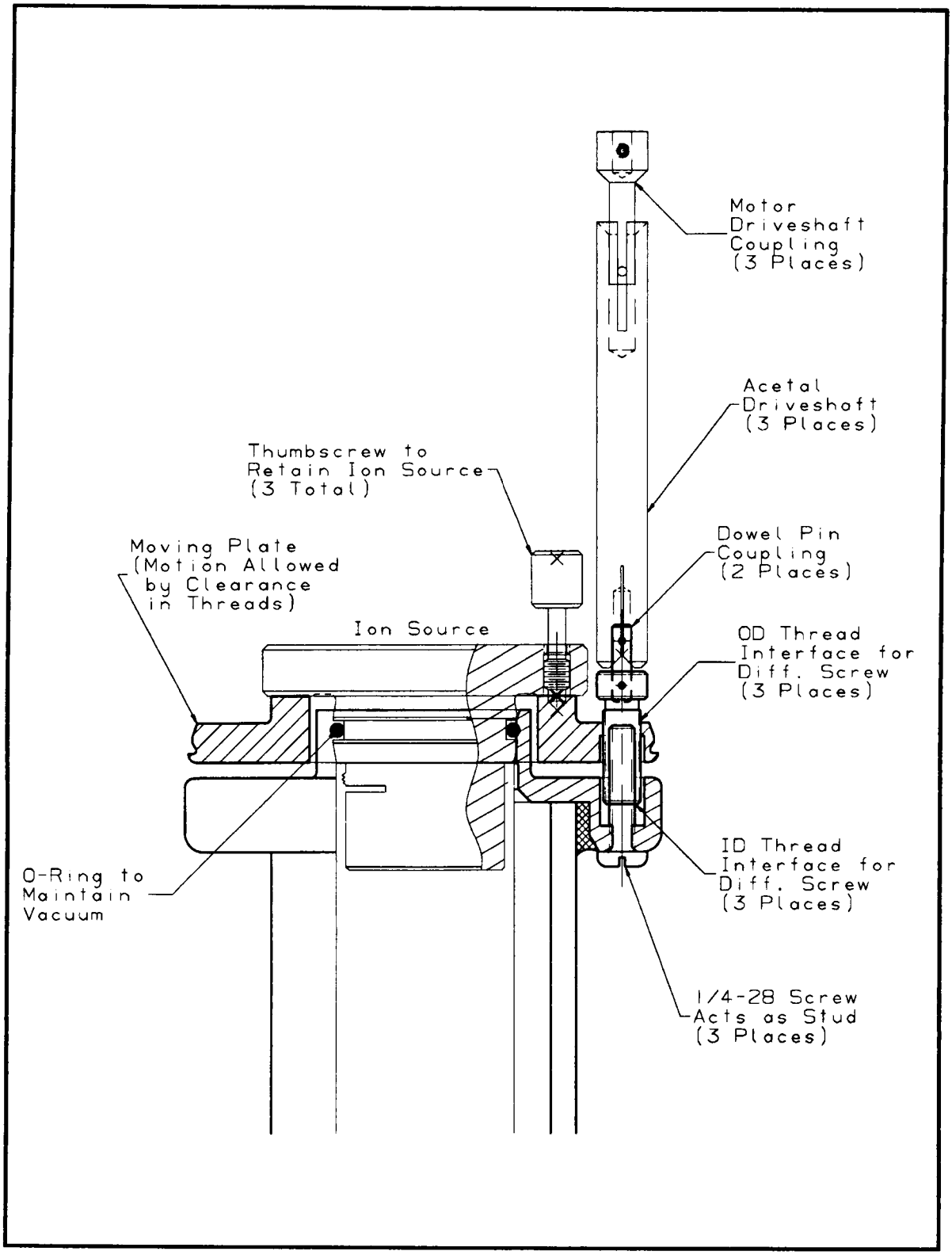


Figure 5
Side View of Motor / Screw System - CTI's Current System

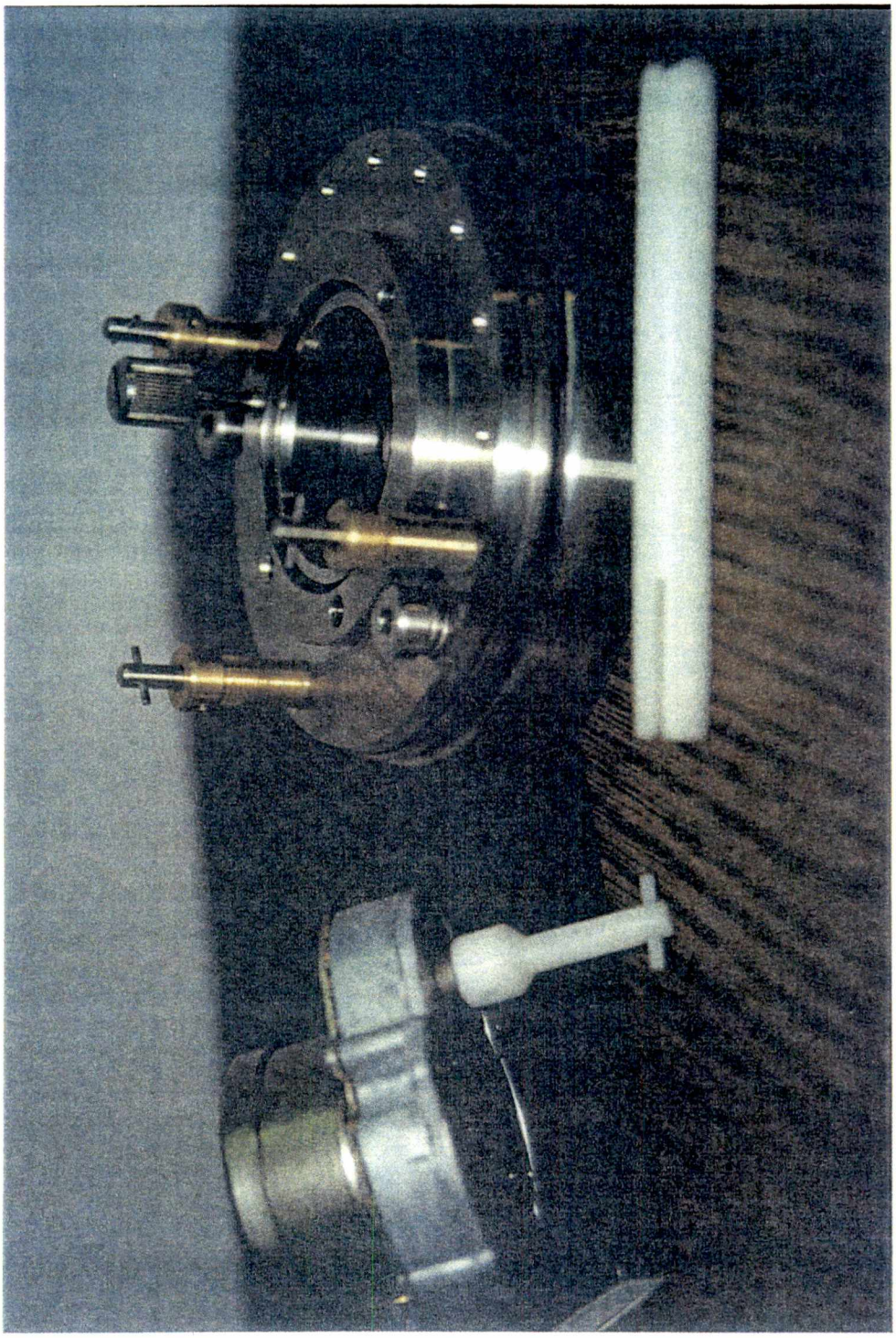


Figure 6
Photograph of CTI's Current Drive System

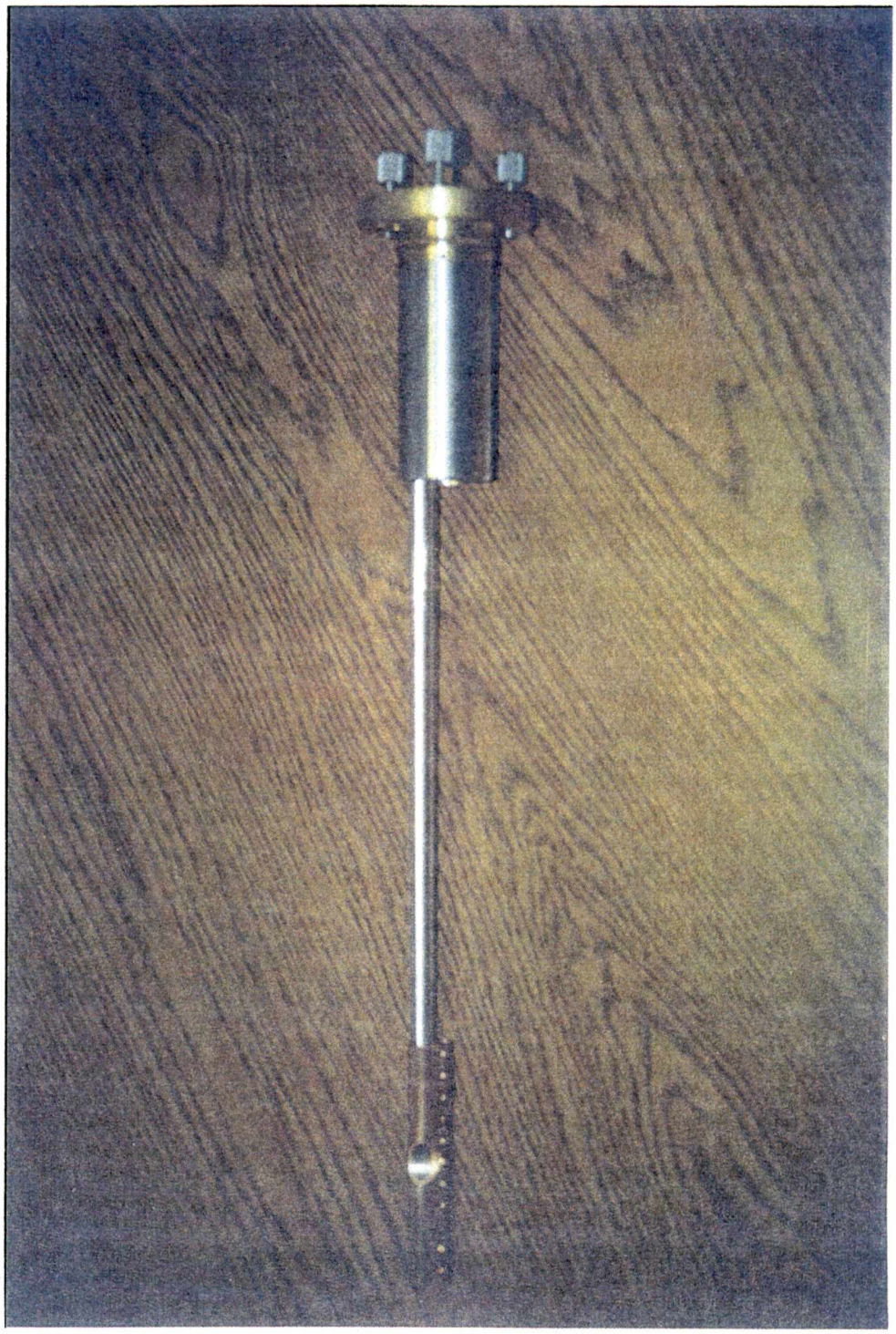


Figure 7
Photograph of Ion Source Used in Cyclotron

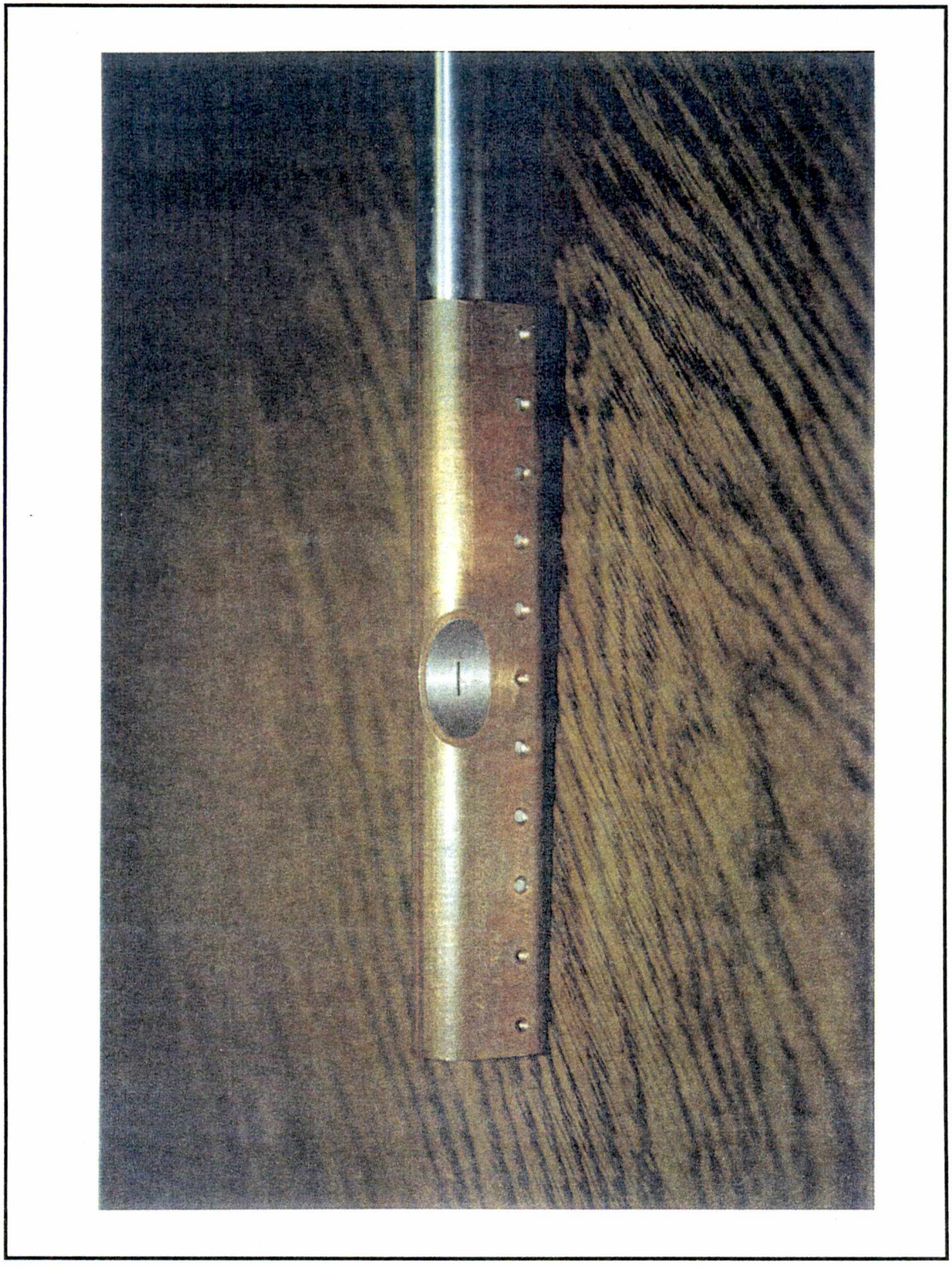


Figure 8
Photograph of Anode Aperture of Ion Source

ion puller. The user can choose trajectories along single axes or specify moves that involve multiple axes.

The following performance parameters guided system development:

- Range of Motion at Anode-to-Puller Interface:
  x & y axes: ± 3 mm (± 0.118 in) from a defined center position.
  z axis: ± 2 mm (± 0.079 in) from a defined center position.
- Accuracy of Motion at Anode-to-Puller Interface:
  x, y, & z axes: ± 0.2 mm (± 0.0079 in).
- Minimum Required Increment of Movement:
  x, y, & z axes: 0.1 mm (0.00394 in).

System evaluation utilized a coordinate measuring machine (CMM) for evaluating the machined dimensions of one part while a dial indicator and height gage were used to assess overall system performance as a result of mechanism actuation. Comparisons between predicted and actual motion were made and are presented in this document.

Project Scope

The scope of this thesis includes the analysis, design and evaluation of a prototype positioning system for an ion source under development at CCS. The main purpose is to demonstrate a functional concept. Full integration of this solution into production hardware is beyond the scope of this project but could be pursued in a follow-on effort.

Project Approach

Project activities proceeded in the following fashion:

1. Investigation of the problem.
2. Selection of a conceptual design using a kinematically-sound approach.

3. Creation of a mathematical model that describes the motion produced by the conceptual design (this model served as the main computational vehicle governing system movement calculations).

4. Detailed design of mechanical system and controls.

5. Generation of software code to control system drive motors.

6. Overall system evaluation.

7. Project write-up.

Sequence of Presentation

Chapter two discusses the analytical approach used to develop the governing mathematical model for this system. Included is the conceptual design evaluation, mathematical model development and results, and system error analysis.

Chapter three presents the detailed design of the mechanical system. Included are the governing design guidelines; torque, stress, and fits analyses; detailed discussion of the improved drive system; and description of mechanism components.

Chapter four describes the system control scheme including choice of software language and platform, implementation of mathematical model, and method for pulsing the motors using a dedicated parallel port of a personal computer.

Chapter five summarizes system evaluation. This material describes the criteria, methods, and results of the system evaluation. Theoretical and measured data are compared to validate the theoretical development of chapter one.

Chapter six states the conclusions of this project and describes recommendations for improving system cost and performance. Results are summarized and the significance of key design features are restated in the context of overall system performance.

References follow the six chapters with Appendices A-F at the end of this document.

specified *xyz* location is a three degree-of-freedom (DOF) problem. The following three options exist for producing the required motion:

- Gantry system (*x-y-z* table).
- Gimbal system with independent z motion.
- Pivoting system (3-leg table).

Figure 9 depicts typical configurations of these three schemes.

In considering these three approaches the gantry system was immediately discarded since the o-ring constraint precludes direct *x-y* actuation of the ion source assembly. Due to the pivoting nature of the allowed motion, the gimbal system and pivoting system received the most consideration.

As shown in Figure 9, the gimbal system consists of two concentric rings which pivot about orthogonal, coplanar axes which intersect at a common center point. The outer ring is connected through bearings to mounting hardware while the inner ring is connected through bearings to the outer ring. By mounting the ion source assembly on a circular plate which serves as the inner ring, the required pivoting motion of the assembly could conceptually be generated.

In contrast, the pivoting system consists of three legs which project through an upper plate onto a lower plate where they maintain constant contact. A uniform gap between the upper and lower plates could be achieved by forcing the three leg lengths between the two plates to be the same value. By varying the lengths of the three legs, the upper plate could be pivoted into an arbitrary position. A scheme based on this method of actuation is used in the current cyclotron.

Each of the two methods of actuation have positive and negative attributes. The following is a brief comparison of these two methods:

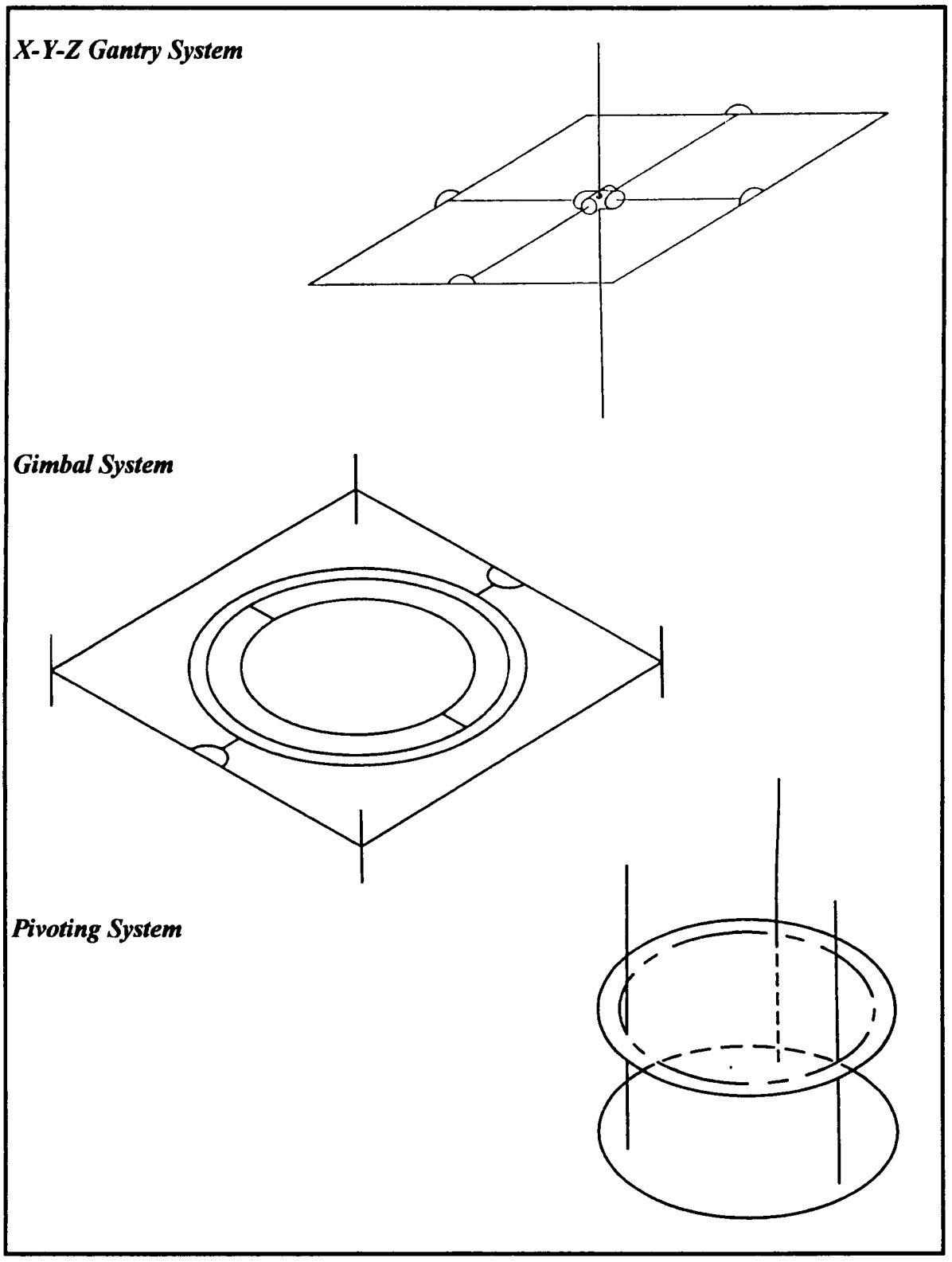


Figure 9
Three Possible Solutions to a 3 Degree-of-Freedom Positioning Problem

Gimbal System - Positive Aspects

- Could provide the required *xyz* motion at the anode-puller interface.
- Is well suited for the type of motion needed to actuate the ion source at the anode-puller interface since it generates rotational motion in two orthogonal axes.
- Generates very little *z* motion that would require compensation when pure *x* or *y* motion was specified.

Gimbal System - Negative Aspects

- *z* motion is not readily generated using this scheme.
- Space surrounding the potential location of a gimbal mechanism is at a premium in the existing hardware, placing strict requirements on the drive mechanism design.
- Drive backlash using rotary motion would become significant with this scheme as maximum angular displacements are on the order of ± .5 - 1 degrees. Linear actuation would be less susceptible to backlash.
- Gear reduction for the small increment of motion would be difficult to implement.
- Sufficient electrical isolation could be a problem since this approach would occupy more space.

Pivot System - Positive Aspects

- A Linear motion mechanism could be packaged such that the thread systems could be less susceptible to backlash problems.
- Reduction in equivalent thread lead may be achieved by using a differential screw concept which has different internal and external thread profiles.
- Space required for mechanism actuation would minimize encroachment into space around current system.
- Both pivot and *z* motion are readily generated with this scheme.
- Could use parts of existing hardware to minimize overall cost.

Pivot System - Negative Aspects

- Generation of pure x and y motion at the anode-puller interface requires coupled motion of the three legs.
- The pivot point is not stationary.
- An overall mathematical model would be required for generating motion.
- Scheme is not easily configured for displacement feedback transducers.

The pivoting system was chosen over the gimbal system due to the drawbacks of space requirements, backlash, and difficulty in generating z motion with the gimbal system. Further consideration of the pivoting system was required, however, to determine how to best implement the mechanism actuation for changing the effective leg lengths.

Actuation Scheme for Pivoting System

As shown in Figure 5, page 8, the current implementation of the pivoting scheme uses a threaded stud, a differential screw with internal and external threads, and a threaded plate. The thread interface between the outer threads of the differential screw and the threaded plate serves both to displace the plate along the turning screw threads and to provide a bearing surface for the upper plate. As described, alternately two of the three screws are held fixed while the third screw is driven. A pivoting motion is generated. Displacement of the upper plate relative to the lower plate is possible due to clearances in the mating threads between the differential screw and the threaded plate. The internal threads of the plate, for example, are a standard 3/8-32 thread profile. The minor diameter of these threads is 3/8 = .375 in. The mating threads of the differential screw outer diameter, however, are a custom .369-32 profile. The diametral difference between the threaded plate and the differential screw is [.375 - .369 = .006 in (.152 mm)]. As pivot angles increase, thread clearance is consumed and a binding condition is eventually

reached as thread friction produces a resisting force (and moment) greater than the drive components can withstand. Analytical prediction of the absolute position of the sliding thread surfaces (and thus the orientation of the pivoting plate) is a formidable task. These two issues provided sufficient incentive for investigating an alternative scheme.

An important simplifying element of the analysis was to specify that the three legs would always remain orthogonal to the pivoting upper plate. This requirement forced the separation of power transmission into two tasks: (1) linear displacement along screw threads; and (2) orthogonality between the plate and legs. A differential lead screw scheme can provide high-precision linear actuation of the upper plate. A precision-ground driveshaft mating with a bronze bushing can maintain orthogonality. Figure 10 depicts the conceptual drive system.

A final assumption for the conceptual design was to eliminate axial rotation about the shared centerline of the upper plate relative to the lower plate. Introduction of axial rotation is inconsistent with generating pure x and y motion in a plane at the anode-puller interface. Constraint of one of the three legs in a radial groove would prevent this relative rotation. Figure 11 shows a schematic representation of the conceptual design approach.

Mathematical Model

With a kinematically non-redundant conceptual design chosen, governing mathematics will be developed to describe system motion. From the "Statement of Problem" of chapter 1, the required inputs for the system are *xyz* values at the anode-puller interface. These values are in turn used to calculate the final leg lengths that will produce the desired displacement of the anode. Conversion of the user inputs into corresponding leg lengths is the essence of the math model.

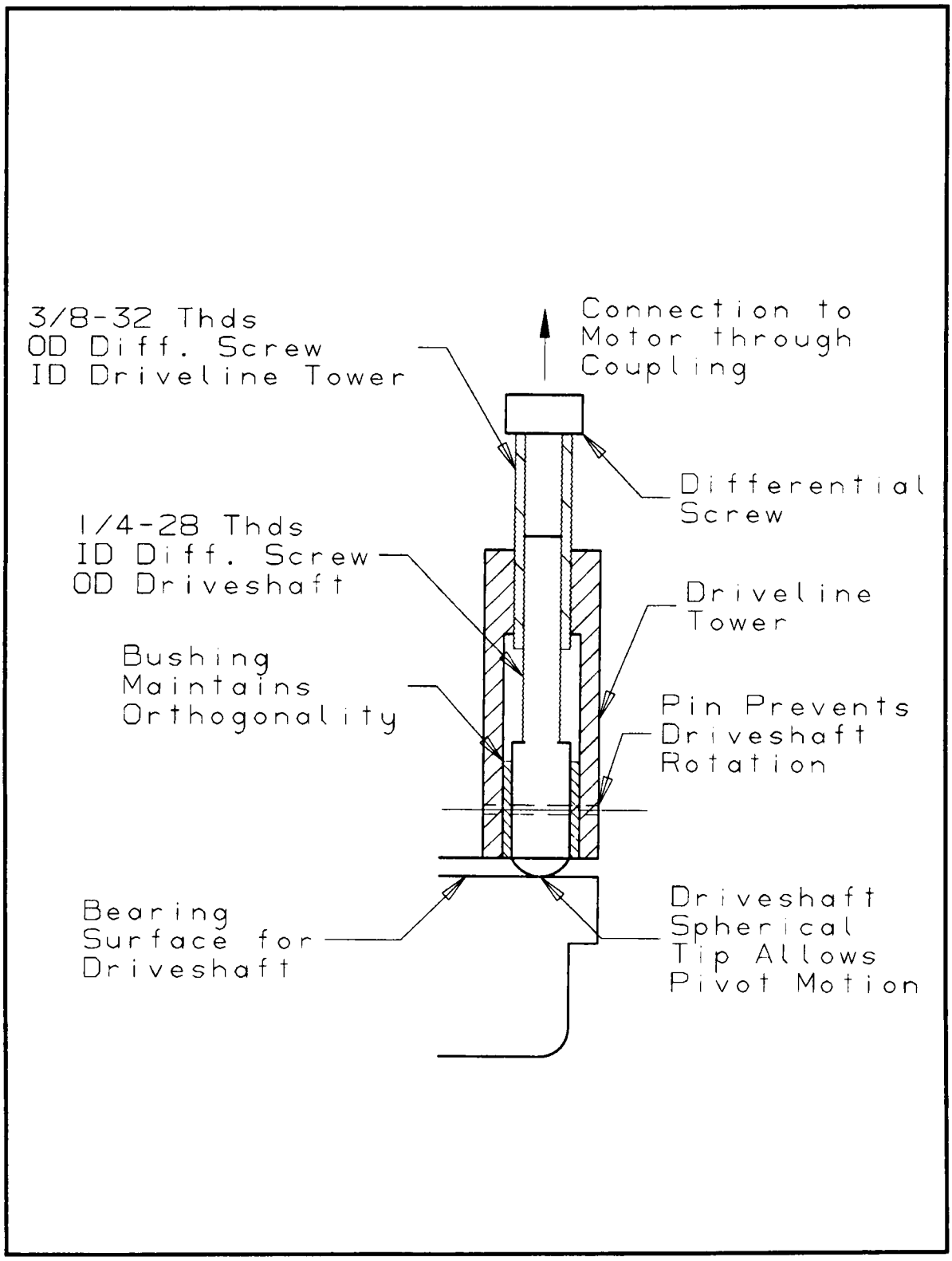


Figure 10
Conceptual Drive Mechanism

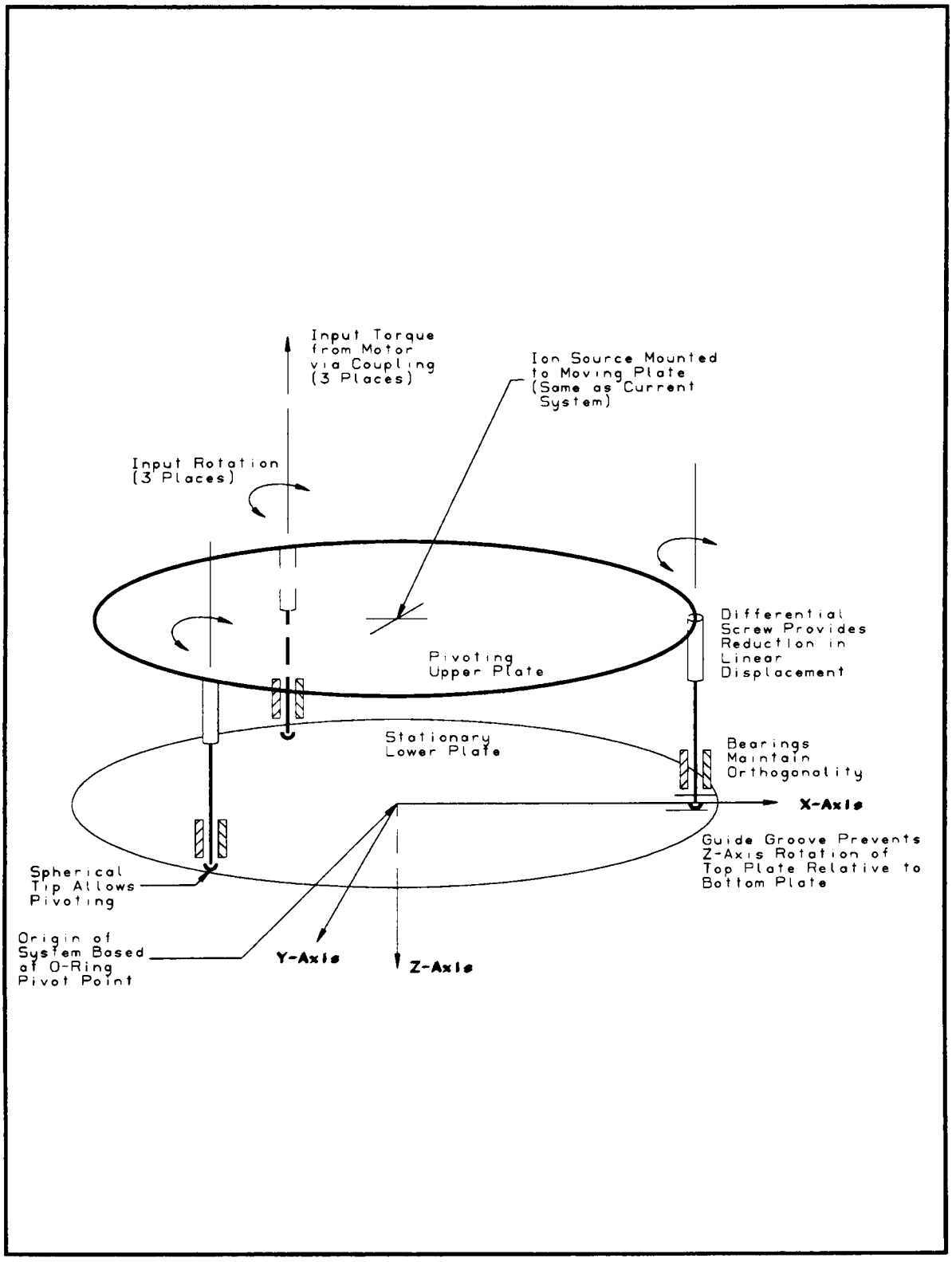


Figure 11
Schematic Representation of Conceptual Drive System Approach

Coordinate Transformations

User-specified *xyz* coordinates can be converted into two angles and a vertical displacement, or αβz, coordinates. The center point of the o-ring is assumed to act as a system pivot point and is a natural choice for the origin of the mathematical model. Figure 12 shows the orientation of the *xyz* triad based at the pivot point relative to the ion source geometry. Definition of α and β are also shown in Figure 12.

Three of potentially four degrees of freedom (DOF's) could be described by (1) rotation about the *x* axis, (2) rotation about the *y* axis, and (3) pure *z* motion. A fourth DOF could be generated by rotation about the *z*-axis. As discussed during the conceptual design definition, a groove could be added to the stationary lower plate to prevent rotation about the *z*-axis.

Without the assumption of an initial coplanar condition, *z*-axis rotation could be induced about a theoretical line extending from the non-coplanar origin to the leg constrained to motion in the groove. Figure 13 presents a schematic illustration of this assumption. The final form of the math model does not require a coplanar condition.

In general, the center point of the o-ring will be displaced from this coplanar position as a result of a system move due to clearances between the o-ring and its groove. All subsequent calculations treat each move as starting from the coplanar condition and subtracting the current value from each calculated value for incremental changes.

The main component of the math model is the description of the position of the upper plate in terms of the original *xyz* coordinate system after it has been rotated through β and α angles. Coordinate transformations accomplish this task by assigning a new coordinate system as a result of β rotations and then as a result of α rotations. Figure 14 shows the progression from the *xyz* system to the *x'y'z'* due to β rotations about the *y*-axis and progression to the *x"y"z"* coordinate system as a result of α rotations about the *x'* axis.

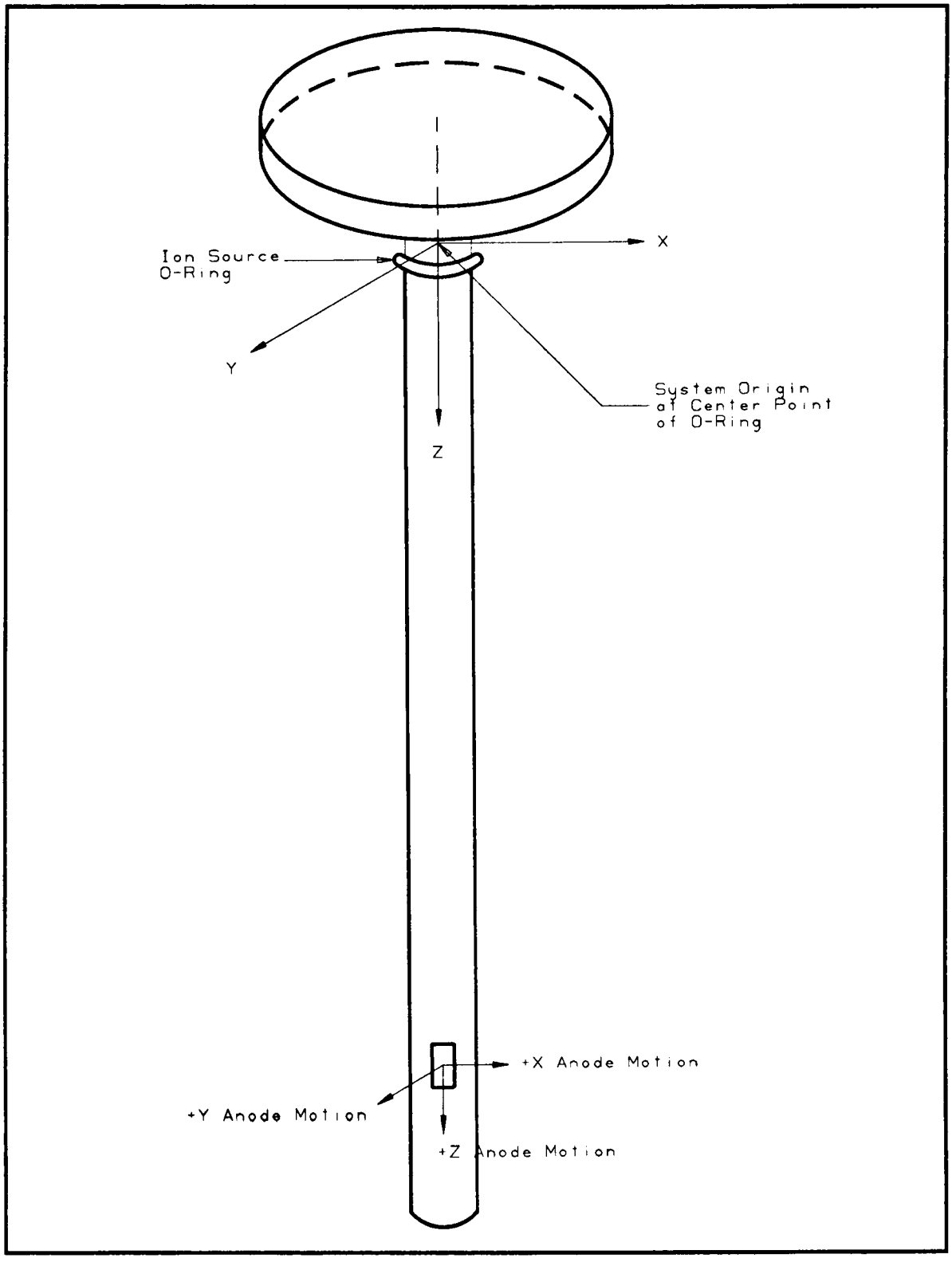


Figure 12
Relationship of Anode Motion to System Origin

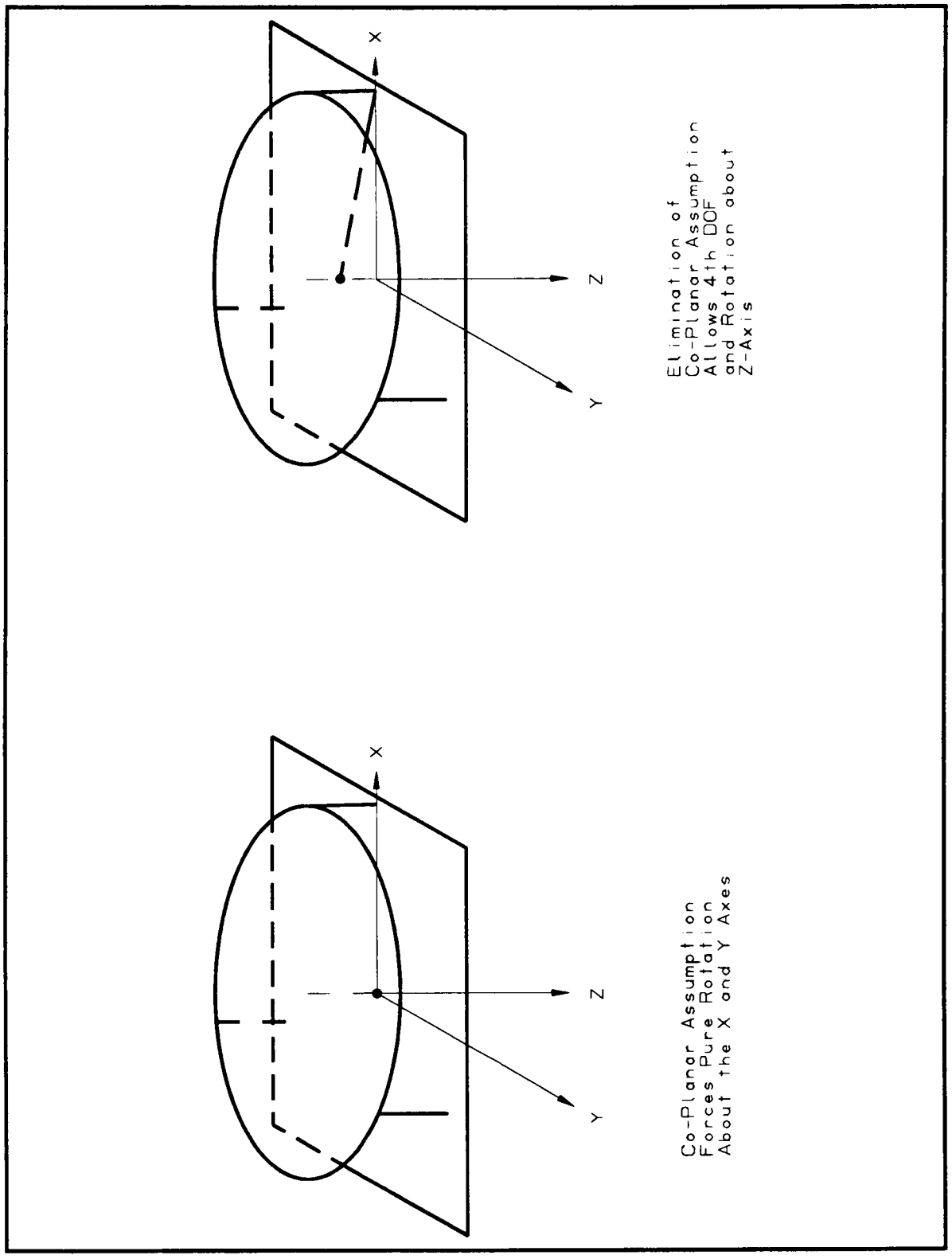


Figure 13
Schematic Representation of Initial Co-Planar Assumption

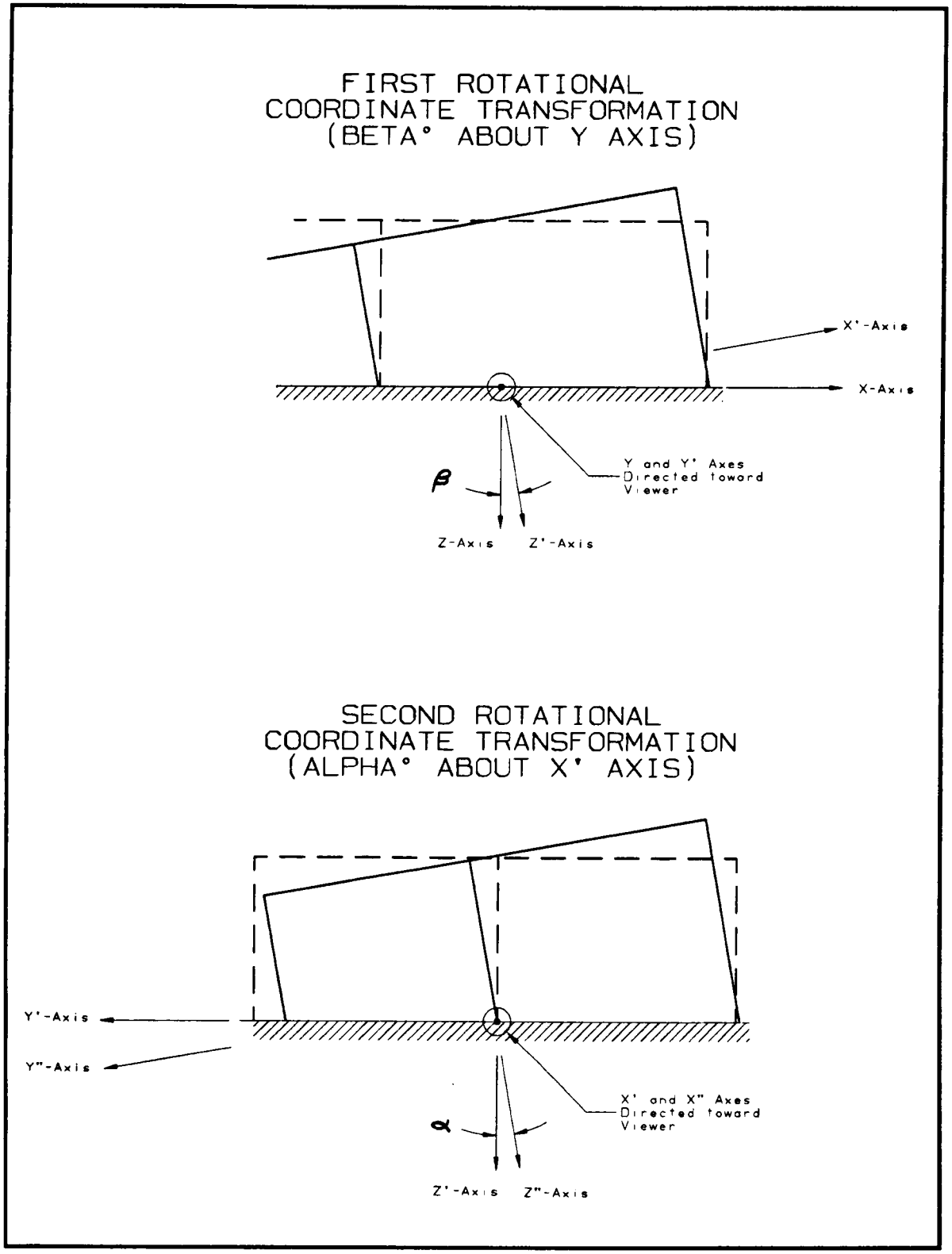


Figure 14
Coordinate Transformations Due to Rotations: β about *y*-axis, α about *x'* axis

Figure 15 extends the double-primed coordinate system to four more coordinate systems pertinent to this analysis.

The desired result from defining the six coordinate systems is to develop expressions which describe the displaced geometry in terms of the original geometry through a conversion matrix. The general development of the math model is as follows:

$$xyz \rightarrow x'y'z' \rightarrow x''y''z'' \rightarrow x'''y'''z''' \rightarrow \begin{cases} x1y1z1 \\ x2y2z2 \\ x3y3z3 \end{cases}$$

or,

$$xyz = \begin{bmatrix} \text{Conversion} \\ \text{Matrix} \end{bmatrix} \bullet f \begin{pmatrix} x1y1z1 \\ x2y2z2 \\ x3y3z3 \end{pmatrix}$$

From Figure 14, it may be seen that, for rotation about the y-axis through β, the original coordinates xyz may be expressed in terms of the new coordinates $x'y'z'$ as:

$$x = x'\cos(\beta) + z'\sin(\beta) \qquad (2-1)$$
$$y = y' \qquad (2-2)$$
$$z = -x'\sin(\beta) + z'\cos(\beta) \qquad (2-3)$$

Again from Figure 14, for rotation about the x'-axis through angle α, the $x'y'z'$ coordinates may be expressed in terms of new coordinates $x''y''z''$ as:

$$x' = x'' \qquad (2-4)$$
$$y' = y''\cos(\alpha) - z''\sin(\alpha) \qquad (2-5)$$
$$z' = y''(\sin\alpha) + z''\cos(\alpha) \qquad (2-6)$$

Stated in terms of matrices,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \qquad (2-7)$$

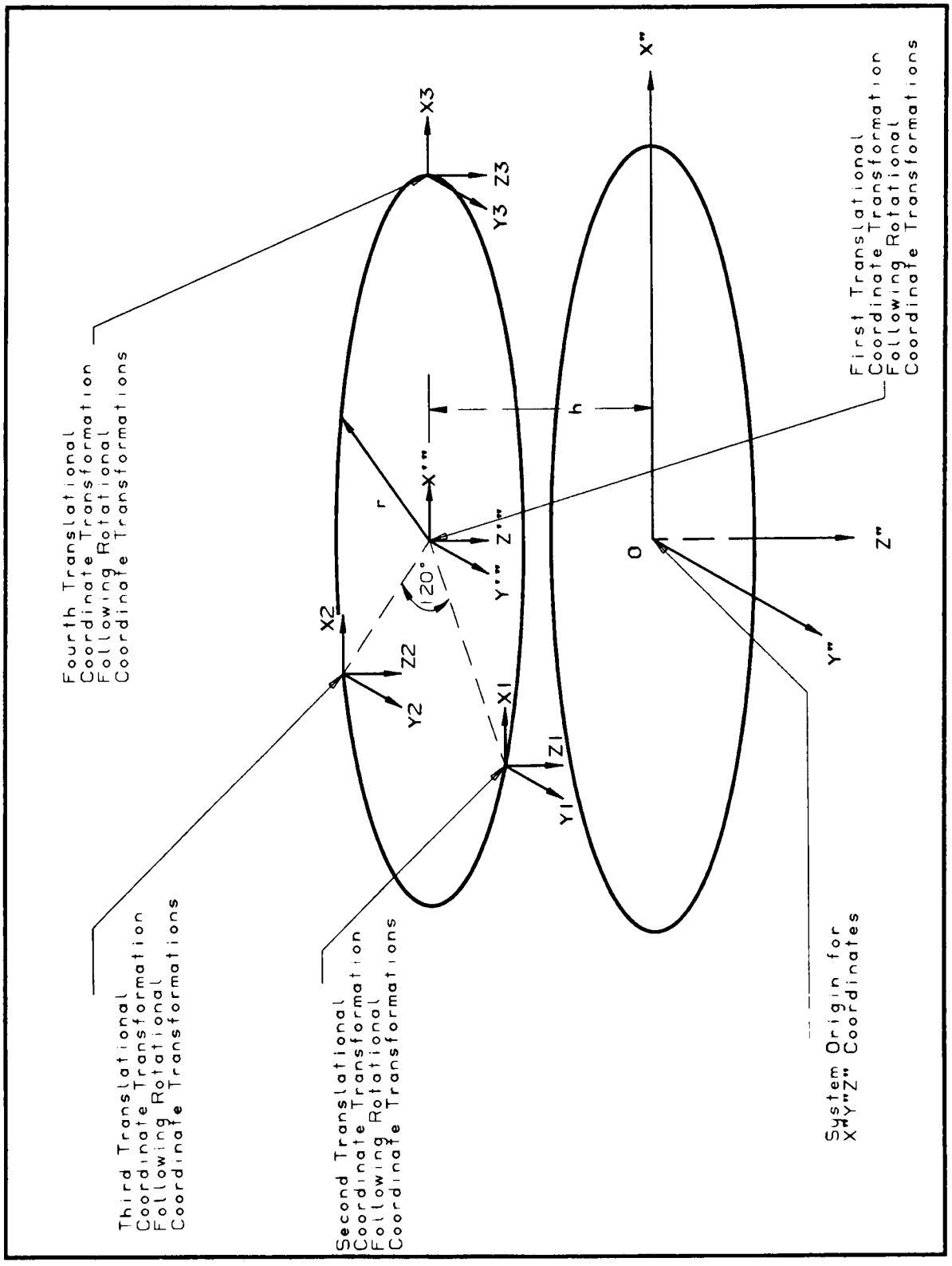


Figure 15
Coordinate Transformations Due to Translations

or,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [A1] \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad \text{where } [A1] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \tag{2-8}$$

Similarly,

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} \tag{2-9}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = [A2] \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} \quad \text{where } [A2] = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{2-10}$$

Relating the *xyz* coordinates to the *x″y″z″* coordinates:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [A1][A2] \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} \tag{2-11}$$

where $[A1][A2]$ can be renamed as a combined matrix $[A]$ is shown in its expanded form below:

$$[A1][A2] = [A] = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ sin(\alpha)sin(\beta) & cos(\alpha) & -sin(\alpha)cos(\beta) \\ -cos(\alpha)sin(\beta) & sin(\alpha) & cos(\alpha)cos(\beta) \end{bmatrix} \tag{2-12}$$

Further definition of coordinates from the ()″ system to the ()‴ system is possible by recognizing that the center point of the pivoting upper plate shown in Figure 15 is located along the *z″*-axis a distance *h* from the origin. Defining a ()‴ coordinate system:

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} x''' \\ y''' \\ z''' + h \end{bmatrix} \tag{2 - 13}$$

which may be stated in terms of the original coordinates as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [A] \begin{bmatrix} x''' \\ y''' \\ z''' + h \end{bmatrix} \tag{2 - 14}$$

The final three coordinate transformations relate the tops of legs *1*, *2*, and *3* (with their corresponding coordinate systems *x1y1z1*, *x2y2z2*, and *x3y3z3*) back to the *x"y"z"* coordinates as shown in Figure 15. The relationship between the *1,2,3* coordinate systems and the ()''' system may be stated as follows:

$$\begin{array}{lll} x1 - \frac{r}{2} = x''' & x2 - \frac{r}{2} = x''' & x3 + r = x''' \\ y1 + \frac{\sqrt{3}}{2} r = y''' & y2 - \frac{\sqrt{3}}{2} r = y''' & y3 = y''' \\ z1 = z''' & z2 = z''' & z3 = z''' \end{array} \tag{2 - 15}$$

The relationship of the *1,2,3* coordinate systems to the *xyz* coordinate system may be stated as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [A] \begin{bmatrix} x1 - \frac{r}{2} \\ y1 + \frac{\sqrt{3}}{2} r \\ z1 + h \end{bmatrix} \quad \begin{bmatrix} x \\ y \\ z \end{bmatrix} = [A] \begin{bmatrix} x2 - \frac{r}{2} \\ y2 - \frac{\sqrt{3}}{2} r \\ z2 + h \end{bmatrix} \quad \begin{bmatrix} x \\ y \\ z \end{bmatrix} = [A] \begin{bmatrix} x3 + r \\ y3 \\ z3 + h \end{bmatrix} \tag{2 - 16}$$

Calculation of Leg Lengths

As stated at the beginning of this development, the overall goal is to find an expression of the required leg lengths for user-specified *x*, *y*, and *z* inputs. The preceding

mathematical development, culminating in the three elements of equation (2-16), provides the means for making such calculations.

Consider an observer located at the top of leg 1, looking down the $z1$-axis toward the x-y plane. At this location, the origin of the $x1y1z1$ coordinate system, $x1$ and $y1$ are equal to zero. Recall that the goal of this development is to state the new length of leg 1 in terms of the user-prescribed input values, xyz, which are converted into β and α values. $z1$ is the variable that represents the new leg length. In the left-end expression in equation (2-16), x and y are the coordinates of the end of the leg that rests on the z=0 plane.

To calculate $z1$, first calculate the values of the $[A]$ array, given by equation (2-12) due to β and α rotations. The resulting right-hand side of the left-end element of equation (2-16) will have one unknown, $z1$. The left-hand side of equation (2-16) still has three unknowns, x, y, and z. By setting z equal to zero, the left-hand side is reduced to only two unknowns. Solution of leg length 1 is summarized as a three-equation, three-unknown problem. Shown in an equation format:

$$x = -A_{11}\frac{r}{2} + A_{12}\frac{\sqrt{3}}{2}r + A_{13}(z1 + h) \qquad (2\text{-}17)$$

$$y = -A_{21}\frac{r}{2} + A_{22}\frac{\sqrt{3}}{2}r + A_{23}(z1 + h) \qquad (2\text{-}18)$$

$$0 = -A_{31}\frac{r}{2} + A_{32}\frac{\sqrt{3}}{2}r + A_{33}(z1 + h) \qquad (2\text{-}19)$$

where A_{xx} values are elements of $[A]$, r is the radius of the circle containing the three leg locations, and h is the initial distance between the upper and lower plates. Solving for $z1$ in the equation (2-19):

$$z1 = \frac{A_{31}\frac{r}{2} - A_{32}\frac{\sqrt{3}}{2}r}{A_{33}} - h \qquad (2\text{-}20)$$

Values for x and y may be obtained by substituting $z1$ back into equations (2-17) and (2-18). Solutions for $z2$ and $z3$ were obtained in a similar manner. A spreadsheet program was used to calculate values for $z1$, $z2$ and $z3$ for different β and α values. Figure 16 shows a typical output from the spreadsheet analysis.

Verification of the Math Model

In order to independently verify the mathematical model, IDEAS® CAD system was used to generate a solids model of the ion source assembly geometry. Figure 17 shows the undisplaced solids model of the ion source assembly. An upper plate was rotated through some β and α values relative to a fixed lower plate. After rotation, the gap between the upper and lower plate was measured at three locations, 120° apart. The method of measurement replicates the dynamics of the conceptual model by projecting a line through the leg location along a path that is orthogonal to the displaced upper plate. Measurement of the length of the projected line from the lower surface of the upper plate to the intersection with the upper surface of the lower plate yields the leg length. Figure 18 shows the overall ion source in a displaced position.

Table 1 shows a comparison of values between the mathematical model and the CAD model for four different angular displacement combinations. The close agreement between these values satisfactorily validates the mathematical model.

Other Contributions to Model

In order to completely describe the motion of the conceptual mechanism, two other contributions to system displacement must be considered. First, up to this point in the model only the effects of β and α have been considered. As mentioned previously, the user may prescribe motion in any or all of the x, y, and z axes at the anode-puller interface. Pure z motion may be accomplished by displacing all three legs an equal amount.

Figure 16
Typical Output from Spreadsheet Implementation of Mathematical Model

Ion Source Positioning System Math Model:						
No aperture calc's, no z-compensation.						
(Unless specified, all values in mm)						
alpha	**beta**					
(deg)	(deg)					**Absolute**
0	0.4		(mm)	(mm)	"Leg Length"	**Value**
			x	**y**	**z1**	**Delta**
pi			-24.8164	42.9822264	3.636750219	0.17325
3.141592654						
			x	**y**	**z2**	
h	**r**	**rad 3/2**	-24.8164	-42.9822264	3.636750219	0.17325
-3.81	49.6316	0.866				
			x	**y**	**z3**	
			49.63281	0	4.156499563	0.3465

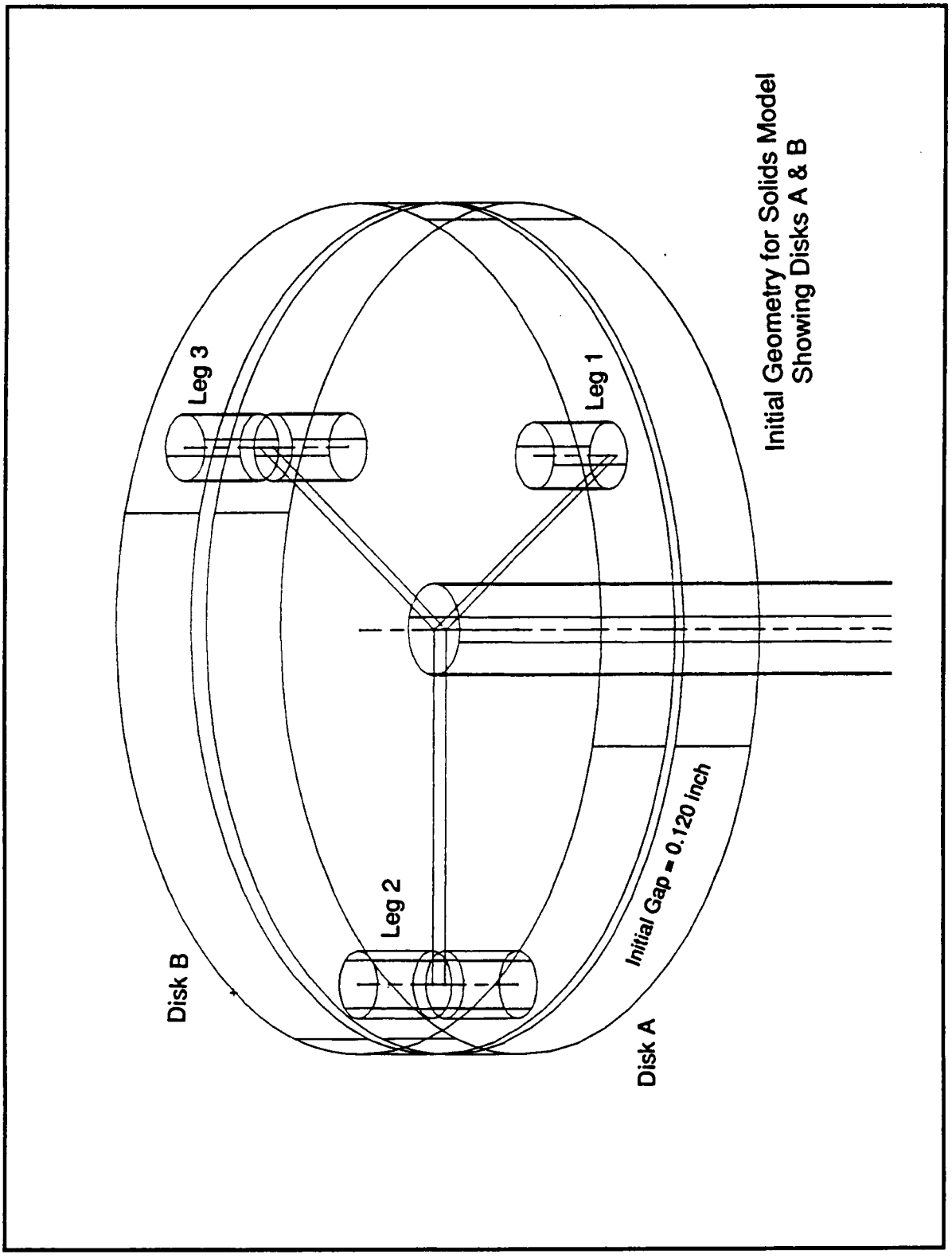


Figure 17
Solids Model for Ion Source in Undisplaced Position

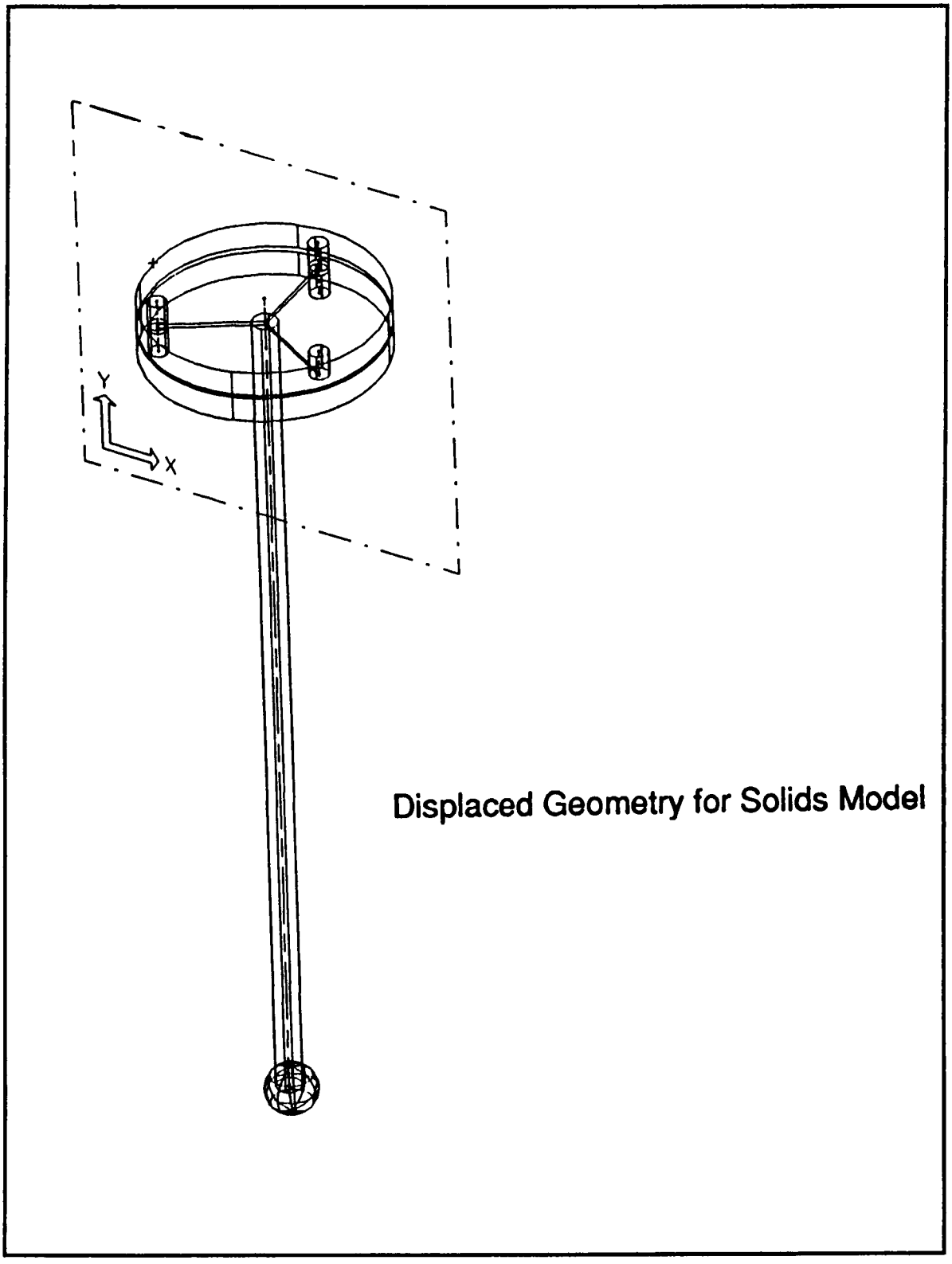


Figure 18
Solids Model for Ion Source in Displaced Position

Table 1
Verification Data from 3D CAD for Ion Source Mathematical Model

			Leg Length	Leg Length
α	β	Location	Math Model	CAD Verification
(deg)	(deg)		(mm)	(mm)
4	0	Leg1	0.804390	0.804390
		Leg2	6.815610	6.815610
		Leg3	3.810000	3.810000
0	4	Leg1	2.074710	2.074710
		Leg2	2.074710	2.074710
		Leg3	7.280580	7.280580
3	3	Leg1	0.253765	0.253765
		Leg2	4.765153	4.765153
		Leg3	6.411082	6.411082
0.4	0.4	Leg1	3.336665	3.336665
		Leg2	3.936835	3.936835
		Leg3	4.156500	4.156500

Figure 19 shows schematically how displacement of some z amount after rotation can cause unwanted horizontal displacement of the bottom of the legs. Horizontal displacement is undesirable only in the leg constrained by the guide groove since it allows displacement only along the groove. Second, as the upper plate of the mechanism is rotated through β and α angles, two small z components (H_x and H_y) are generated. These two values represent error in the z axis and should be included as z-values along with the user-prescribed z-motion discussed above. By including all user-prescribed and motion-induced z values in equation (2-19), ($z \neq 0$) prior to multiplication by the rotational coordinate transformation matrix, final calculations for leg length and subsequent moves produce no unwanted horizontal motion. Figure 20 shows a schematic representation of the two z-axis induced errors, H_x and H_y, which are defined below.

$$H_x = x \cdot tan\left(\frac{\beta}{2}\right) \qquad (2-21)$$

$$H_y = y \cdot tan\left(\frac{\alpha}{2}\right) \qquad (2-22)$$

Relating Model Variables to Operator Inputs

Some final calculations relate x, y, and z values to quantities familiar to a cyclotron operator. The actual alignment process positions a 0.5 x 5.0 mm aperture on the anode with a 1.5 x 7.5 mm aperture on the puller. The x and y axes were defined such that the "Gap" between these two apertures falls along the y-axis, while the in-plane "X-Aperture" alignment orientation falls along the x-axis. Vertical alignment of these two apertures, "Z-Aperture", falls along the z-axis. The following are physical analogs to x, y, and z values:

- x-axis: x alignment of the anode aperture with the puller aperture;
- y-axis: gap between the anode aperture and puller aperture along the y axis;
- z-axis: z alignment of the anode aperture with the puller aperture.

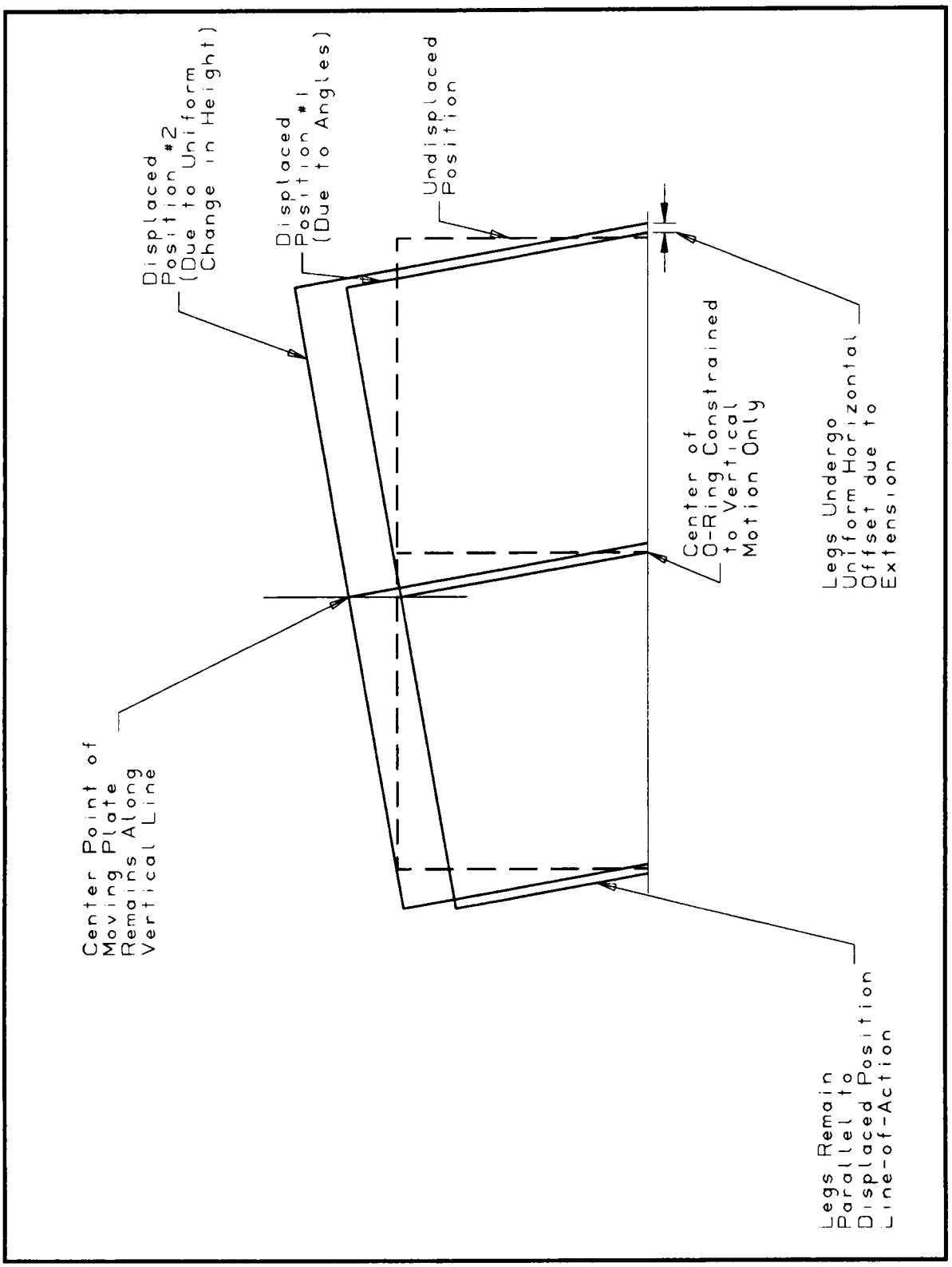


Figure 19
Effects of Z-Axis Motion on Leg Locations

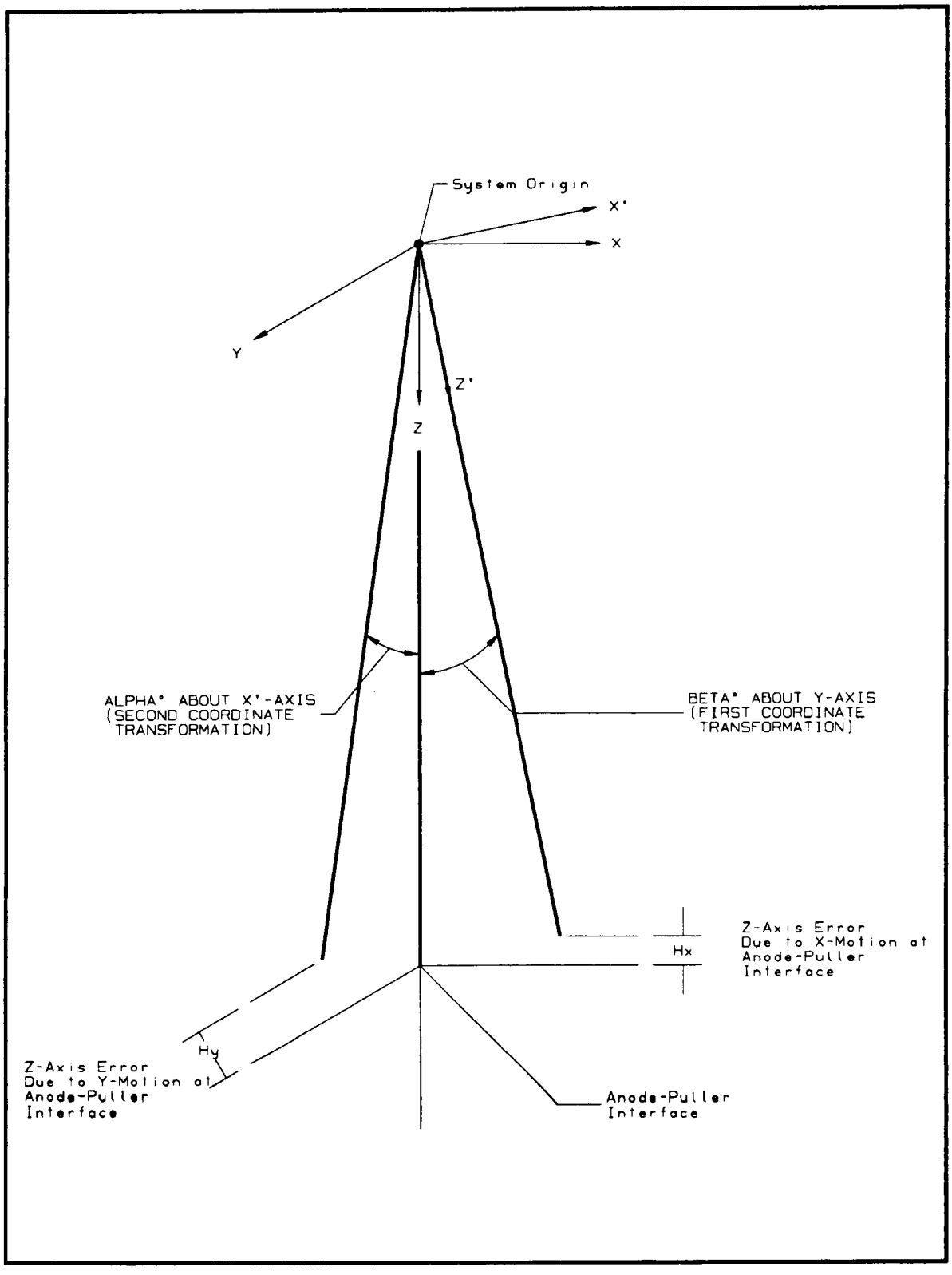


Figure 20
Definition of Errors Hx and Hy Due to Z-Axis Compensation

Figure 21 shows a top view of the anode-puller interface with labels showing the physical representation of the alignment values corresponding to the x and y axes. This *x-y* plane is offset from the origin along the *z*-axis by 424.71 mm (16.721 in). The offset values shown in Figure 21 are included in the software code.

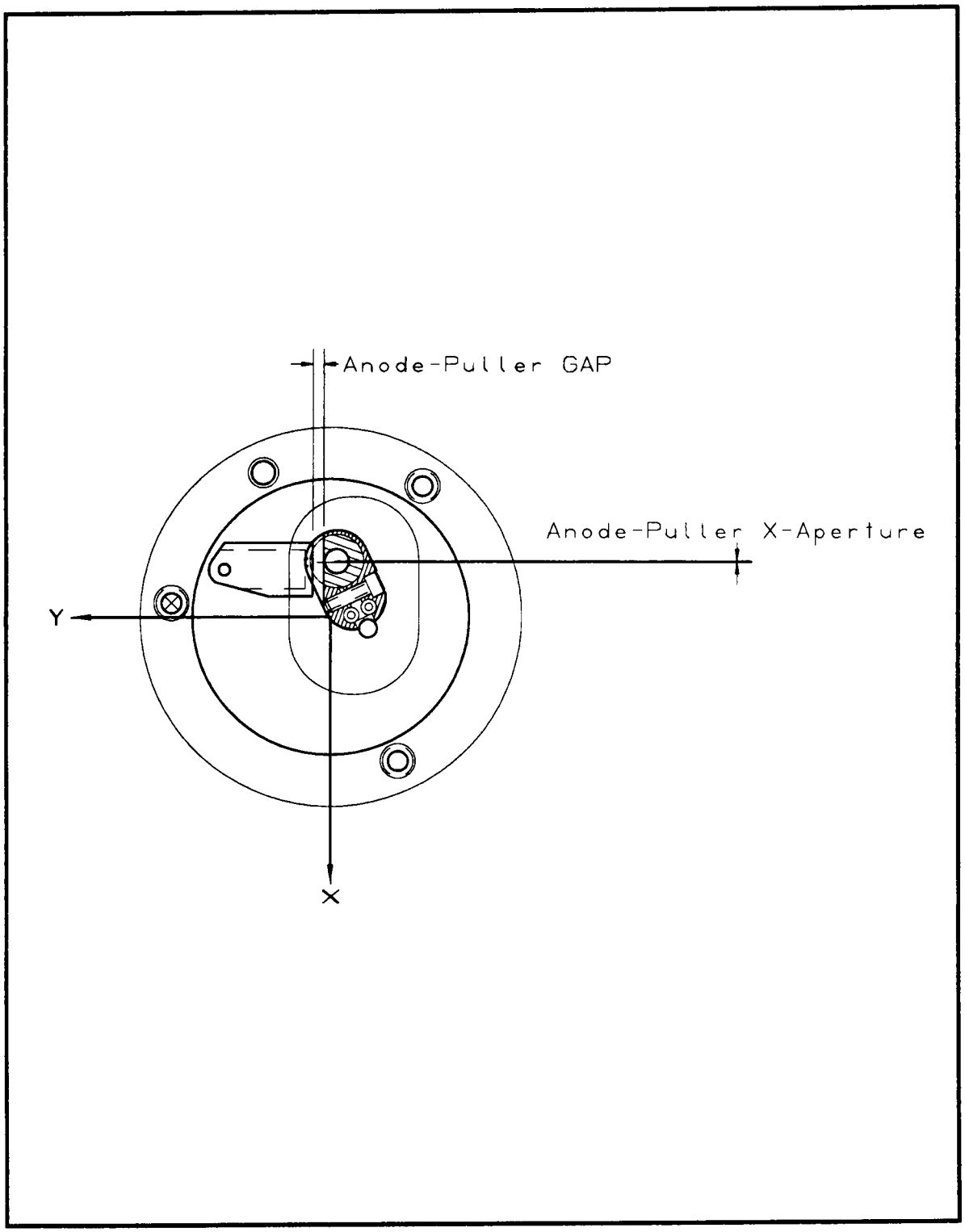


Figure 21
Top View of Anode-Puller Interface Showing Correlation of Alignment Values with x and y Axes.

CHAPTER THREE

DETAILED SYSTEM DESIGN

Governing Guidelines

The governing guidelines for the mechanical design configuration ensure that the three analytical assumptions are maintained and that the minimum increment of motion is achieved. Recall that the first assumption was that the three legs would always remain orthogonal to the pivoting upper plate. The second assumption was that the base of the three legs which mated with the lower plate would initially be coplanar with the o-ring. The third assumption was that the upper plate would be constrained not to rotate relative to the lower plate about the z-axis (through their centerlines). Design of the drive system for adjusting the leg lengths focused on generating, with margin, a minimum increment of motion of $\pm$ 0.1 mm at the anode-puller interface.

Other practical considerations that governed the design approach included designing features for homing the mechanism and creating an assembly that could easily be retrofitted into existing hardware. The scope of this effort was to generate a prototype mechanism that demonstrated the ability to very accurately position the ion source from a remote actuation point. It was important to minimize the overall cost.

Maintaining Key Assumptions

Keeping Legs Orthogonal to Pivoting Upper Plate

The first key assumption was that the three legs would always remain orthogonal to the pivoting upper plate. Figure 22 shows the main components of the drive system and their relative orientation with the rotating upper plate. Observe that the upper plate

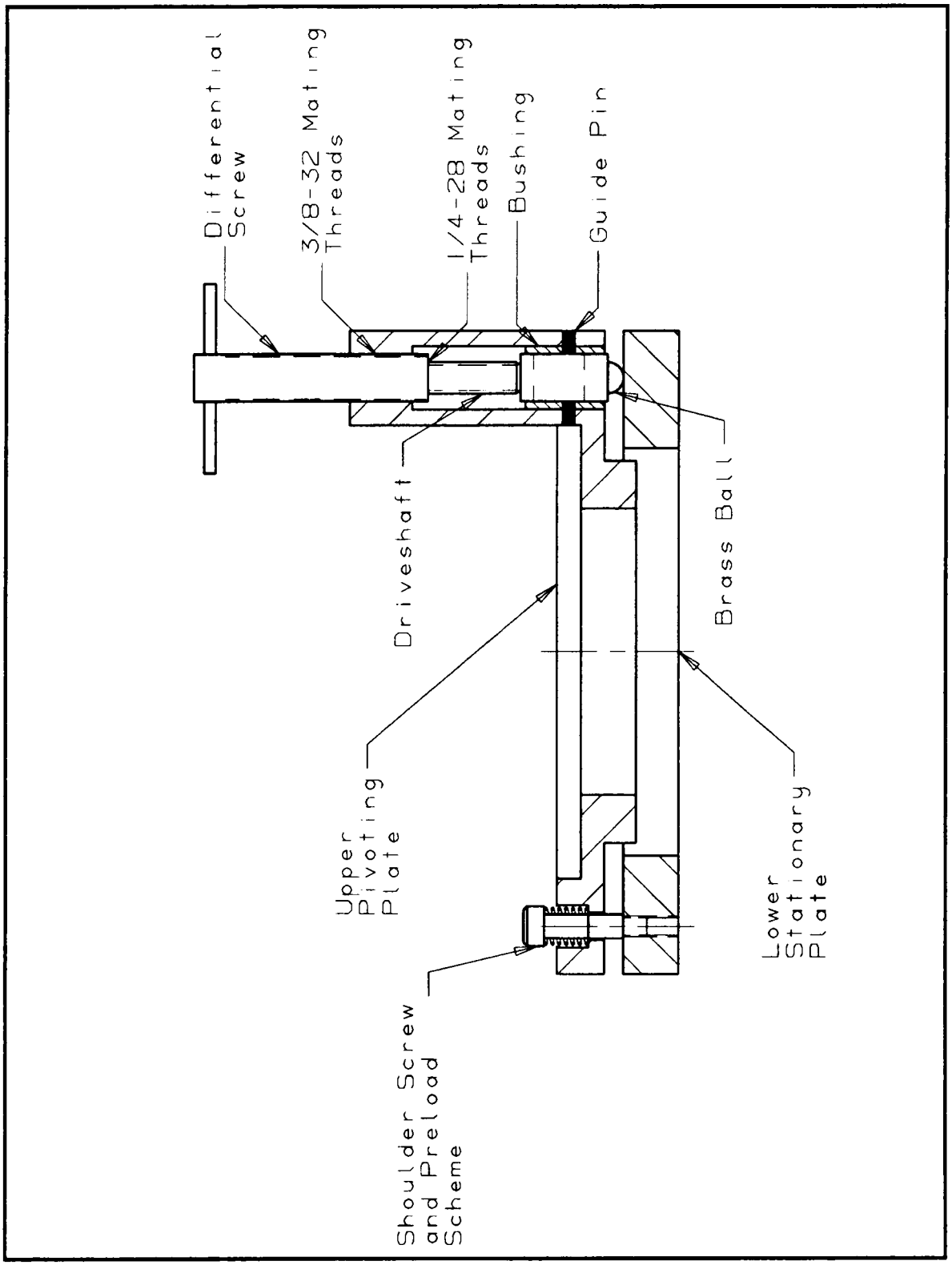


Figure 22
Main Components of Drive System

combines the two distinct tasks that are required for actuating the upper plate: threads for linear motion and a bushing to maintain orthogonality.

The two other components that are concerned with the linear displacement of the upper plate are the driveshaft and the differential screw. Notice that the driveshaft integrates several important design features. First, the 3/8" precision ground driveshaft which has a 10 μin surface finish mates with a bronze bushing to maintain orthogonal motion of the shaft relative to the pivoting plate. Second, a brass ball, pressed and bonded into the end of the driveshaft, serves as the bearing surface with the stainless steel lower plate. Third, the driveshaft is threaded on the upper end to serve as a fixed, threaded rod in the differential screw drive scheme. A slot in the driveshaft provides clearance for a pressed-in guide pin that keeps the driveshaft from spinning axially and allows it to act as a threaded stud.

Figure 23 shows an exploded view of the differential screw drive system concept. Three separate parts with two mating thread surfaces comprise this sub-system. The threaded end of the driveshaft is a 1/4-28 externally threaded rod which mates with the 1/4-28 internal threads of the differential screw. The 3/8-32 external threads of the differential screw mate with the 3/8-32 threads of each tower in the pivoting upper plate. Two opposing displacements are induced with each input clockwise revolution of the differential screw. To appreciate the motion generated by these three elements, first consider the driveshaft as a fixed, threaded rod. For each clockwise revolution, the differential screw may be considered a nut which moves down the driveshaft at a lead of (1/28 = .0357 in). Next, consider the OD threads of the differential screw and its mating part, the upper pivoting plate. Assume that the bottom of the differential screw is held against a plane but it is still allowed to rotate axially. For a given clockwise revolution, the pivoting upper plate moves up the differential screw at a lead of (1/32 = .0313 in). Combining these two motions into one event, the net displacement for a given clockwise

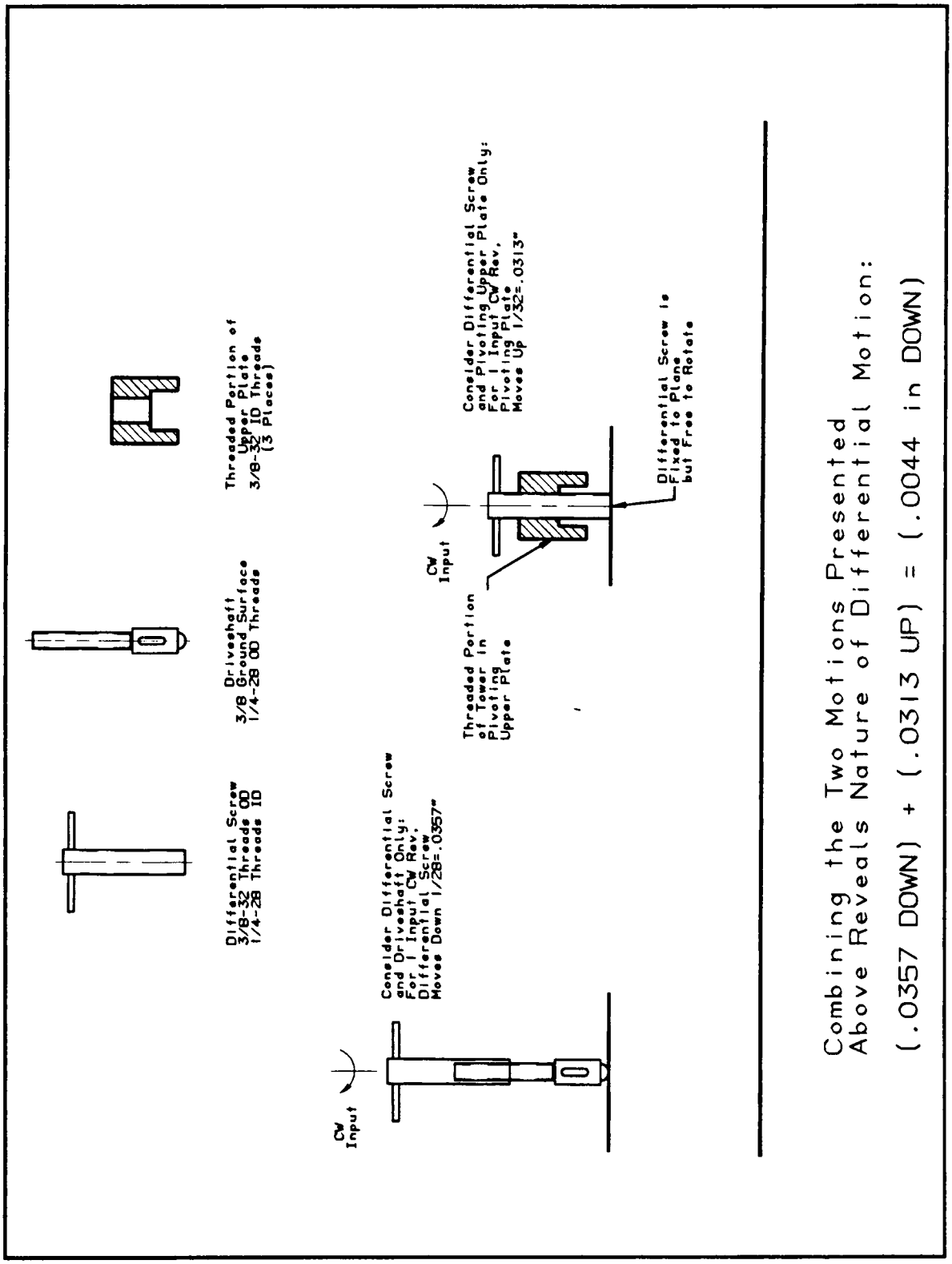


Figure 23
Detailed Presentation of Differential Screw System

revolution is (.0357 in ↓) + (.0313 in ↑) = (.00446 in ↓), .1134 mm downward displacement of the pivoting upper plate. For a counterclockwise revolution, the directions are reversed and the pivoting upper plate moves upward .1134 mm.

The final component of the drive system is an acetal coupling which is mounted onto each motor output shaft. The lower portion of the coupling is slotted so that the differential screw cross bar may nest into it. Extra clearance between the inner diameter (ID) of the coupling and the outer diameter (OD) of the differential screw supports the requirement of orthogonality between the drive system and the upper plate by allowing angular offset between the centerline of the motor shaft and the centerline of the drive system. The coupling is shown in Figure 24.

Base of Driveshaft Coplanar with O-ring

The second key assumption was that the base of the three legs which mate with the lower plate would initially be coplanar with the o-ring. Figure 25 shows how the lower plate is attached to the stationary flange of the existing hardware. Observe that the mating surface between the brass balls of the driveshafts and the lower plate is indeed coplanar with the centerline of the o-ring. Figure 26 shows the two countersunk clearance holes in the lower plate that allow the retrofit mechanism to be secured to existing hardware.

Upper Plate Constrained from Rotating Relative to Lower Plate

The third key assumption was that the upper plate would be constrained not to rotate relative to the lower plate about the common axis through their centerlines. Figure 26 shows a spherical groove oriented radially from the center of the lower plate. This groove, located along the theoretical x-axis, matches the radius of the brass ball pressed into the driveshaft and ensures that the third assumption is maintained.

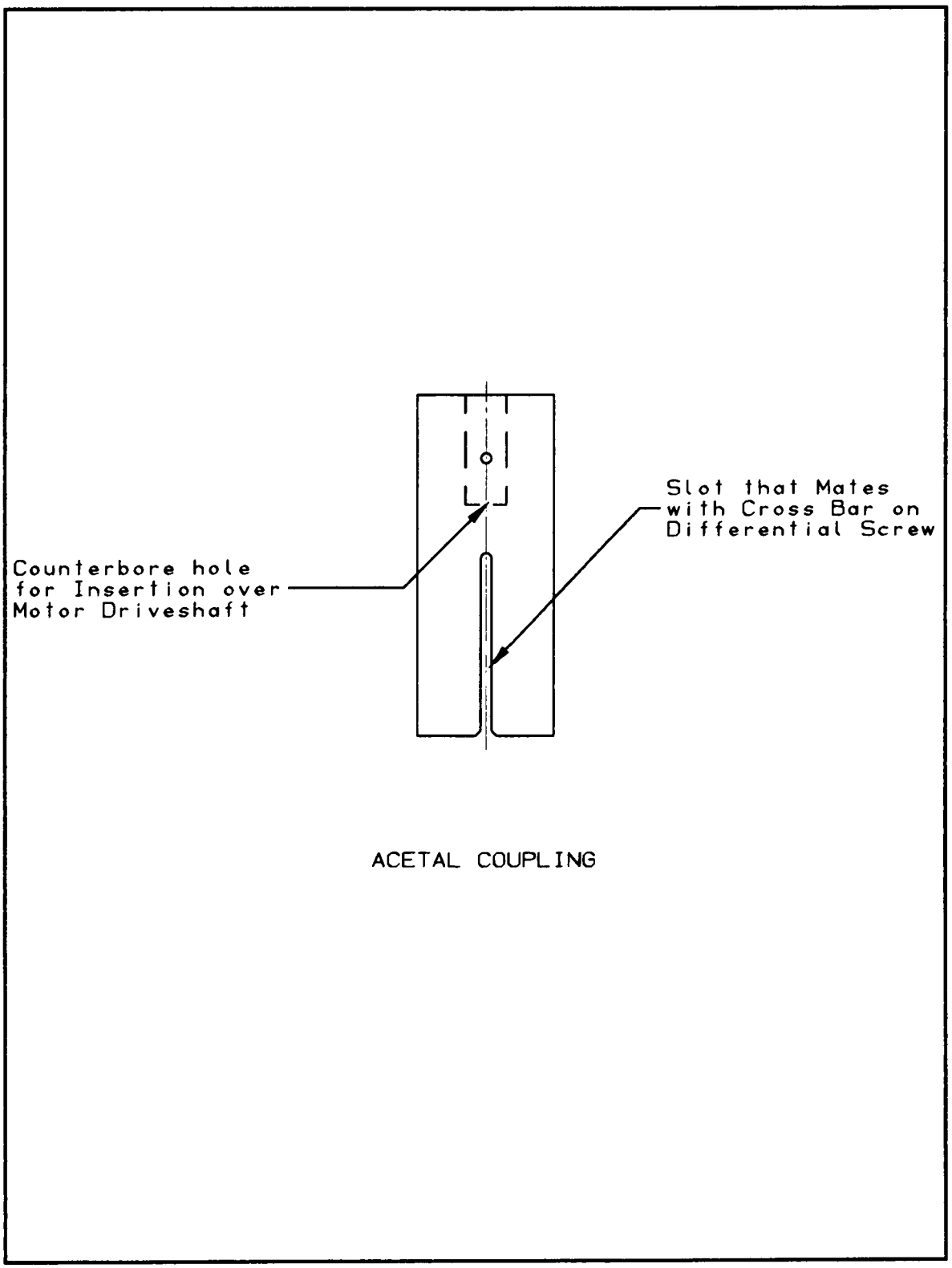


Figure 24
Acetal Driveline Coupling

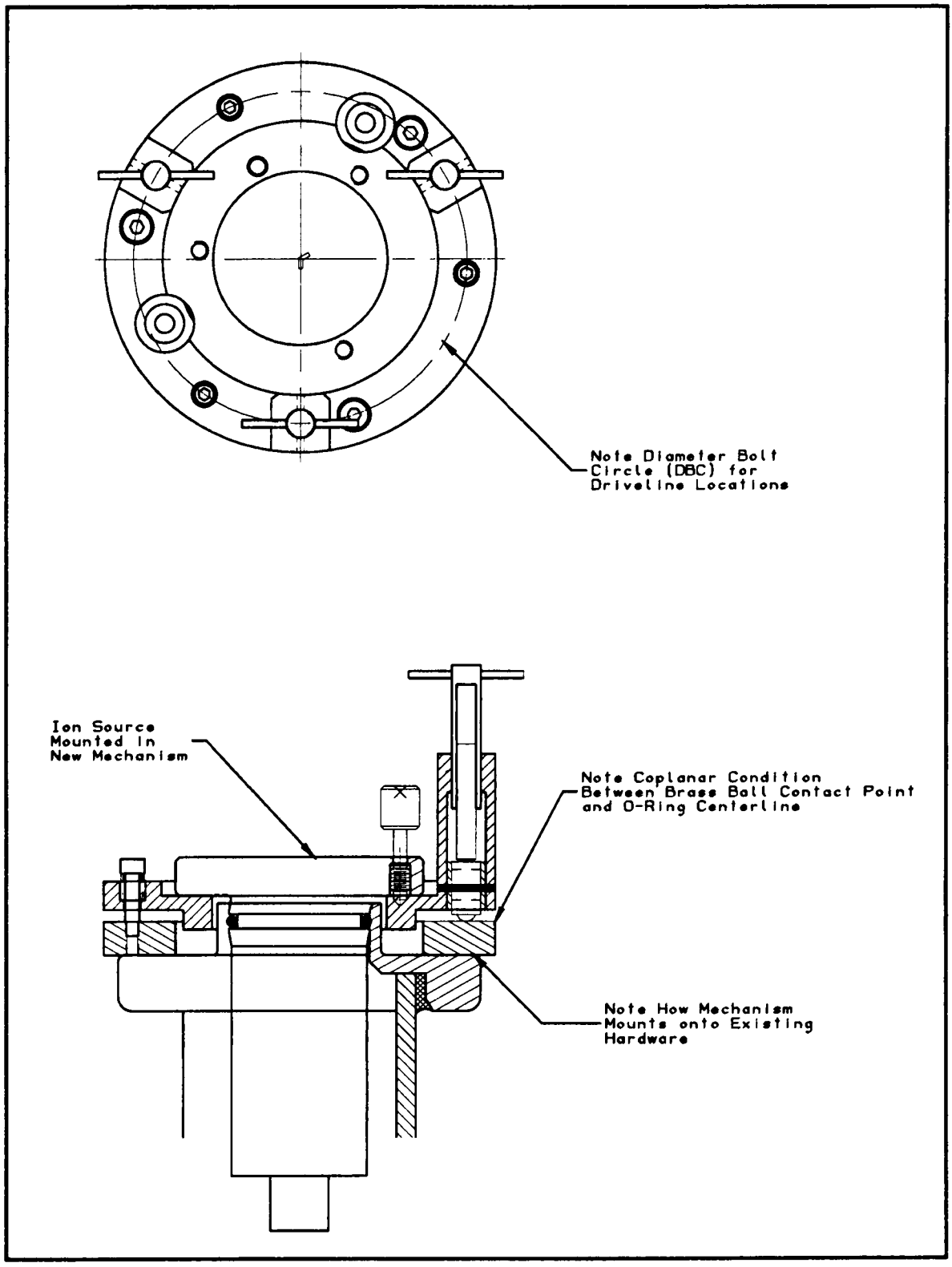


Figure 25
Side View of New Mechanism

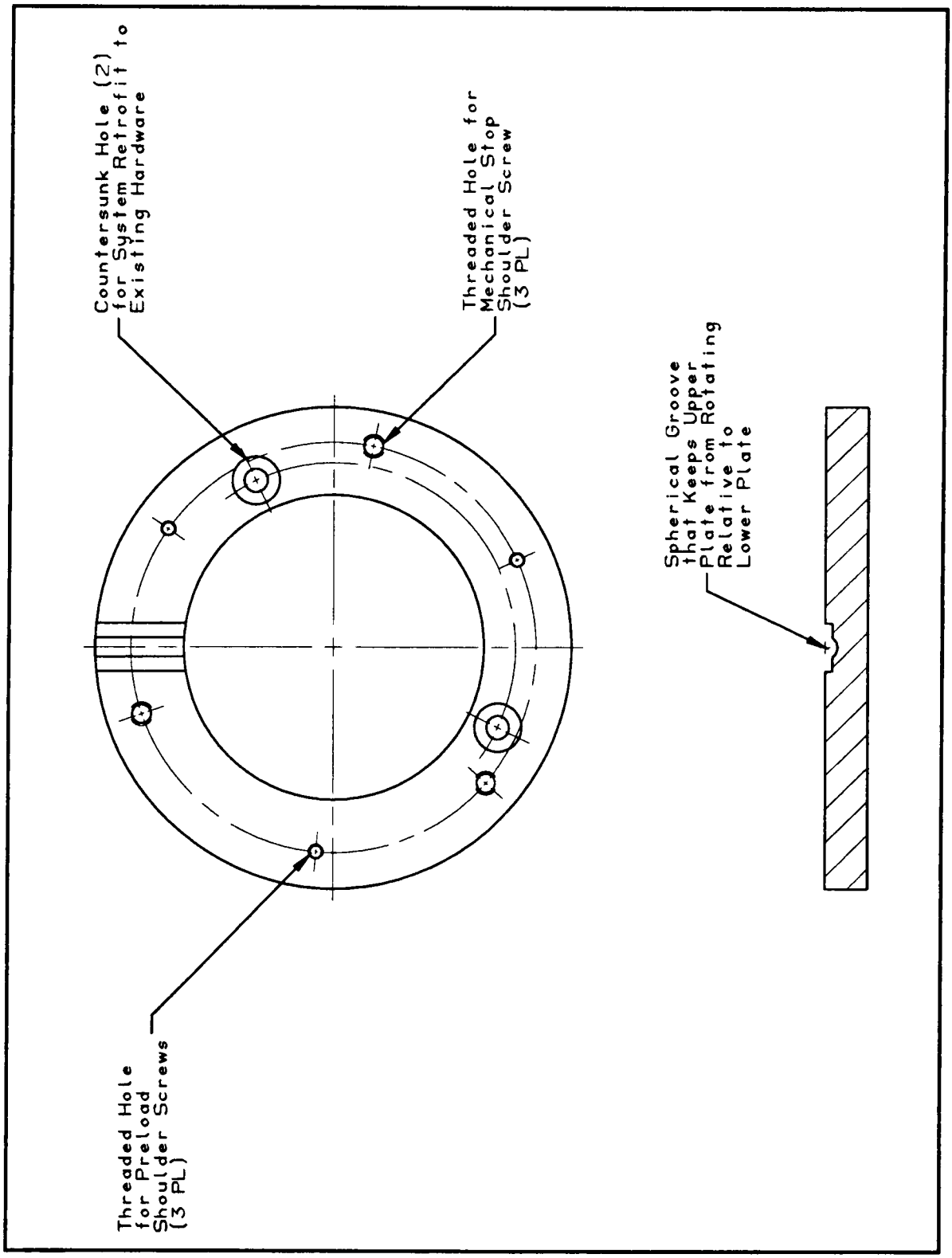


Figure 26
Lower Plate of New Mechanism

Figure 25 also shows two sets of shoulder screws that complete the layout of the mechanism. First, three equally-spaced shoulder screws are fastened into the lower plate and compress nested springs which provide system preload to eliminate drive system backlash. Second, another three shoulder screws are screwed into the lower plate and extend through the upper plate. The lower face on the heads of these screws are the mechanical stops used during system homing. By driving the upper plate into the stops a uniform gap is set between the upper and lower plates. From this position the upper plate is displaced into the starting position where the o-ring is coplanar with the lower plate.

Achieving System Minimum Increment of Motion

The final governing guideline was to ensure that the system could achieve the minimum increment of motion of ± 0.1 mm at the anode-puller interface. Movement of the anode along a single axis a value of 0.1 mm corresponds to a change in leg length of ε. The differential drive system produces δ amount of linear motion per input revolution. This minimum increment, δ, corresponds to a value of γ degrees of rotation by the input shaft. Using a 200 step/revolution stepper motor in the half-step mode (400 step/revolution), this corresponds to ϕ steps/degree. Thus, γ·ϕ yields the minimum number of steps for the minimum increment of motion. Shown in equation format:

$$\varepsilon = .00584 \text{ mm} = .00023 \text{ in}; \quad \delta = .1134 \text{ mm} = .0045 \text{ in}$$

$$\gamma = \frac{\varepsilon}{\delta} \cdot (360^\circ) = \frac{.0058}{.1134} \cdot (360^\circ) = 18^\circ \qquad (3\text{-}1)$$

$$s_{1/2} = 400 \text{ steps}$$

$$\varphi = \frac{s_{1/2}}{360^\circ} = \frac{400}{360^\circ} = 1.11 \text{ steps}/^\circ \qquad (3\text{-}2)$$

$$s_{min} = \gamma \cdot \varphi = 18^\circ \cdot 1.11 \text{ steps}/^\circ = 21 \text{ steps} \qquad (3\text{-}3)$$

This level of precision is easily achieved with the control software and motor driver cards operating in the half-step mode.

Other considerations for the design of the moving upper plate include clearance for attaching the ion source assembly into its mounting holes and clearance for the cooling lines and gas delivery lines for the plasma. Isolation of the 3/8-32 threads and the brass bushings into three separate towers provides enough clearance in the upper pivoting plate to accommodate the other system components. Appendix B contains all engineering drawings created for this project.

Drive System Calculations

Four sets of calculations undergird system design. These calculations include a system torque analysis for sizing drive components, a shear stress analysis for sizing the differential screw, a hoop stress analysis for determining an appropriate press fit for the brass ball and mating hole, and a fits analysis for the mating surfaces of the drive system. A discussion of the pertinent equations and sample results are shown in the following section. Detailed calculations are presented in Appendix A.

System Torque Analysis

The torque analysis for this system is unique in that it considers the torque both to raise and to lower a given load. The following equations[7] are used in these calculations:

For lowering a load:

$$T = \frac{(Q)(d_m)}{2}\left[\frac{\mu\pi d_m - p\cdot\cos(\alpha_n)}{\pi d_m \cos(\alpha_n) + \mu p}\right] \quad (3\text{-}4)$$

For raising a load:

$$T = \frac{(Q)(d_m)}{2}\left[\frac{\mu\pi d_m + p\cdot\cos(\alpha_n)}{\pi d_m \cos(\alpha_n) - \mu p}\right] \quad (3\text{-}5)$$

where:

T = System torque
Q = System load
d_m = mean diameter of thread
α_n = thread angle in normal plane.
p = lead of thread.
μ = coefficient of friction.

The reason that both raising and lowering a load are considered in this analysis is because of the opposing action of the differential screw. For a given input clockwise revolution of the differential screw, the 1/4-28 mating threads lower the system load while the 3/8-32 mating threads raise the load. This concept was presented in greater detail in the earlier section "Keeping Legs Orthogonal to Pivoting Upper Plate" and in Figure 23.

The system load consists of several components. First, the source assembly exerts a constant downward load due to its weight. Second, the system preload exerts a downward load dependent on the height of the upper plate above the lower plate. A maximum value for preload was assumed for the torque calculation. Third, the o-ring offers resistance to motion due to friction against its mating surface. Finally, the force of atmospheric pressure exerts a constant downward load due to the pressure differential across the o-ring and the evacuated area surrounding the ion source assembly. The following are the values used for the load, which is divided into three locations.

- Weight of ion source: 4 lb. (measured);
- Preload @ 3 places: 10 lb. (calculated - see Appendix A);
- O-ring resistance: 5 lb. (measured);
- Pressure Differential: 38 lb. (calculated - see Appendix A);

The mean diameters were chosen according to the thread profile under investigation. The coefficient of friction, μ, was chosen as 0.35 initially (brass on steel) to see worst-case torque but later used as 0.20 (brass on steel, lubricated) for use in sizing the system motors. The thread lead, p, was chosen appropriately for the thread system being analyzed. The maximum torque required for the system is 36 oz-in, assuming an unlubricated condition for the drive threads. The specified motor will deliver 150 oz-in of

torque (no-load). Since system inertias are low, no ramping of the stepper motors is necessary for system moves. Table 2 shows a summary of the torque, shear stress, and hoop stress analyses. Solution details are presented in Appendix A.

Differential Screw Shear Stress Calculation

The shear stress calculations were based on two sets of equations[6]. First, the shear stress was calculated using the standard equation:

$$\tau = \frac{T_1 r_1}{J_1} + \frac{T_2 r_2}{J_2} \qquad (3-6)$$

where τ = shear stress, T = system torque, r = radius where torque is applied, and J is the polar moment of inertia.

Next, the factor of safety was calculated assuming that failure in the differential screw would occur at the point of maximum shear stress. Since the differential screw is subjected to pure torsion with no bending, the shear stress calculated above is equal to the maximum shear stress ($\tau = \tau_{max}$). Assuming a value of σ_{yp} = 10,000 psi[5] for brass, the factor of safety may be calculated as follows from the Maximum Shear Stress theory of failure[6]:

$$FS = \frac{0.5\sigma_{yp}}{\tau_{max}} \qquad (3-7)$$

In order to ascertain the worst-case loading on the differential screw, the shear stress equation was applied both to the inside thread pattern and the outside thread pattern. The highest-load motion is seen when the load is lowered. For this case, at the maximum torque of 53 oz-in (μ = 0.20), the factor of safety for the differential screw failing due to shear loading is 3.5. This is an acceptable factor of safety value.

Table 2
Summary of Torque, Shear Stress, and Hoop Stress Analyses

System Variable	Intermediate Variables	Calculated Value (Worst-Case)	Design Value
Torque		T = 36 oz-in	T = 150 oz-in
	Load Q = 57/3 lb		
	μ = 0.35 (unlubricated)		
Shear Stress		τ_{TOT} = 1419 psi	FS = 3.5
	Torque T = 94 oz-in	$\tau_{ID\ thd}$ = 1082 psi	FS = $.5\sigma_{yp}/\tau_{TOT}$
	Mom. Inert. J = .0013 in^4	$\tau_{OD\ thd}$ = 337 psi	
	σ_{yp} = 10 ksi		
Hoop Stress		u_h = 0.0005 in	
	E_{SST} = 30 ksi	(hole increase)	
	Δ = 0.002 in	p = 66.67 psi	
	μ = 0.34	(interference pressure)	

Hoop Stress Calculation for Ball/Shaft Press Fit

To maintain a consistent driveshaft diameter where the brass ball was pressed into the stainless steel driveshaft, a stress analysis was conducted for this fit. The following equations[6] may be used first to calculate the pressure exerted on the ID of the stainless steel shaft where the ball is pressed in, and second the amount of which the OD of the driveshaft grows due to this press fit:

$$p = \frac{E\Delta}{4a}\left[1-\left(\frac{a}{b}\right)^2\right] \quad (3-8); \qquad \text{and} \qquad u_h = \frac{ap}{E_h}\left[\frac{\left[1+\left(\frac{a}{b}\right)^2\right]}{\left[1-\left(\frac{a}{b}\right)^2\right]}+\mu\right] \quad (3-9)$$

where,

p = pressure exerted onto the inner diameter of the driveshaft (calculated).
Δ = diametral interference between diameter ball and diameter hole (chosen).
a, b = inner and outer diameters, respectively (a = .25 in; b =.375 in).
E, E_h = modulus of elasticity for hole (stainless steel: E=E_h= 30,000 psi).
μ = Poisson's ratio (brass: μ = 0.34).
u_h = growth of outer diameter of outside diameter of driveshaft due to press fit.

Using a value of Δ = 0.002, the calculated pressure p = 66.7 psi, and the resulting value for u_h was 0.0005 in. This value is acceptable as an increase in the driveshaft OD.

Fits Analysis

Finally, a fits analysis for all the mating drive components was performed for appropriate callouts on the detailed fabrication drawings. Fit between the three driveshafts and mating bushings, the bushings and their mating holes, roll pins in both the drive system towers and acetal couplings, and roll pin into the top of the differential screw were all considered. Figure 27 shows a summary of the various components, their mating parts, and associated fits.

Interface: OD Bushing to ID Tower (3 PL)

Hole in Tower (in)	OD of Bushing (in)
.503, +.002 -.000	.502, +.000 -.002

Interface: ID Bushing to OD Driveshaft

ID Bushing (in)	OD Driveshaft (in)
.376, +.001 -.000 (STK)	.3747, +.0000 -.0002 (STK)

Interface: Guide Pin Hole and OD Roll Pin

ID Hole (in)	OD Roll Pin (in)
.094, +.002 -.000	.094 - .099
Later modified to .099, +.002 -.000	

Interface: Cross Bar Pin Hole in Differential Screw with Roll Pin
Motorshaft Pin Hole with Roll Pin

ID Hole (in)	OD Roll Pin (in)
.094, +.004 -.001	.094 - .099

Figure 27
Summary of Mechanical Fits

Error Analysis

In conjunction with performing the detailed analyses for proper parts design an error analysis can highlight areas that could contribute error to system motion. The sequence of activities that the system undergoes to position the ion source in the critical region of the anode-puller interface includes a variety of tasks which are listed below with their expected error contributions. Complete calculations for the various error values may be found in Appendix A.

Software / Electrical Errors

- Software mishandles receipt of x, y, and z values from user and improperly converts data to α, β, z parameters for coordinate transformations. Mistakes in calculation of exact solutions for $z1$, $z2$, $z3$ leg values.
  Expected Error Contribution: None.

- Software inconsistently calculates number of revolutions and steps for move.
  Expected Error Contribution: None.

- Stepper motor pulsing data is lost during transmission to motor driver cards.
  Expected Error Contribution: None. Comparison of actual with theoretical pulse output would eliminate this as an error contributor.

Drivetrain Errors

- Inefficiencies in transmission of torque through the drivetrain caused by system backlash or inexact thread profiles results in linear displacement errors.
  Expected Error Contribution: Minimal. Screw threads will be preloaded by constant downward atmospheric pressure on o-ring, preload springs, and weight of source. Both screw thread profiles are standard unified thread profiles which will be created with standard taps.

- Resistance to sliding motion of brass balls on stainless steel plate due to excessive frictional forces causes increase in system torque load.

Expected Error Contribution: None. For a μ=.35 (brass on steel), the frictional force value is μN=(.35)·(57/3) = 7 lb. This amount of force will not cause any significant bending moment in the 3/8" stainless steel driveshaft at the entrance to the bushing or any extra torque load on the motors.

- O-ring loses lubricity and triples resistance to motion value.
  Expected Error Contribution: Minimal. 3X increase in o-ring resistance (5→15 lb.) offers minimal impact on system torque load. Torque load factor of safety is three.

- Excessive torque is generated because system is run without lubrication in drive system threads.
  Expected Error Contribution: None. Maximum expected torque load in a no-lubrication condition is 93 oz-in. Each stepper motor is rated at 150 oz-in (no-load) torque.

Loss of Orientation - Inexact Positioning Errors

- Stepper motors lose step periodically due to over-pulsing.
  Expected Error Contribution: None. Pulse rate limited by software.

- System is inadvertently driven into existing hardware causing motors to stall.
  Expected Error Contribution: None. Range of motion limited by software.

- System loses step when driven against the mechanical stops and all motors do not move uniformly to the start position.
  Expected Error Contribution: Minimal. The motors could be out of phase by at most 8 steps (for 1/2-step mode) which corresponds to (8/400*360) = 7.2°. The minimum increment of system motion is 21 steps. Since this error value is a one-time, non-cumulative contributor, the 8 steps are not considered to offer significant error.

Overall, this design approach seems relatively insensitive to error. The accuracy of this error analysis will be evident in Chapter Five, "System Evaluation".

Cost Summary

Table 3 shows a comparison of the current system cost and the new system actual cost for those components that would change with the new mechanism design. The difference between the new system cost and the current system cost is ~ $1500. The main reason for the difference in these two values is due to the cost differential between the presently-used alternating current motors with no driver cards and the new step motors with associated driver cards. Other significant contributors include a 2.5X increase in cost between the current pivoting upper plate and the new pivoting upper plate. The reason for this difference is the increase in functionality that the new pivoting upper plate offers by combining linear actuation and the maintaining of orthogonality between the pivoting upper plate and the driveshafts.

If the new system is chosen for use as regular production hardware, costs would be lowered in the following areas:

- PN 1403529, Flange, Top, Ion Source Tower could be integrated with PN 1403886, Flange, Stationary Plate, Ion Source Positioning with a single cost similar to that of 1403886.
- PN 3301341, Assembly - Cover - Faraday Cage could be stamped sheet metal with a lower cost instead of being machined as it was for the prototype.
- PN 3301338, Assembly - Flange, Pivot, Plate - Ion Source Positioning could potentially be made out of aluminum instead of stainless steel if it is compatible with the electrical requirements.

Total potential savings if the above three changes were implemented would amount to ~ $616. This would correspond to a total system cost of ~ $2400. This is a $1000 increase over the current system. Table 3 summarizes this information.

Table 3
Cost Comparison Between Current and New Systems

Parts Used in Current System:

PN	Description	Qty	Current Cost	Current Ext. Cost
1403530	Flange, Adjusting Plate, Ion Source	1	241.23	241.23
1403529	Flange Top Ion Source RDS 111	1	266.31	266.31
1403717	Safety Cage, Ion Source	1	333.50	333.50
1403713	Shaft Screw Adjustment	3	20.00	60.00
1403714	Coupling Screw Adjustment	3	30.00	90.00
1403566	Screw Differential Ion Source Adjust.	3	28.90	86.70
	Motor, A/C	3	37.00	111.00
2360046-03	Compression Spring	3	0.50	1.50
1403567	Pin Differential Screw (modified PUR part)	3	83.37	250.11
2330047-04	Pin Spring 1/16" diameter	6	0.25	1.50
2600476	Screw 1/4-20 x .625 L Shldr Skt Hd SST	3	0.25	0.75
2600522	Screw 1/4-28 x 1.25 L Pan Hd Slt SST	3	0.29	0.87
	TOTAL:		$1041.60	$1443

Parts Used in New System:

PN	Description	Qty	Actual Cost	Extended Cost
3301338	Assy, Flange, Pivot, Plate	1	695.00	695.00
1403529	Flange Top Ion Source RDS 111	1	266.00	266.00
1403886	Flange, Stationary, Plate	1	320.00	320.00
3301339	Driveshaft, Threaded, Ion Source RDS	3	118.00	354.00
1403717	Safety Cage, Ion Source (Mod'd Cover)	1	200.00	200.00
5650050	Motor, Step, Hybrid, Size 34	3	114.00	342.00
5650051	Driver, Step Motor, Full & Half Step	3	138.00	414.00
1403889	Coupling, Driveshaft	3	77.00	231.00
3301340	Assy, Screw, Diffrential, Ion Source Adjust.	3	86.00	258.00
	System Cables (Estimated Internal Labor)	2		120.00
2360046-03	Compression Spring	3	0.50	1.50
2330047-04	Pin Spring 1/16" diameter	6	0.25	1.50
2600476	Screw 1/4-20 x .625 L Shldr Skt Hd SST	3	0.25	0.75
NEW	Screw 8-32 x .625 L Shldr Skt Hd SST	3	1.00	3.00
NEW	Screw, 1/4-28 x .625 L Flat Hd SST	2	0.50	1.00
	TOTALS:			$3200

CHAPTER FOUR

SYSTEM CONTROL SCHEME

Overview

The system control scheme consists of an executable 'C' code, which converts user-prescribed inputs into step and direction data for motor pulsing, and five pieces of hardware which use the control logic ultimately to drive three step motors. Controlling hardware will be described first, followed by a description of the software code.

Control Scheme

Figure 28 shows a block diagram of the control scheme for this system. Note that the major components of the control scheme include a personal computer (PC) executing a 'C' control software, a parallel-port input/output (I/O) card that plugs into the backplane of the PC, an open collector driver circuit, three motor driver cards, and three stepper motors. Each component will be described in this section.

The personal computer used in this project was an 8-MHz, AT-style PC. Faster computers could be used; however, the 80286-class microprocessor in this PC was more than adequate for the mechanism actuation / control task.

A parallel-port I/O card was used as the output path for sending pulse and direction bits to the motor driver cards. The card is divided into three, 8-bit ports that may be used for either input or output tasks. For this application, all ports were set to output. For convenience in programming, Port A was chosen for step pulse output bits and Port B for direction output bits. By segregating step data from direction data, the software can independently write a byte (8-bits) to each port and simultaneously step and

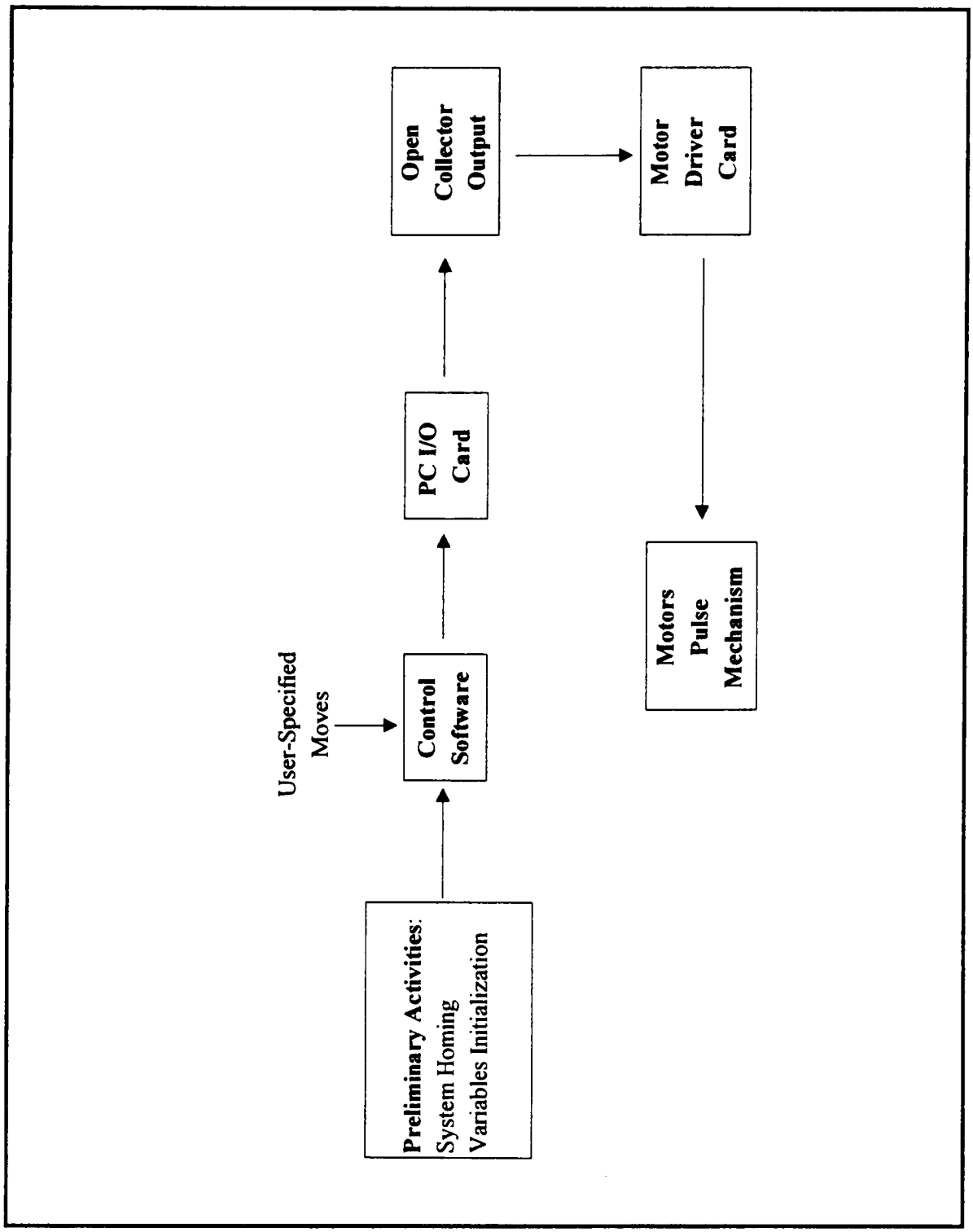


Figure 28
Control System Block Diagram

set direction on all three motors. This approach allows system moves to be made in a more expedient manner than sequential stepping of each motor would allow.

Setup activities performed by the software include defining the base address for the I/O card, specifying the address offsets for each port, and specifying either input or output data direction for each port. See Appendix C for a complete listing of the 'C' source code.

Open Collector Driver Circuit

Step and direction output from the PC I/O card next passes into an NPN open collector driver circuit. The motor driver cards require a logic source that is capable of sinking 5 mA[8]. The PC I/O card is capable of sinking only 1.7 mA[9]. To account for the mismatch, an NPN open collector driver circuit was inserted between these two cards. The external open collector circuit will sink 12.5 mA, well in excess of the required value. The design equations for the open collector driver are included in Appendix E.

Input logic to the open collector driver is inverted by the transistors in the circuit. For this reason, the 'C' control code was arranged to invert the logic for step and direction prior to sending it to the open collector output. This inversion was accomplished by taking the one's complement of the output byte prior to making the port call. In 'C', one's complement may be accomplished by

```
Outvalue1 = ~Outvalue1;
```

where Outvalue1 is the variable name for the output byte.

Motor Driver Card

The 2035 Step Motor Driver[8] uses dual, bipolar H-bridge power amplifiers that switch at 20 kHz. The input voltage may range from 12-35 VDC. The output current ranges from 0.125 - 2.0 amps/phase. The drive is capable of supplying 70 watts maximum

output power. Automatic idle current reduction reduces the current to 50% of specified current after one second of inactivity. For the ion source positioning mechanism, input voltage is 24 VDC and the output current is 1.75 amps/phase, chosen to match the motor values for maximum performance.

Stiffler[10] refers to this type of drive as a chopped-voltage drive ("chopper driver") as it applies higher-than-rated voltage to the motor to induce a rapid rise in current with torque following the current. As current exceeds rated current, a current sensing network switches off the voltage until the winding current decays below rated current at which point voltage is reapplied. The net effect is that the motor is driven by rated current (averaged) but responds more quickly than if rated voltage alone was applied. Motor inertia is sufficiently large to average the chopped torque and to provide smooth motion.

The driver card uses optical isolation circuitry to prevent the electrical noise generated by the switching amplifiers from interfering with the driving logic. Optical isolation is accomplished by powering the motor driver from a different supply than the 5 VDC (from the PC backplane) that drives the step and direction logic. Signal communication from the logic input circuitry to the driver circuitry is accomplished with infrared light that turns off and on as dictated by the logic. Phototransistors in the drive circuit generate motor pulses corresponding to step and direction logic. Complete specifications for the motors and motor driver cards are included in Appendix D.

Step Motors

The final hardware components of the control scheme are the three step motors that provide the input torque to drive the system. As calculated in the "Drive System Calculations" section, the maximum torque required by the system is on the order of 50 oz-in. Requiring a factor of safety of at least two, preferably three, the 4034-322 step motor was chosen from Applied Motion Products. This motor is built around the smallest

NEMA class 34 chassis and delivers 150 oz-in of holding torque under no-load, zero-velocity conditions. Space constraints between the top of the upper yoke and the inside face of the lead / concrete shielding of the cyclotron dictated that the motor height had to be less than 2.5 inches. The 4034-322 has a height of 2.48 inches.

The motor was modified in two ways for use in this mechanism. First, a 0.094 in (2.388 mm) hole was drilled into each motor shaft to accommodate a roll pin that is pressed also through the acetal coupling. The acetal coupling transmits the motor torque to the differential screw. Second, one corner of the motor mounting flange was milled off at a 45° angle so that all three motors could be nested into position on the mounting plate. The system layout required that only two motors be modified. However, the choice was made to alter all three motors so that CTI has to maintain only one part number and can easily interchange any of the three motors.

Six-lead motors can be connected to the driver card in either a series mode or a center tap configuration. In series mode, motors produce more torque at low speeds but cannot run as fast as the center tap configuration. Since inertial loads are low for this system, the center-tap configuration was chosen with the motors operating in half-step mode. Appendix D shows the center-tap connection diagram for these motors.

System Software

General Approach

The main computational engine for the software code is the mathematical model that was presented in Chapter Two. Figure 29 shows a block diagram of the software algorithm. The software code was written in the 'C' programming language for compatibility with other CTI / CCS programs and ease of future maintenance. The user runs an executable file when positioning the ion source with this mechanism. A complete listing of the source code for the controlling software may be found in Appendix C.

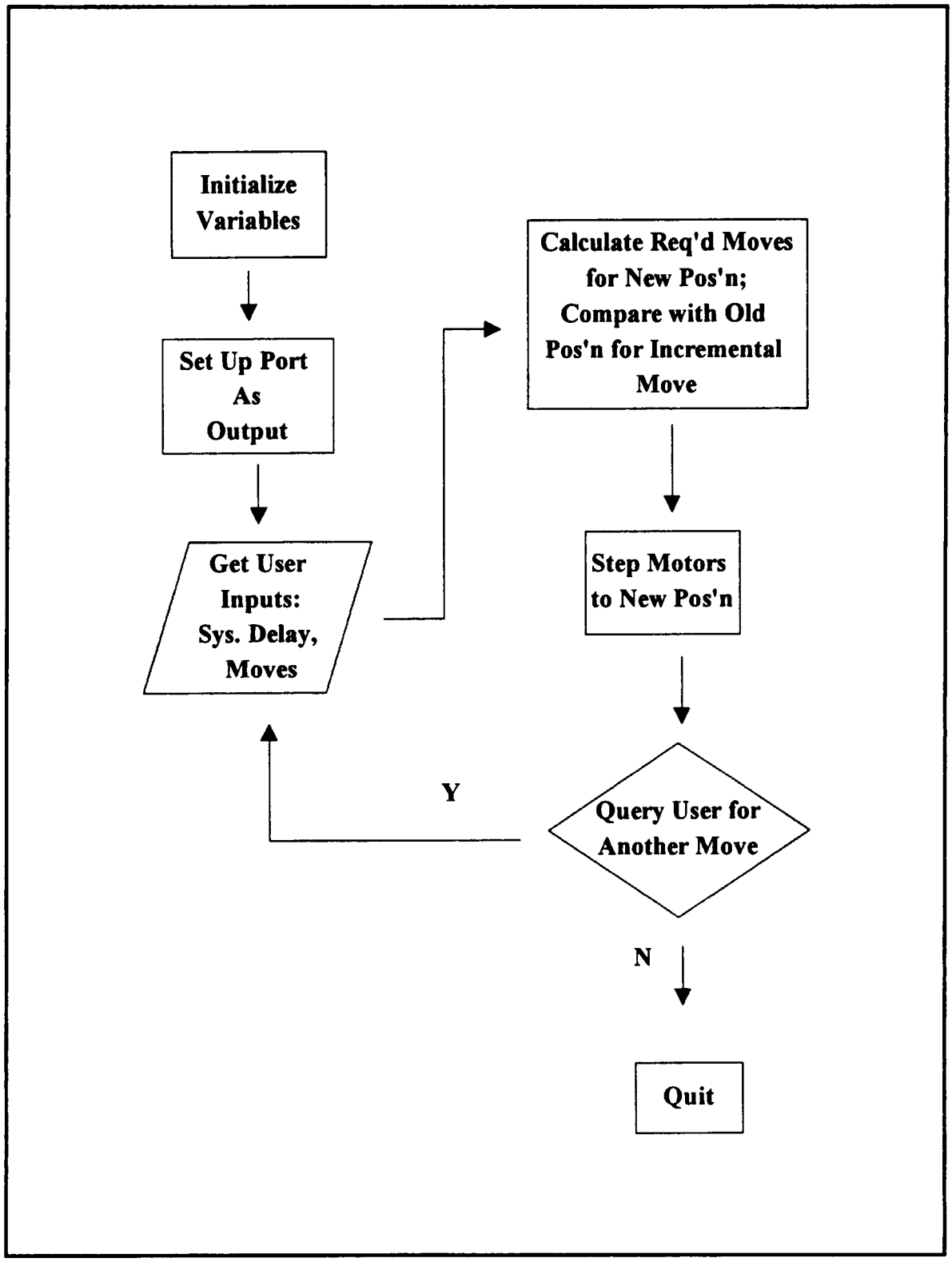


Figure 29
Block Diagram of Software Algorithm

The software performs the following computational tasks using double-precision, floating calculations:

- Queries users for desired gap, *xy*-aperture alignment, and *z*-aperture alignment values.
- Converts these values into *xyz* values for moving the anode aperture relative to the puller aperture.
- Converts *xyz* values into $\alpha\beta z$ values and calculates H_x, H_y values.
- Calculates new leg lengths based on $\alpha\beta z$ and H_x, H_y values.
- Calculates number of steps and direction data for each leg.

Having converted user-prescribed inputs into logical data for pulsing the motors, the software performs the following input/output (I/O) and motion control tasks:

- Sets the base address for the personal computer (PC) parallel-port I/O card.
- Sets the offset address for the three ports on the I/O card.
- Sets the three ports to output only.
- Writes data to the I/O card to home the motors and place the mechanism in the "start" position using a fixed routine for pulsing the motors.
- Writes the "move" data to the output port based on system calculations causing simultaneous pulsing of the three step motors.

Software Delay

In order to successfully pulse the motor, the software cycles the output step bits from a logical low to a logical high state at the desired pulsing frequency. Direction bits are set once for each move. This cycling requires the use of a software delay function that keeps the pulses in either a high or low state for an exact length of time before switching to the opposite state.

The DOS/BIOS timing function of all true PC-compatible machines has a 54 msec system clock pulse that keeps PC functions synchronous. To achieve timing functionality below this threshold, two options are available.

First, a machine-dependent loop can be established that simply makes a function call a calibrated number of times to accomplish the desired delay. This function would be called each time an output pulse was required to wait between a high and a low state. The main drawback to this solution is its lack of portability between different hardware. This method also requires calibration of the output signal with an oscilloscope.

Second, specialized software[14] may be purchased that can access the 8253 timer chip found in all compatible PC's and perform micro-second level timing. This software, and its affiliated function calls and libraries, allows complete portability between PC-compatible machines and offers ultra-precise timing capability to the programmer.

During the course of this project an older version of micro-second-delay timing software was evaluated for use in this application. Executing the demonstration delay routines in this shareware software revealed that, indeed, micro-second delays were possible. An attempt to deploy these function calls and library files directly into mechanism source code was unsuccessful. The manufacturers encoded the demonstration software with a flag that flashed a message about the company and said to "press any key to continue" each time the delay routine was called. This message, embedded in the library routine, rendered the software useless for any purposes other than demonstration. For a price, however, this software is available and could be integrated at a future time. The current revision of the mechanism control software has all of the function calls and variable declarations included as comments.

In the spirit of prototype demonstration, a machine-dependent delay was used for this application. Different multipliers corresponding to numbers of passes through the delay loop were used to calibrate the delay function. Figure 30 shows the correlation of

numbers of passes to elapsed time as measured by an analog oscilloscope on the output pin of the SSI board. A value of 20, for example in Figure 30, corresponds to a hardware delay of 1 msec.

Finally, the executable software that the operator sees was designed to be "user-friendly". Appendix C shows sample output from the three screens that serve as the interface for this control software.

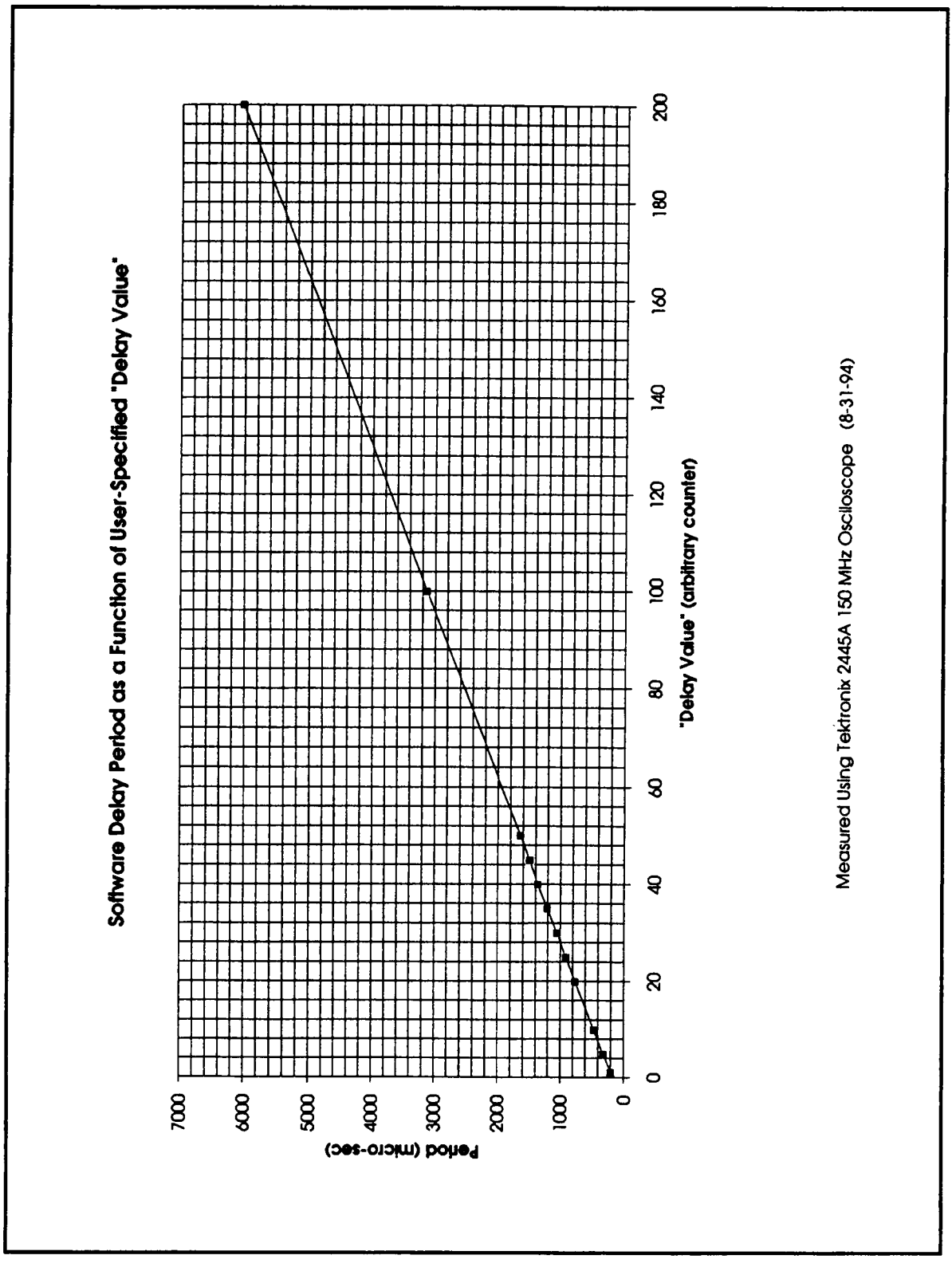


Figure 30
Software Delay Calibration Curve for Dedicated 8-MHz, AT PC

CHAPTER FIVE

SYSTEM EVALUATION

Overview

After assembling the mechanical hardware and completing the controlling software code development, a range of evaluation tests were conducted to assess system performance. These tests included measurement of individual piece parts for comparison of post-machining dimensions with that of design specifications as well as system testing for minimum increment of motion and complete range of motion. Evaluation of the control system included verification that an accurate pulse count was being sent to the stepper motors and that accurate numbers of revolutions of the motors were achieved for a calibrated input.

Original plans included the use of a coordinate measuring machine (CMM) for minimum increment of motion and range of motion testing. Several attempts were made to use an LK-850C computer-controlled CMM in use at Martin Marietta Energy System's Y-12 Operations in Oak Ridge, Tennessee. However, due to administrative problems at the Y-12 Plant, the CMM was not available for system test. Instead, hand tools were used to verify system performance along single axes of motion. A detailed description of the tools and methods used for evaluation follow in this chapter.

Test Configuration

The basic test configuration for measuring the minimum increment of motion, the system range of motion, and system repeatability is shown in Figure 31. The test stand is constructed of extruded aluminum members with machined ends which allowed for construction of a rigid platform for mounting the positioning mechanism and ion source

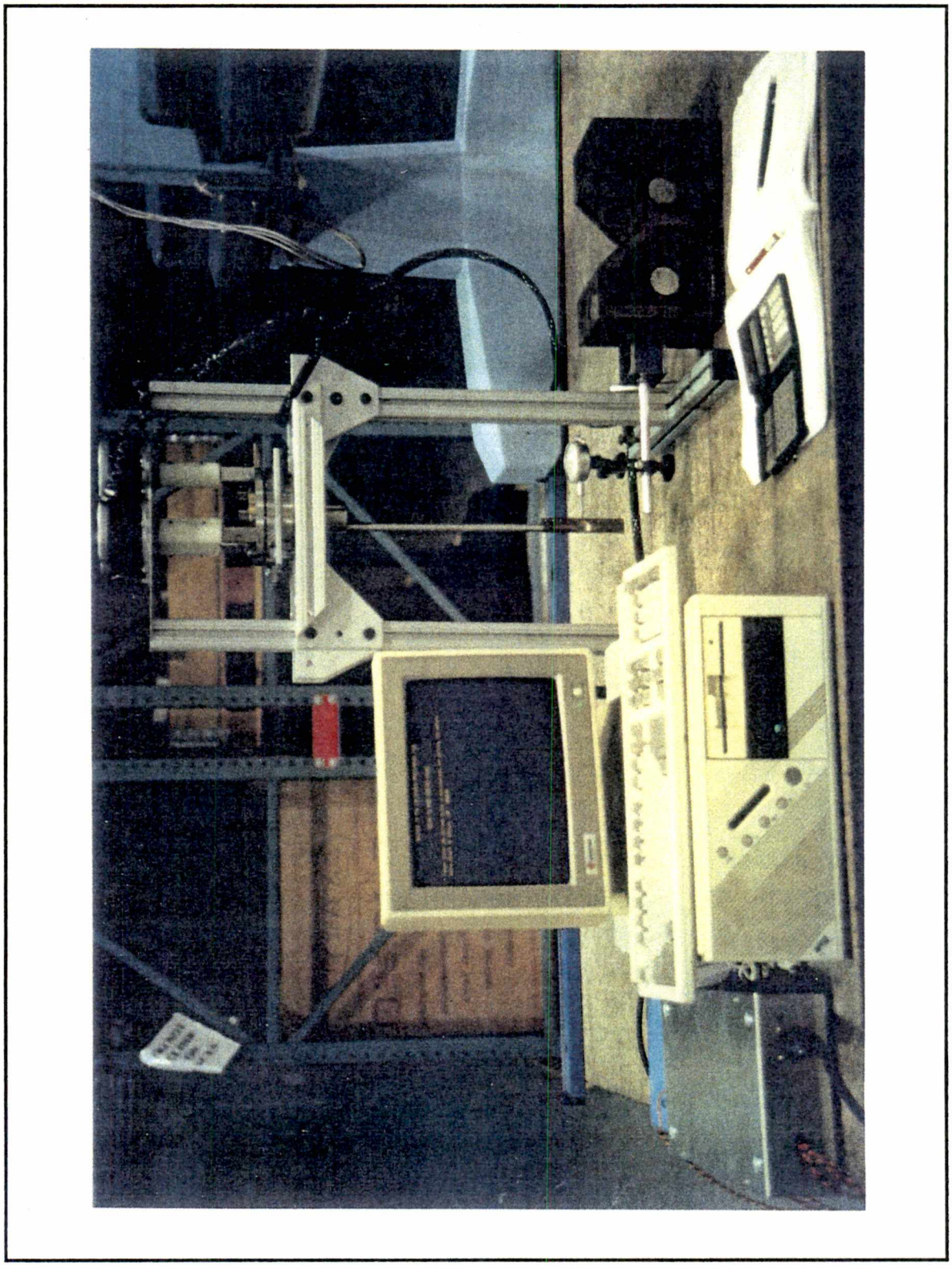


Figure 31
Photograph of Test Configuration

assembly. Prior to each test run the stand was adjusted to an approximately level condition and securely clamped to a granite surface plate. The stand did not have to be absolutely level or perfectly square since all measurements were recording relative displacements from an arbitrary starting point.

Measurement Tools

As shown in Figure 32, a height gage was used to measure the vertical displacements of the positioning mechanism. Consistent probing forces were maintained by incorporating a probing dial indicator with the standard height gage to monitor probing deflections with each measurement. Figure 33 shows the test configuration for measuring system y-axis motion. As shown, a standard two-inch-travel dial indicator was suspended by a magnetic stand attached to a large steel section such that the probing tip of the indicator sat squarely against the anode adjacent to the aperture. A known initial deflection was applied to the dial indicator so that full range of system motion could be measured without repositioning the dial indicator. Figure 34 shows the test configuration for measuring x-axis system motion. Similar to the y-axis setup, the probing tip of the dial indicator was placed on the curved edge of the copper housing surrounding the anode at a height corresponding to a plane cutting through the center of the anode aperture. The dial indicator was preloaded by a known amount for full range of x-axis motion to be tested without repositioning the dial indicator.

Measurements of most piece-parts were made with standard hand tools such as calipers and micrometers. However, CTI's manual CMM was used to measure the diameter bolt circle of the rotating upper plate since this measurement could not have been accomplished easily with hand tools.

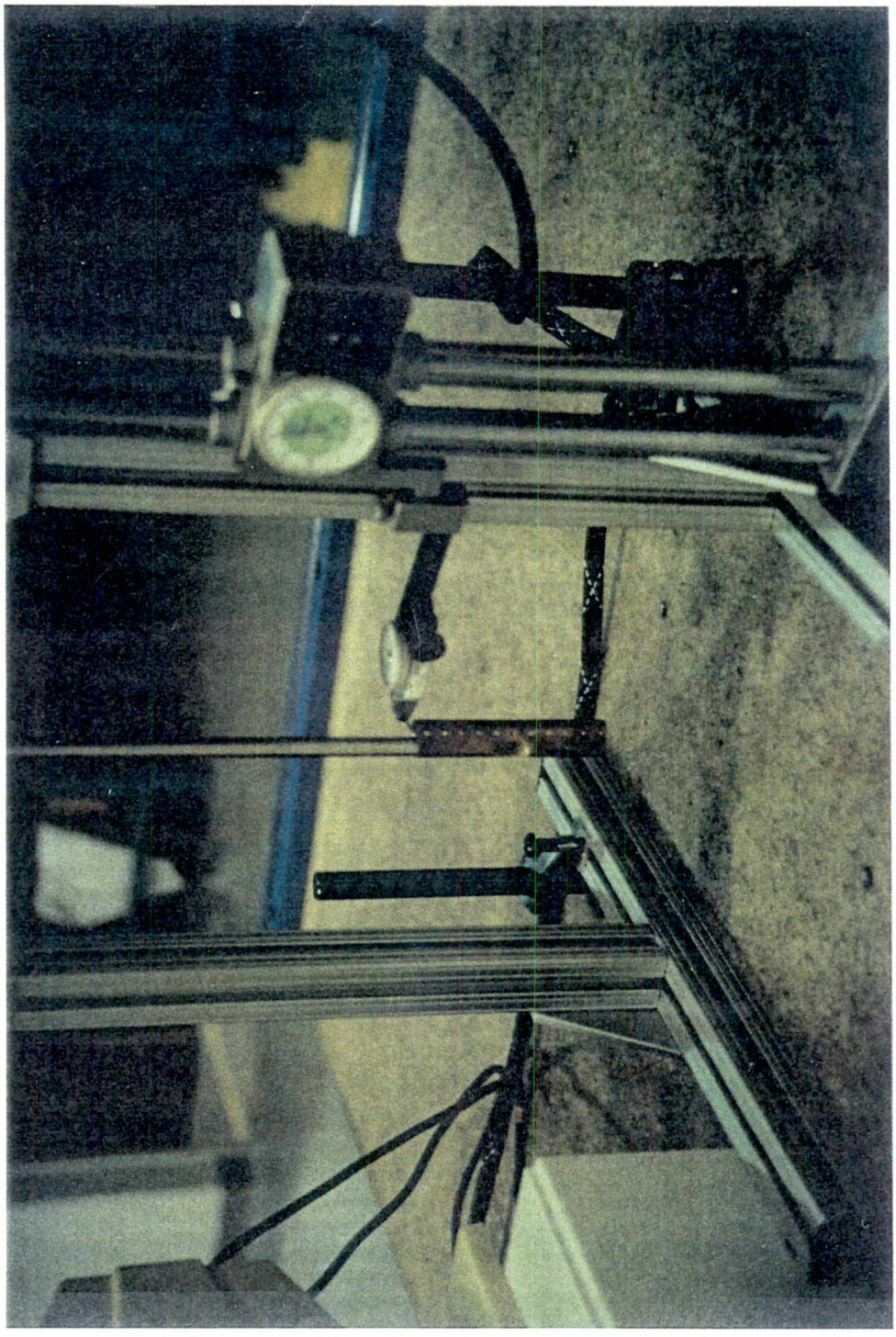


Figure 32
Photograph of Z-Axis Measurement Scheme

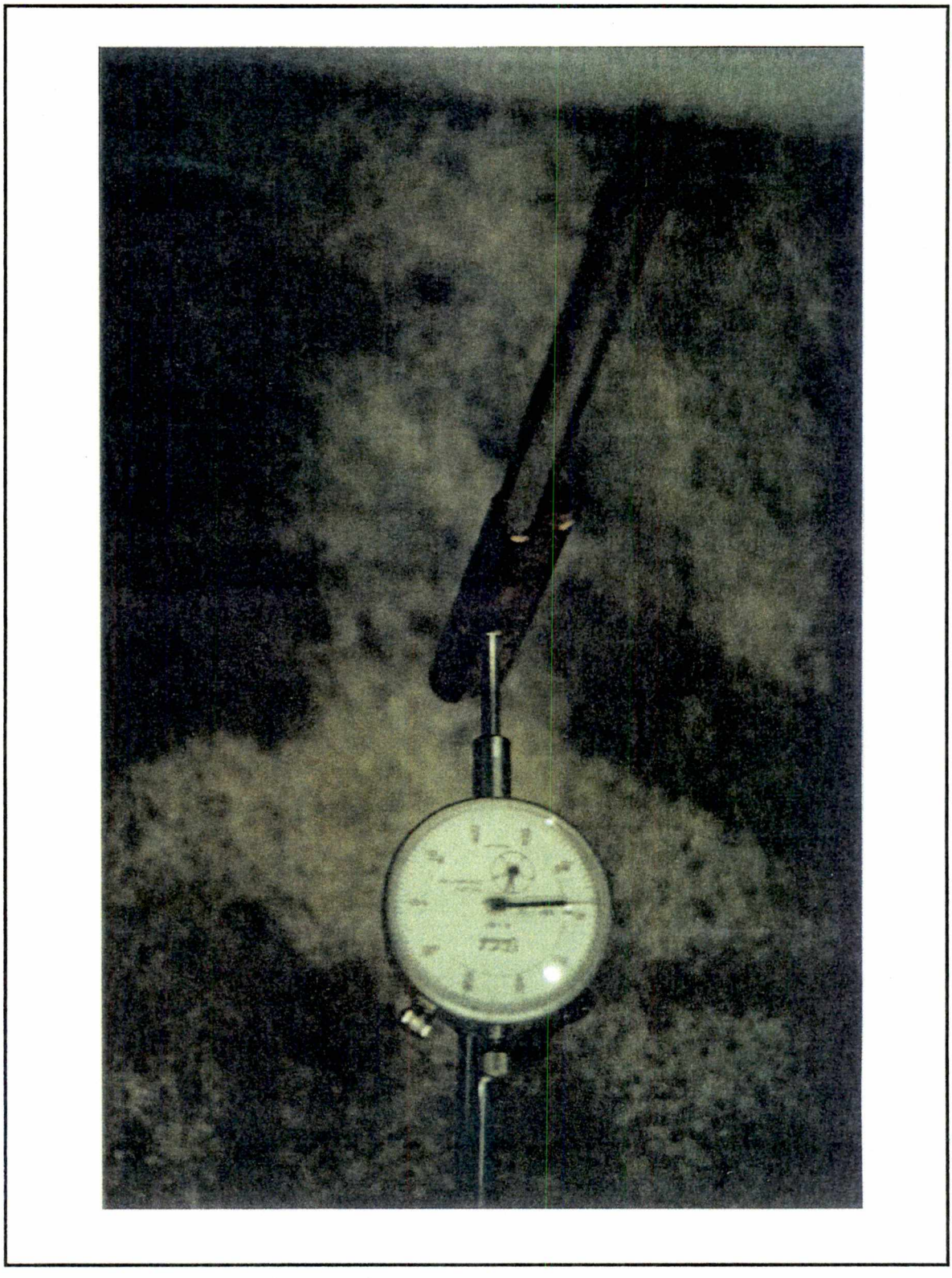


Figure 33
Photograph of Y-Axis Measurement Scheme

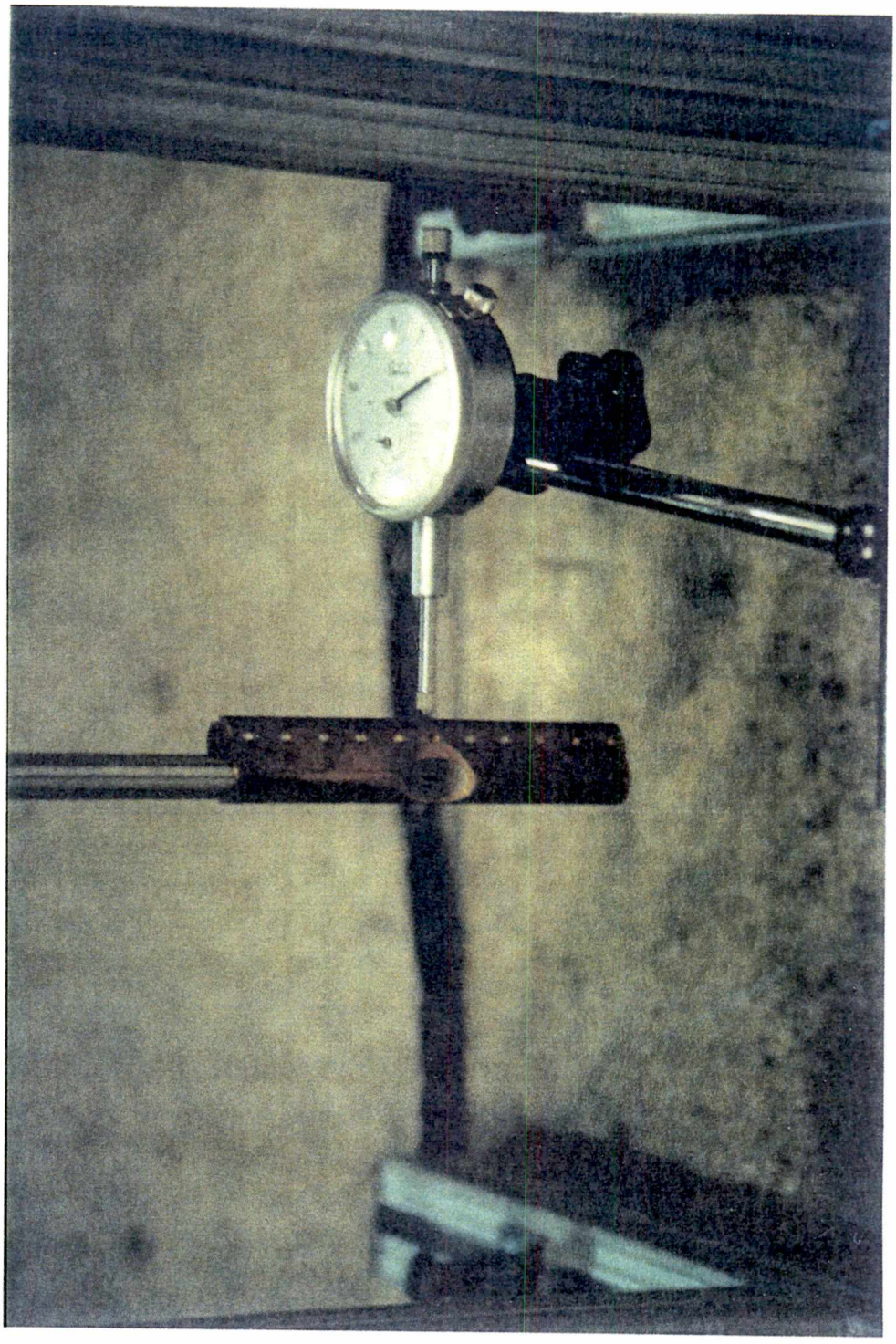


Figure 34
Photograph of X-Axis Measurement Scheme

Test Conditions Relative to In-Use Conditions

Since existing hardware in the cyclotron hinders direct measurement of the mechanism motion at the anode-puller interface, minor changes were introduced to the system configuration to best model end-use conditions during testing. These modifications include:

- Use of a functionally equivalent ion source assembly with only those components required to establish the critical distance from the positioning mechanism to the anode-puller interface. The assembly also maintained the rotational orientation of the ion source mounting holes relative to the anode aperture.

- Placement of a higher-than-specified preload on the rotating upper plate to simulate the constant downward load of atmospheric pressure on the o-ring. This additional load is critical to generate the correct torque conditions on the stepper motors.

- Manually aligning of the motor driveshafts with the differential screw centerlines in the test stand using visual comparison only. This alignment method is used since no special locating features are present in the test stand. This alignment, though inexact, induces minimal error into system motion since the drive coupling acts as a universal joint.

Evaluation Criteria

Engineering drawings of system components are the primary evaluation criteria for measuring the performance of the fabrication processes that make the parts. Project performance goals, stated previously in the "Statement of Problem" section, are the primary metrics for evaluating system performance. Design goals are restated below:

- Minimum Increment of Motion: 0.1 mm.

- Range of Motion for *x*-axis: ± 3.0 mm (± 0.118 in).
  y-axis: + 2.2 mm (+ 0.088 in), - 3 mm (- 0.118 in).
  z-axis: ± 2.0 mm (± 0.079 in).

- Accuracy of Motion for *x*, *y*, and *z* axes: ± .2 mm (± .008 in).

Original range-of-motion design goals specified ± 3 mm along the *y*-axis. Later investigation revealed that existing hardware prevents travel of a full + 3 mm of motion in the *y*-direction. All other displacement target values were achievable within the constraints of existing hardware.

Evaluation of the ability of the positioning system to home itself by driving against mechanical stops and move to a desired start position was also assessed. Although end-users stated that manual homing of the mechanism was an acceptable start-up method, the final mechanism was designed to automatically home the system and set a starting gap of 3.81 mm (0.150 in) from an arbitrarily displaced position. Performance relative to these criteria is discussed below.

Results

Software Pulse Count and Motor Travel Verification

The strategy for assessing the functionality of the control system was to capture as many upstream operational steps as possible by a single downstream measurement. By counting the number of pulses being sent by the control code to the motor driver cards, the software code is treated as a black box with user-specified *x*, *y*, *z* inputs and pulse counts as output. An ORTEC Model 871 Timer and Counter, plugged into a standard nuclear instrumentation module (NIM) chassis, was used to count the number of pulses sent to the motor driver cards for a given calculation. Comparison of theoretical to actual pulses was straightforward since the software presents a data screen which details the key parameters for future moves by the system. Testing showed exact agreement between the software-generated pulse counts for a given move and that which was seen at the input to the motor driver cards.

Motor travel verification was conducted by first calculating a set of system moves in the *z*-direction that would generate a progressive series of integer output revolutions.

The *z*-direction was chosen so that all the motors would move an equal amount enabling review of any discrepancies that might exist between the three driveshafts for a uniform move. The set of input test parameters covered the range of output motor driveshaft revolutions from one to fifteen revolutions in one revolution increments. The performance of the three output driveshafts was measured by using the lower slot in the acetal coupling as a timing mark which was compared to a reference mark placed on each of the three fixed towers on the rotating upper plate. For this test, all upstream calculations were treated as a black box with user-specified *x*, *y*, *z* inputs and motor driveshaft revolutions as output. Evaluation revealed an exact relationship between the calculated number of revolutions required to complete a system move and that which each of the three motor driveshafts executed for the input move.

By experimentally validating the analytical algorithm embedded in the software code, the number of pulses sent to the motor driver cards, and the number of revolutions of the motor driveshafts, performance of the mechanism was shown to be a function of the mechanical hardware downstream from the motor driveshafts. Early establishment of the control system functionality allowed primary focus to be placed on the hardware during system troubleshooting.

Pivoting Upper Plate Evaluation

The rotating upper plate serves as an integrator for many different aspects of the mechanism function. It supports and retains the plasma ion source, facilitates linear displacement through the differential screw and driveshaft, and maintains "leg" orthogonality via the three bushings. The most critical function that this part performs is the transmission of pivoting motion corresponding to the desired anode moves from linear displacement of the drive screws. Inspection of this part revealed that the 3/8-32 UNEF threads were properly tapped into the three stainless steel towers. Inspection also showed

that the three 0.094 in pin holes were properly drilled through the tower wall and brass bushing. The diameter bolt circle (DBC) for the 3/8-32 UNEF holes was also fabricated properly.

Wobble Error Induced by Test Configuration

The test stand shown in Figure 31, page 71, allows some variability in the initial alignment between the fixed location of the motor driveshafts and the variable location of the mechanism. Since the fixture has no special locating features, test alignment is set by visual inspection only. An induced error, though small, was revealed as a periodic wobbling motion evident during system testing. This wobble was propagated by a misalignment between the centerline of each of the three driveshafts with each of the three acetal couplings. To minimize the wobble error, the coupling was modified by adding ID clearance so that the coupling could act as a universal joint between the motors and the differential screws. The modified coupling is an enhancement to overall system flexibility since the end-use mounting scheme for the drive motors is based on a low-tolerance drawn sheet metal box mounted on the cyclotron. Built-in compensation for driveline misalignment makes the overall mechanism a more robust design.

Driveline Errors

Overall system range of motion was reduced due to a design error in the driveline calculations. Determination of the optimal engagement thread length for the differential screw and driveshaft should be based on the thread pattern with the larger lead value. Since the 1/4-28 thread profile generates a longer linear displacement than the 3/8-32 thread profile for a given input revolution (1/28 > 1/32), sizing calculations should have used the 1/4-28 lead value of .0357 in (.91 mm) instead of the 3/8-32 lead value of .0313 (.79 mm). To avoid catastrophic driveline failure during operation, the mechanism is

assembled so that mechanical homing is possible without disengagement between either the differential screw and the driveshaft or the differential screw with the mating tower threads. Biasing the driveline displacements to envelope the 0 → -2 mm portion of the z-axis motion allows positive z-axis motion of only 0 → 1.4 mm. Correction to this problem requires increasing the thread engagement length for both the differential screw and the driveshaft based on travel calculations using the 1/4-28 thread profile and remaking these parts.

Finally, the last piece-part error that was found during system evaluation was a mismatch between the driveshaft guide pin holes in the towers of the upper plate and the width of the guide slots in the driveshafts. This error became apparent only when system actuation along an axis changed direction. During the transition stage between loading from one direction to the other, the differential driveline delivered a lead of 1/32 in instead of the differential lead discussed in Chapter 3. This is because the preloaded driveshaft tends to rotate with the differential screw unless it is constrained to act as a threaded stud. When acting as a threaded stud, the driveshaft's 1/4-28 threads oppose the motion of the 3/8-32 threads and the differential screw delivers the net differential motion. An error of ~ .001 in at the driveline location in the mechanism gets magnified by a (17.671 / 1.954) or 8.6 factor so that a .009 in error is seen at the anode-puller location. The solution to this error is to increase the hole size from .094 in to .099 in and accept the minimal error of .002 in at the anode to leave assembly margin to account for inexact drilling of the guide pin hole.

On-Axis Displacement Data

Figure 35 shows the results of system testing along the *y*-axis using a dial indicator to record displacements. The abscissa units reflect the operator inputs in millimeters while

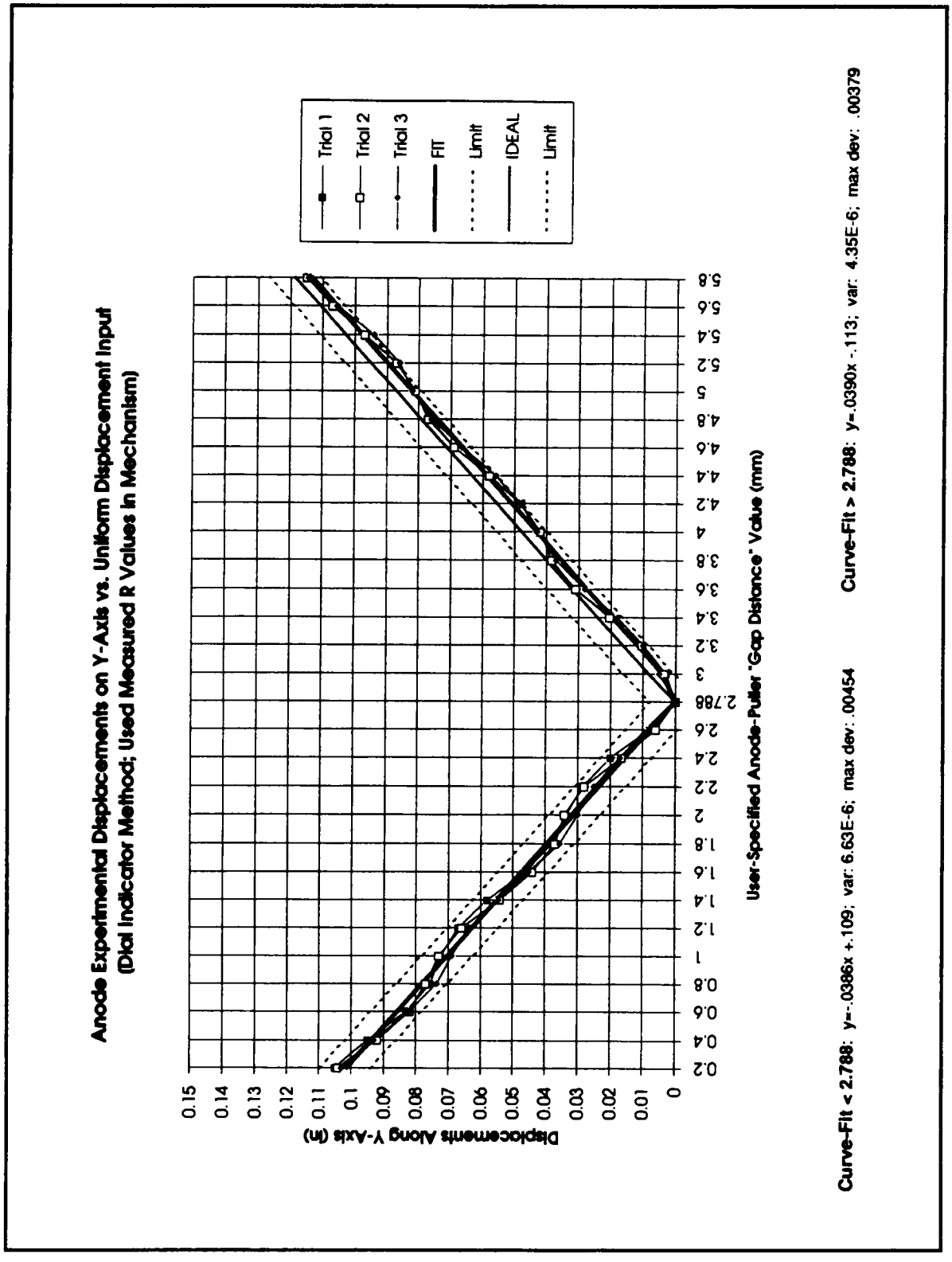


Figure 35
Y-Axis Displacement Results

the ordinate axis shows displacement values in the units of the test equipment - inches. The darkest linear lines in Figure 35 represent the best-fit line using a least-squares regression analysis from the experimental data collected for three trials. The three lighter lines adjacent to the best-fit line are the experimental data taken in successive trials which depict total system displacement (in) as a function of 0.2 mm progressive moves along the y-axis. The labeled "IDEAL" line and the two companion dashed lines represent the target value and tolerance band for the displacements. The V-shape of the displacement curve reflects the positive values that were recorded for y-axis displacements in both directions from the starting anode-puller "gap" value of 2.788 mm.

The curve-fit equations, variances, and maximum deviations are shown at the bottom of each figure. The variance, labeled as "var" on all data figures, is defined[12] as follows:

$$\text{var} = \sigma^2 = \frac{\sum_{i=1}^{n} \left(y_{act_i} - y_{bestfit_i} \right)^2}{n-1} \qquad (5\text{-}1)$$

Note the periodic nature of the experimental data. This behavior is a reflection of the wobble effect manifested as an elliptical path of the coupling / differential screw interface. Since the axial alignment between paired motor shafts and differential screws is based on visual observation during test setup, this wobble effect may be more pronounced along a single axis or spread more evenly across both the x and y axes. Exact positioning of the origin of the mechanism directly below the center of the DBC of the motors would minimize this effect.

Figure 36 presents the experimental data for motion along the x-axis using the same dial indicator method as discussed for Figure 35. Notice that data for Figure 36 does not exhibit as pronounced of a periodic motion as the y-axis data in Figure 35. The data, though very repeatable, still display a deviation from the ideal slope value of .0394. The worst-case slope error is given by the following equation:

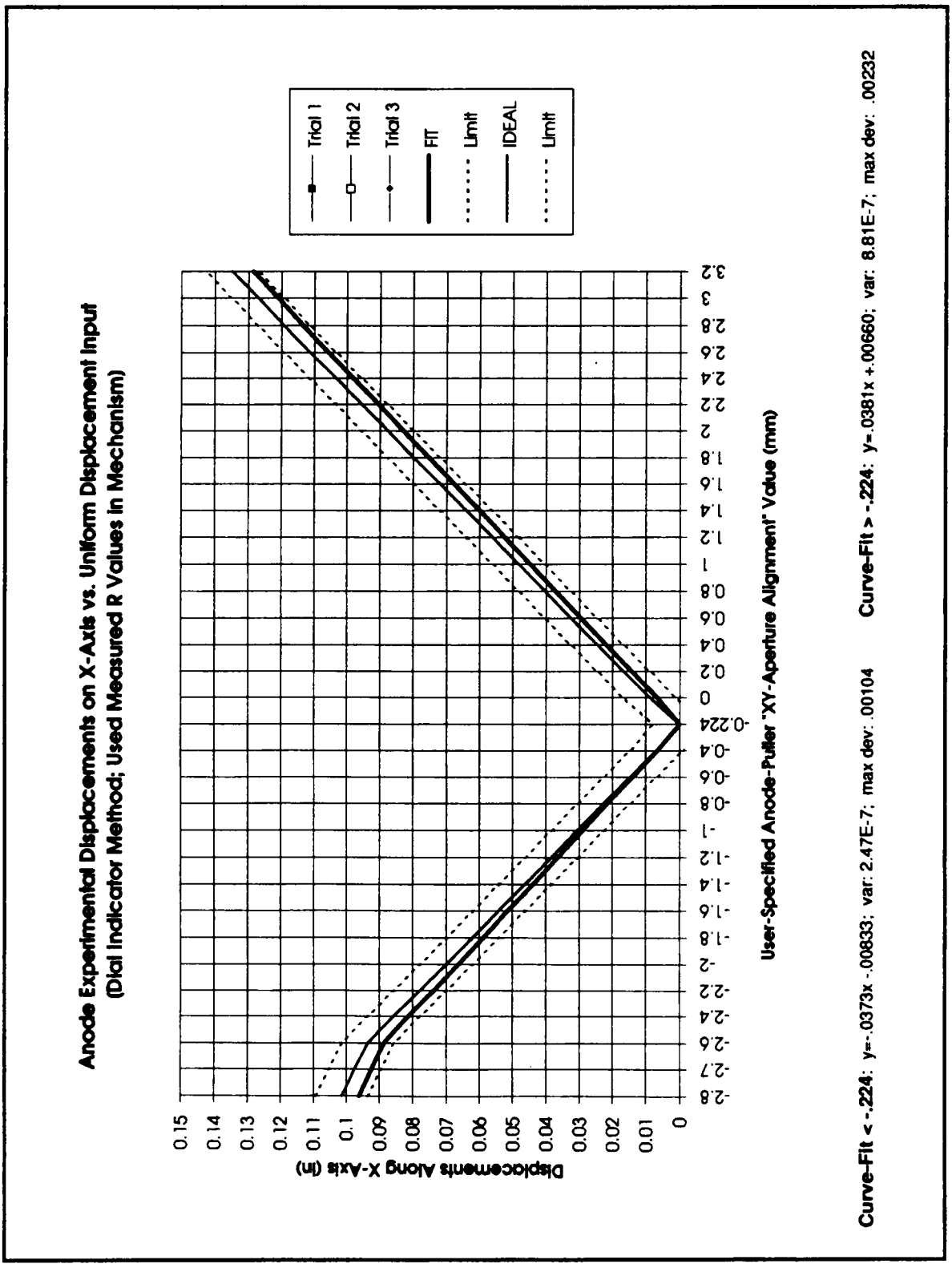


Figure 36
X-Axis Displacement Results

$$\%Error = \frac{(.0394 - .0373)}{.0394} \cdot 100 = 5.26\% \qquad (5\text{-}2)$$

Although the *y*-axis data seen in Figure 35 behave in a periodic manner which was attributed to set-up misalignment, data in Figure 36 point to a systemic error independent of test setup induced errors. Even with this 5.3% error, however, the *x*-axis motion is still within the target band.

Figure 37 shows the data for z-axis motion. As with the *x* and *y* axis motion, the dark line through the data represents the calculated, best-fit line for that data. Statistical data are given at the bottom of Figure 37. *Z*-Axis motion is accurate and highly repeatable as given by the low variance and max deviation numbers. Figure 38 summarizes the curve-fit equations, variances, maximum deviations, and the final error offset values in the extreme ends of each axis for the associated experimental data.

Contact Stress Analysis

The loading condition along the drive system from the preload, vacuum load, weight of source, and resistance of o-ring is concentrated on the tip of each of the three brass balls that mate with the stainless steel lower plate. Although the theoretical contact area for a curved surface such as a sphere is a point, the elastic nature of the materials produces a small contact area because of deflections. Equations[7] for the pressure and contact area quantify these effects. First, consider a quantity Δ which is a function of Young's Modulus (E) and Poisson's ratio (ν) for the two mating bodies:

$$\Delta = \frac{1 - \nu_1^2}{E_1} + \frac{1 - \nu_2^2}{E_2} \qquad (5\text{-}3)$$

Using this expression for Δ, the expressions for contact pressure (p_0) and the radius of the area of contact (a_0) may be stated as follows:

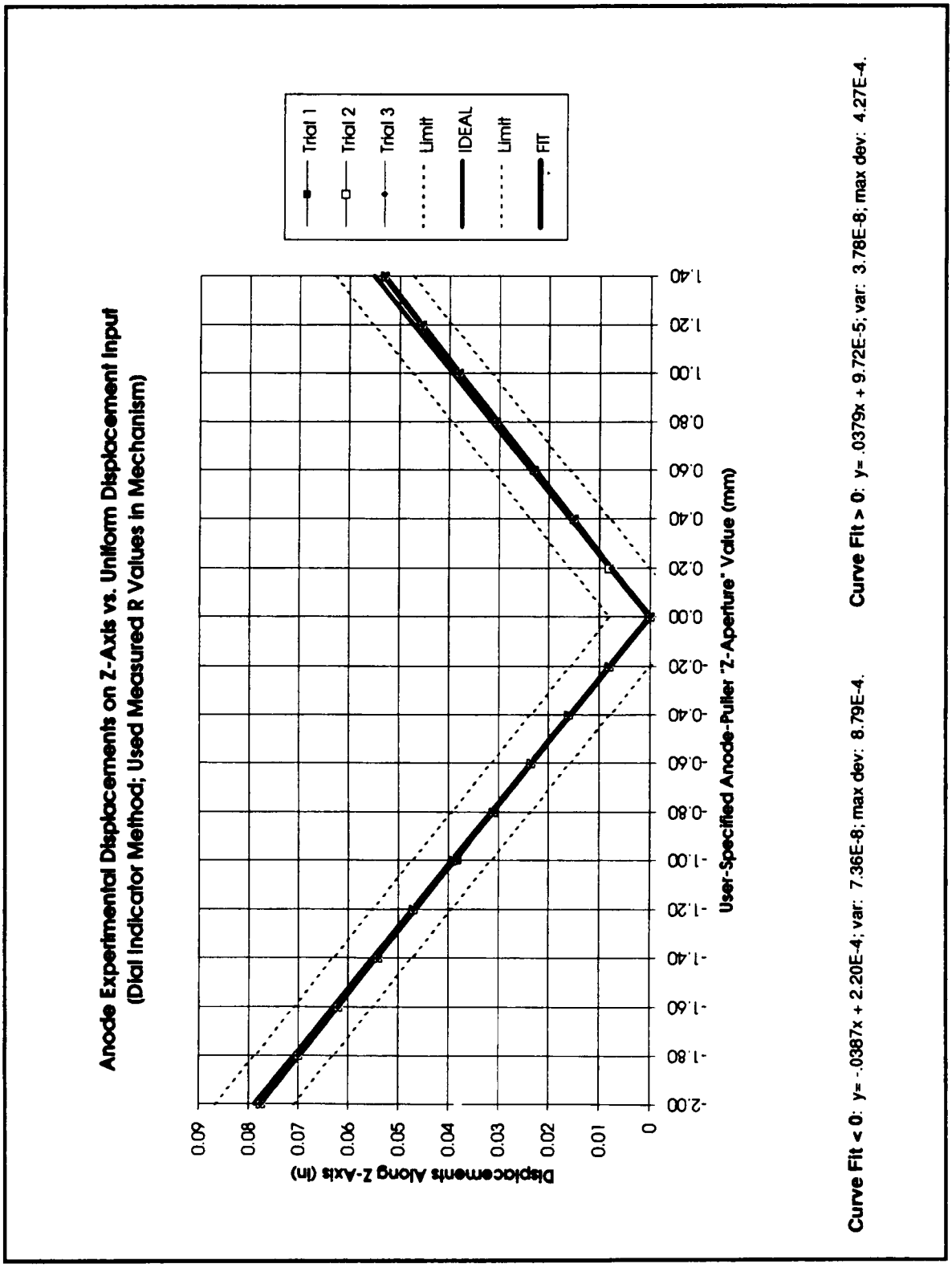


Figure 37
Z-Axis Displacement Results

Figure 38
Summary of Curve-Fit Data for X, Y, Z Axes Actuation

Summary of Curve-Fit Data for X, Y, & Z Axes Actuation

	Curve-Fit Equations	Variances	Max Deviations
Y Axis: Dial Indicator Method, Assumed Uniform, Ideal R Values			
Negative Sloped Line	y = -.0344x + .0955	1.12E-06	0.00237
Positive-Sloped Line	y= .0340x -.092	9.41E-07	0.00215
Y Axis: Calipers Method, Assumed Uniform, Ideal R Values			
Negative Sloped Line	y = -.0344x + .0961	7.61E-07	0.00166
Positive-Sloped Line	y = .0345x - .0953	2.05E-06	0.00317
Y Axis: Dial Indicator Method, Used Measured R Values			
Negative Sloped Line	y = .0386x + .109	6.63E-06	0.00454
Positive-Sloped Line	y = .0390x - .113	4.35E-06	0.00379
X Axis: Dial Indicator Method, Assumed Uniform, Ideal R Values			
Negative Sloped Line	y = -.0342x - .00546	2.68E-06	0.0031
Positive-Sloped Line	y = .0344x + .00654	2.87E-06	0.00391
X Axis: Dial Indicator Method, Used Measured R Values			
Negative Sloped Line	y = -.0373x - .00833	2.47E-07	0.00104
Positive-Sloped Line	y = .0381x + .00660	8.81E-07	0.00391
Z Axis: Height Gage Method, Used Measured R Values			
Negative Sloped Line	y = -.0387x + .00022	7.36E-08	0.00088
Positive-Sloped Line	y = .0379x + .000097	3.78E-08	0.00043

$$p_o = 0.578 \cdot \sqrt[3]{\frac{F(1/R_1 + 1/R_2)}{\Delta^2}} \qquad (5\text{-}4)$$

and

$$a_o = 0.908 \cdot \sqrt[3]{\frac{F\Delta}{1/R_1 + 1/R_2}} \qquad (5\text{-}5)$$

where:
F is the loading force;
R_1 is the radius of the sphere;
R_2 is the radius of the mating part (flat plate in this case so $R_2 = \infty$ in this case);

For the mechanism under investigation, the following values[5] were used to calculate Δ, p_0 and a_0:

$F = 57/3 = 19$ lb	$E_1 = 15 \cdot 10^6$ psi	$\nu_1 = 0.34$
$R_1 = .125$ in	$E_2 = 28 \cdot 10^6$ psi	$\nu_2 = 0.30$

The resulting values for p_0 and $a_0$were 304,000 psi and .0055 in. This value for p_0 is well beyond the elastic limit for brass. Thus, the brass ball will yield and develop a flat spot where the ball contacts the stainless steel plate. The two non-groove brass balls were measured and the diameter of the flat spot was found to be ~ .010 in. The value for p_0 is also beyond the elastic limit for stainless steel so that the stainless steel plate will experience a dimpling effect that may be seen on the surface of the plate over time. The combined effect of a recessed lower plate and a flattened brass ball on the driveshaft could result in increased resistance to motion of the bottom end of the driveshaft which could be reflected back to the motors as an additional torque requirement. A spreadsheet showing the effects of different R values with different load values is shown in Appendix F. This matter should be investigated more deeply to safeguard the long-term reliability of this mechanism. Several minor hardware changes could be considered to eliminate this effect.

CHAPTER SIX

CONCLUSIONS & RECOMMENDATIONS

Conclusions

The following conclusions may be made regarding the outcome of this project:

- Efforts have resulted in a working mechanism that demonstrates repeatable actuation of a plasma ion source to an accuracy of ± 0.2 mm from a remote actuation point located 425 mm away.

- The mechanism meets the design goal for minimum increment of motion by effectively moving the anode a minimum of 0.1 mm. The mechanism also produced positioning motion that fell within the range of acceptable values dictated by design goals.

- The closed-form solution to the mathematical model generated for this mechanism gives CTI new insight into the governing motion of the current and new systems. This knowledge can be helpful to CTI for overall optimization of cyclotron performance. This algorithm may have wider applicability to solving general three degree-of-freedom positioning problems.

- The main drivers of the 2.5X increase in system cost over the current system are the stepper motors and driver cards and the higher cost for the rotating upper plate which integrates several functions into one part.

- The mathematical algorithm that supports the software code is a robust element in the overall system design in that it may be calibrated to account for deviations in part fabrication methods.

Recommendations

Several areas for improvement were encountered during the course of this project. These elements include the following items:

- Improve the ease of retrofit of the mechanism onto existing hardware by adding clearance holes for the 1/4-28 studs that are integral to the mating hardware.

- Correct the design error in the differential screw and driveshaft to allow the system to move over the entire range of motion. Verification of this requirement with the end-user might reveal that original design goals were conservative.

- Add a machine-independent delay routine to the software code to enhance the portability of the code.

- Further investigate the effects of experimental setup variation on the induced wobble effect evident particularly in the y-axis data.

- Investigate further the high contact stress phenomenon. Pursue additional experimentation and analysis to fully assess the impact of this systemic error. Consider small interface element to increase the surface area of the mating parts while still allowing for flexibility in system motion.

This project has been a challenging analytical and design effort. The original scope was to demonstrate the ability to mathematically predict and execute the user-specified orientation of a cyclotron ion source using a positioning mechanism to achieve the desired motion in x, y, and z axes at the critical anode-puller interface. The data presented herein demonstrate that this goal was achieved. Other interesting phenomenon were propagated during the course of this development which are worthy of additional consideration.

LIST OF REFERENCES

LIST OF REFERENCES

1. Hubner, Karl F.; Collmann, Jeff; Buonocore, Edward; and Kabalka, George; Clinical Positron Emission Tomography, pp. 17 - 27. Mosby - Year Book Inc., 1992.

2. "The CTI Story", CTI-Internal Document, 1992.

3. Wolf, A.P., and W. B. Jones, W.B., "Cyclotrons for Biomedical Radioisotope Production". *Radiochimica Acta*, Volume 34, 1-7, 1983.

4. Livingood, John, Cyclic Particle Accelerators, pp. 7 - 9, 118 - 123. D. Van Nostrand Company, Inc., 1961.

5. Oberg, Erik, et. al., Machinery's Handbook, 24th Edition. Industrial Press, Inc. 1992.

6. Spotts, M. F., Design of Machine Elements, 6th Edition. Prentice-Hall, Inc., 1985.

7. Juvinall, Robert C., Fundamentals of Machine Component Design. John Wiley & Sons, 1983.

8. Applied Motion Products Catalog, *Motor Driver Card Model 2035*.

9. Keithley-Metrabyte *Model PIO-12 User's Guide*.

10. Stiffler, A. Kent, Design with Microprocessors for Mechanical Engineers. McGraw-Hill, Inc., 1992.

11. Ryle Design, *Microsoft C High Resolution Timer Toolbox*.

12. DataMyte Corporation, *DataMyte Handbook*, 4th Edition. DataMyte Corporation, 1989.

APPENDICES

A. Detailed Mathematical Computations

B. Engineering Drawings

C. Software Source Code

D. Motor and Driver Card Specifications

E. Open Collector Driver Calculations

F. Contact Stress Analysis Spreadsheets

APPENDIX A

DETAILED MATHEMATICAL COMPUTATIONS

SUMMARY OF LOADS APPLIED TO SYSTEM

The following is a summary of the manner in which the system loading is applied for the two cases of lowering and raising the upper plate of the mechanism.

As shown in Figure 23, page 44, for a given input clockwise revolution the 1/4-28 threads of the differential screw move downward ↓ at a lead of 1/28 = .0357 in/rev. At the same time the 3/8-32 threads of the differential screw displace the mating threads of the upper plate upward ↑ at a lead of 1/32 = .0313 in/rev. Thus, the differential screw drive produces a net motion of (.0357 ↓) + (.0313 ↑) = .0045 in ↓ (downward) for a given input clockwise revolution. For this configuration, the loading is as follows:

↓	Preload (against motion)	10 lb
↓	Weight of Source (against motion)	4 lb
↓	Friction of o-ring (against motion)	5 lb
↓	Vacuum load	38 lb
	Total ↓	57 lb

For a given input counterclockwise revolution, the differential screw drive produces a net motion of (.0357 ↑) + (.0313 ↓) = .0045 in ↑ (upward) for a given input counterclockwise revolution. For this configuration, the loading is as follows:

↓	Preload (with motion)	10 lb
↓	Weight of Source (with motion)	4 lb
↑	Friction of o-ring (against motion)	5 lb
↓	Vacuum load	38 lb
	Total ↓	52 lb

Thus, the maximum loading is for lowering the load. This value is 57 lb.

CALCULATIONS FOR SYSTEM TORQUE

The following two equations, one to raise a load and one to lower a load, are taken from Juvinall[7], pages 281-282. Note that these equations are for any type of thread profile and could be simplified down to the case of square threads by setting $\alpha_n = 0$.

For Raising a Load and For Lowering a Load

$$T = \frac{W \cdot d_m}{2}\left(\frac{f \cdot \pi \cdot d_m + L \cdot \cos(\alpha_n)}{\pi \cdot d_m \cdot \cos(\alpha_n) - f \cdot L}\right) \qquad T = \frac{W \cdot d_m}{2}\left(\frac{f \cdot \pi \cdot d_m - L \cdot \cos(\alpha_n)}{\pi \cdot d_m \cdot \cos(\alpha_n) + f \cdot L}\right)$$

where the following definitions apply:

W = Axial load in lbs
d_m = mean diameter of threads
f = coefficient of friction
L = Lead of Screw
α_n = thread angle in normal plane

For the 1/4-28 Threads:

$d_m = .223"$

$L = \frac{1}{28} = 0.0357"$

$\alpha_n = 29.934°$

For the 3/8-32 Threads:

$d_m = .350"$

$L = \frac{1}{32} = 0.0313"$

$\alpha_n = 29.982°$

In both cases the value for the load, W, is calculated from the following

$$W = \frac{(9.72) + (4) + (5) + (38.32)}{3} = \frac{57.04}{3} = 19.01 \text{ lb}$$

Calculation of Torque Values for Various Loading Combinations

Case 1: 1/4-28 Threads Raising a Load

$$T = \frac{(19) \cdot (0.223)}{2}\left[\frac{(.35) \cdot \pi \cdot (.223) + (.0357) \cdot \cos(29.934)}{\pi \cdot (.223) \cdot \cos(29.934) - (.35) \cdot (.0357)}\right] = (2.12) \cdot \left(\frac{0.2454 + .03094}{.5946}\right)$$

$T = 0.9851 \text{ lb-in} = 15.76 \text{ oz-in}$

Case 2: 1/4-28 Threads Lowering a Load

$$T = \frac{(19)\cdot(0.223)}{2}\left[\frac{(.35)\cdot\pi\cdot(.223)-(.0357)\cdot\cos(29.934)}{\pi\cdot(.223)\cdot\cos(29.934)+(.35)\cdot(.0357)}\right] = (2.12)\cdot\left(\frac{0.2454-.03094}{.6071+.0125}\right)$$

$T = 0.733 \text{ lb-in} = 11.73 \text{ oz-in}$

Case 3: 3/8-32 Threads Raising a Load

$$T = \frac{(19)\cdot(0.350)}{2}\left[\frac{(.35)\cdot\pi\cdot(.350)+(.0313)\cdot\cos(29.982)}{\pi\cdot(.350)\cdot\cos(29.982)-(.35)\cdot(.0313)}\right] = (3.325)\cdot\left(\frac{0.3848+.0271}{.9415}\right)$$

$T = 1.455 \text{ lb-in} = 23.28 \text{ oz-in}$

Case 4: 3/8-32 Threads Lowering a Load

$$T = \frac{(19)\cdot(0.350)}{2}\left[\frac{(.35)\cdot\pi\cdot(.350)-(.0313)\cdot\cos(29.982)}{\pi\cdot(.350)\cdot\cos(29.982)+(.35)\cdot(.0313)}\right] = (3.325)\cdot\left(\frac{0.3848-.0271}{.9634}\right)$$

$T = 1.235 \text{ lb-in} = 19.75 \text{ oz-in}$

From the nature of the differential screw motion, there are two possible scenarios in which torque is applied to the system. They are shown as follows:

1/4-28 Raises Load ↑ while 3/8-32 Lowers Load ↓.

For this combination, the combined torque equals

$$T_{tot} = 15.76 + 19.75 = 35.51 \text{ oz-in.}$$

1/4-28 Lowers Load ↓ while 3/8-32 Raises Load ↑.

For this combination, the combine torque equals

$$T_{tot} = 11.73 + 23.28 = 35.01 \text{ oz-in.}$$

The worst-case torque configuration is when 1/4-28 Raises while the 3/8-32 Lowers the Load. The resisting torque that the motor must overcome is ~36 oz-in.

SHEAR STRESS CALCULATIONS

The only component that is placed in shear during operation of the mechanism is the differential screw. Since this part has both internal and external threads, the total shear stress that the part experiences is a summation of the shear stress along the internal threads and of the stress along the external threads. Thus, the total shear stress may be expressed as:

$$\tau_{TOT} = \tau_{ID} + \tau_{OD} = \frac{T \cdot r_{ID}}{J_{ID}} + \frac{T \cdot r_{OD}}{J_{OD}}$$

where:

T = system torque.
τ_{ID} = shear stress experienced along internal threads.
τ_{OD} = shear stress experienced along external threads.
r_{ID} = mean radius of internal threads.
r_{OD} = mean radius of internal threads.
J_{ID} = moment of inertia for internal threads.
J_{OD} = moment of inertia for external threads.

Substituting in actual values for lowering the plate, considering the 1/4-28 threads:

$$\tau_{ID} = \frac{T \cdot r_{ID}}{J_{ID}} = \frac{(11.73) \cdot (0.1153) \cdot (32)}{\pi \cdot (0.2268)^4 \cdot (16)} = \frac{43.279}{0.1330} = 325.4 \text{ psi}$$

For the external threads (3/8-32), considering lowering of the plate:

$$\tau_{OD} = \frac{T \cdot r_{OD}}{J_{OD}} = \frac{(23.28) \cdot (0.1782) \cdot (32)}{\pi \cdot (0.3547)^4 \cdot (16)} = \frac{132.752}{0.796} = 166.85 \text{ psi}$$

Thus, the total shear stress is (325.4) + (166.85) = <u>492.3 psi</u> for the case of lowering the upper plate of the mechanism.

Substituting in actual values for raising the plate, considering the 1/4-28 threads:

$$\tau_{ID} = \frac{T \cdot r_{ID}}{J_{ID}} = \frac{(15.76) \cdot (0.1153) \cdot (32)}{\pi \cdot (0.2268)^4 \cdot (16)} = \frac{58.148}{0.1330} = 437.2 \text{ psi}$$

For the external threads (3/8-32), considering raising of the plate:

$$\tau_{OD} = \frac{T \cdot r_{OD}}{J_{OD}} = \frac{(19.75)\cdot(0.1782)\cdot(32)}{\pi\cdot(0.3547)^4\cdot(16)} = \frac{112.622}{0.796} = 141.49 \text{ psi}$$

Thus, the total shear stress is (437.2) + (141.49) = 578.7 psi for the case of raising the upper plate of the mechanism.

Factor of Safety Calculation

From Spotts[6], the maximum shear stress theory of failure was used to calculate the factor of safety for the above-calculated shear stress values. The maximum shear stress theory of failure may be stated as follows:

$$FS = \frac{0.5\cdot\sigma_{YP}}{\tau_{MAX}}$$

where:

FS = Factor of Safety for system.

σ_{yp} = Yield Point of the material under investigation. (10 ksi for brass[5]).

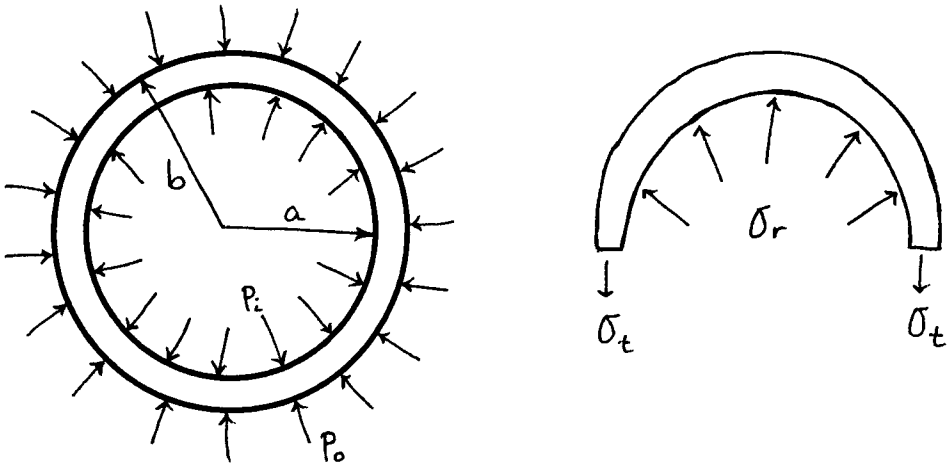
τ_{MAX} = Maximum shear stress for system under investigation.

Since the differential screw is in pure torsion with no bending, τ_{MAX} is assumed to be the same value as τ_{TOT}. Thus, substituting the greater of the two values calculated above, a factor of safety may be calculated for the system in worst-case loading.

$$FS = \frac{0.5\cdot\sigma_{YP}}{\tau_{MAX}} = \frac{(0.5)\cdot(10{,}000)}{(579)} = 8.6$$

HOOP STRESS CALCULATIONS

In order to allow flexibility for pivoting motion by the tips of the driveshafts against their bearing surface, the lower plate, brass balls were chosen to be inserted into the ends of the driveshafts. However, the outer diameters of the driveshafts must remain essentially unchanged so that a binding condition is avoided at the interface between the driveshafts and the brass bushings. Hoop stress calculations are used to quantify the amount that the driveshafts will grow as a result of pressing in the brass balls. Consider the figure below:



For the configuration used in this development, $p_0 = 0$. The pressure exerted against the internal walls, p_0 , is named p and is a function of the diametral interference. Only the radial stress, σ_r will be considered in this development. From Spotts[6], the equations for the radial stress is defined as:

$$\sigma_r = \frac{a^2 p}{b^2 - a^2} \cdot \left(1 - \frac{b^2}{a^2}\right)$$

where:

a = internal radius of cylinder
b = external radius of cylinder
p = internal pressure exerted onto cylinder

The diametral interference may be defined as a function of the internal pressure. By specifying the interference, the pressure may be calculated and substituted into an equation for the increase in hole size. The diametral interference, Δ, is given by

$$\Delta = \frac{4 \cdot a \cdot p}{E} \cdot \left[\frac{1}{1 - \left(a/b \right)^2} \right]$$

where E is the modulus of elasticity for the cylinder. Solving for p from this equation,

$$p = \frac{E \cdot \Delta}{4 \cdot a} \cdot \left[1 - \left(a/b \right)^2 \right]$$

With p known, the increase in hole size is defined as u_h by the following equation:

$$u_h = \frac{a \cdot p}{E} \cdot \left[\frac{1 + \left(a/b \right)^2}{1 - \left(a/b \right)^2} + \mu \right]$$

where μ is Poisson's ratio and is assumed[5] to be 0.3 (brass) for the inserted balls.

Sample calculations of the above quantities are shown below. A spreadsheet was constructed to study the effects of different interference fits between the ball and cylinder. Sample of output of the spreadsheet is shown on the following page. Considering an interference fit where Δ = .002 in, a = .125 in, b = .1875 in, and E = 30 ksi

$$p = \frac{E \cdot \Delta}{4 \cdot a} \cdot \left[1 - \left(a/b \right)^2 \right] = \frac{(30{,}000) \cdot (.002)}{(4) \cdot (.125)} \cdot \left[1 - \left(\frac{.125}{.1875} \right)^2 \right] = 120 \cdot (.556) = 66.7 psi$$

Substituting this value of p into the u_h equation,

$$u_h = \frac{a \cdot p}{E} \cdot \left[\frac{1 + \left(a/b \right)^2}{1 - \left(a/b \right)^2} + \mu \right] = \frac{(.125) \cdot (66.7)}{30{,}000} \cdot \left[\frac{1 + \left(.125/.1875 \right)^2}{1 - \left(.125/.1875 \right)^2} + (.3) \right]$$

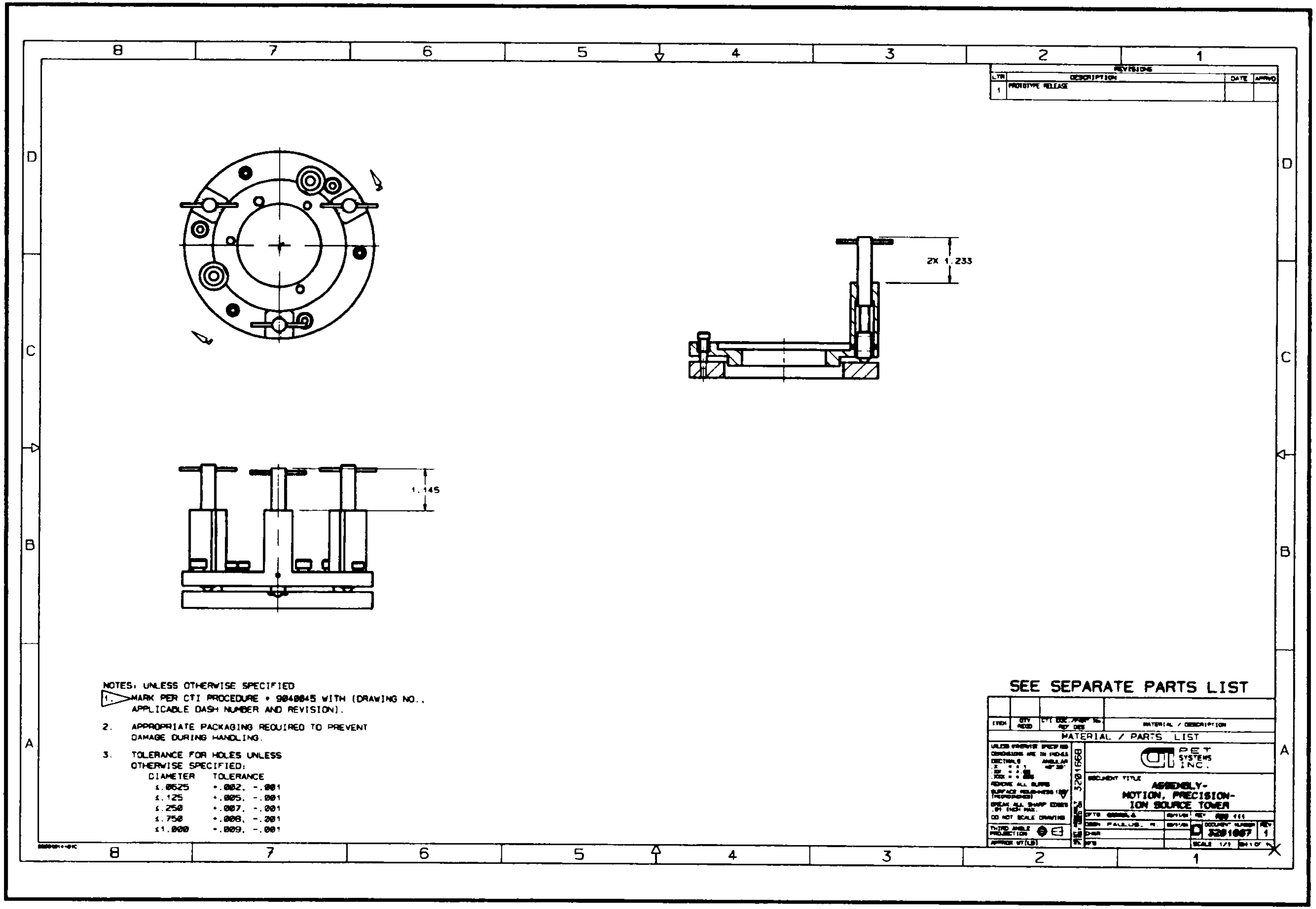
$$= (2.78\text{E} - 4) \cdot (1.79) = .00049 \text{ in}$$

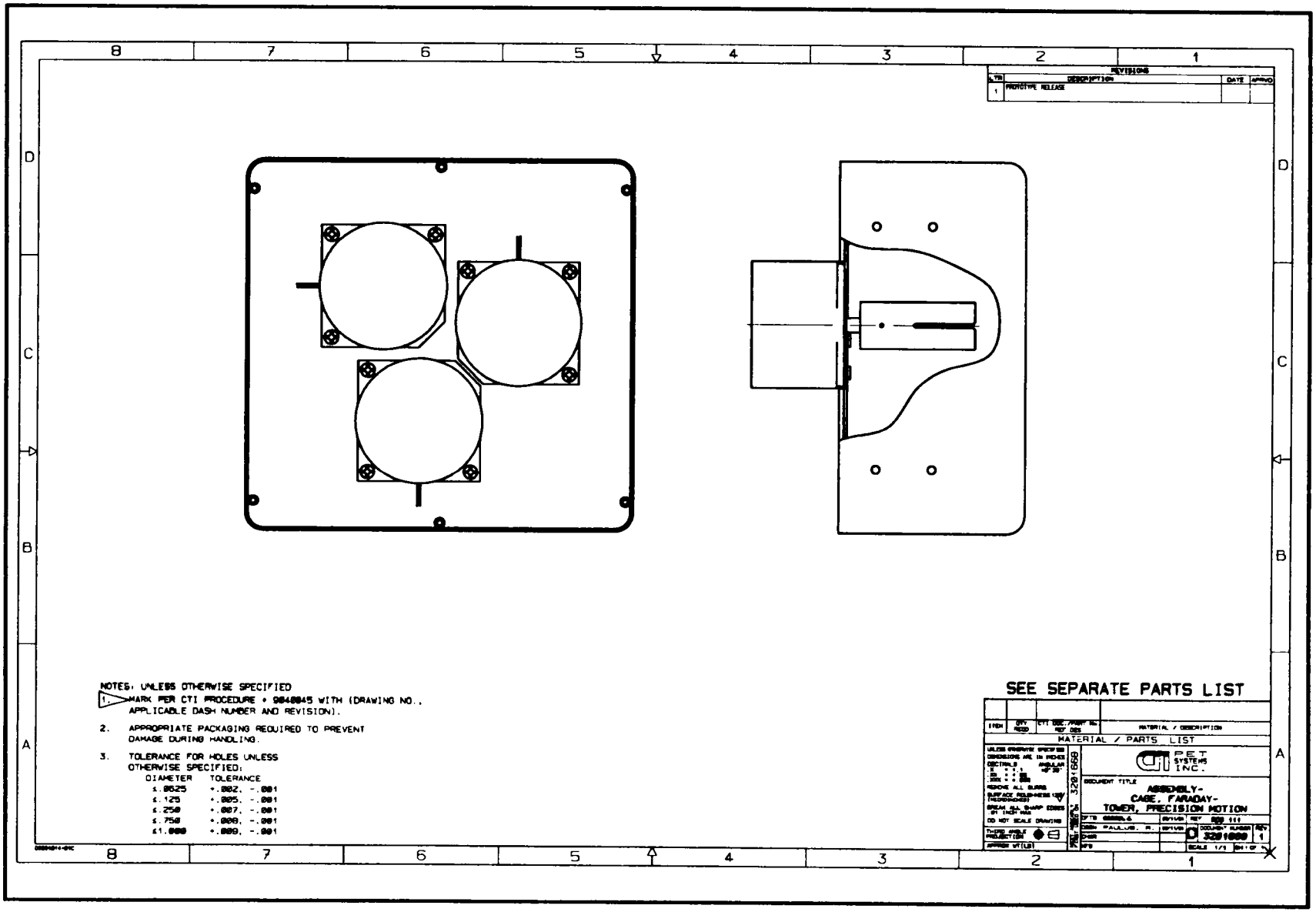
Thus, for an interference of .002 in, the diameter of the stainless steel cylinder will increase by .0005 in. What follows is sample output from a spreadsheet that was used to select the final configuration.

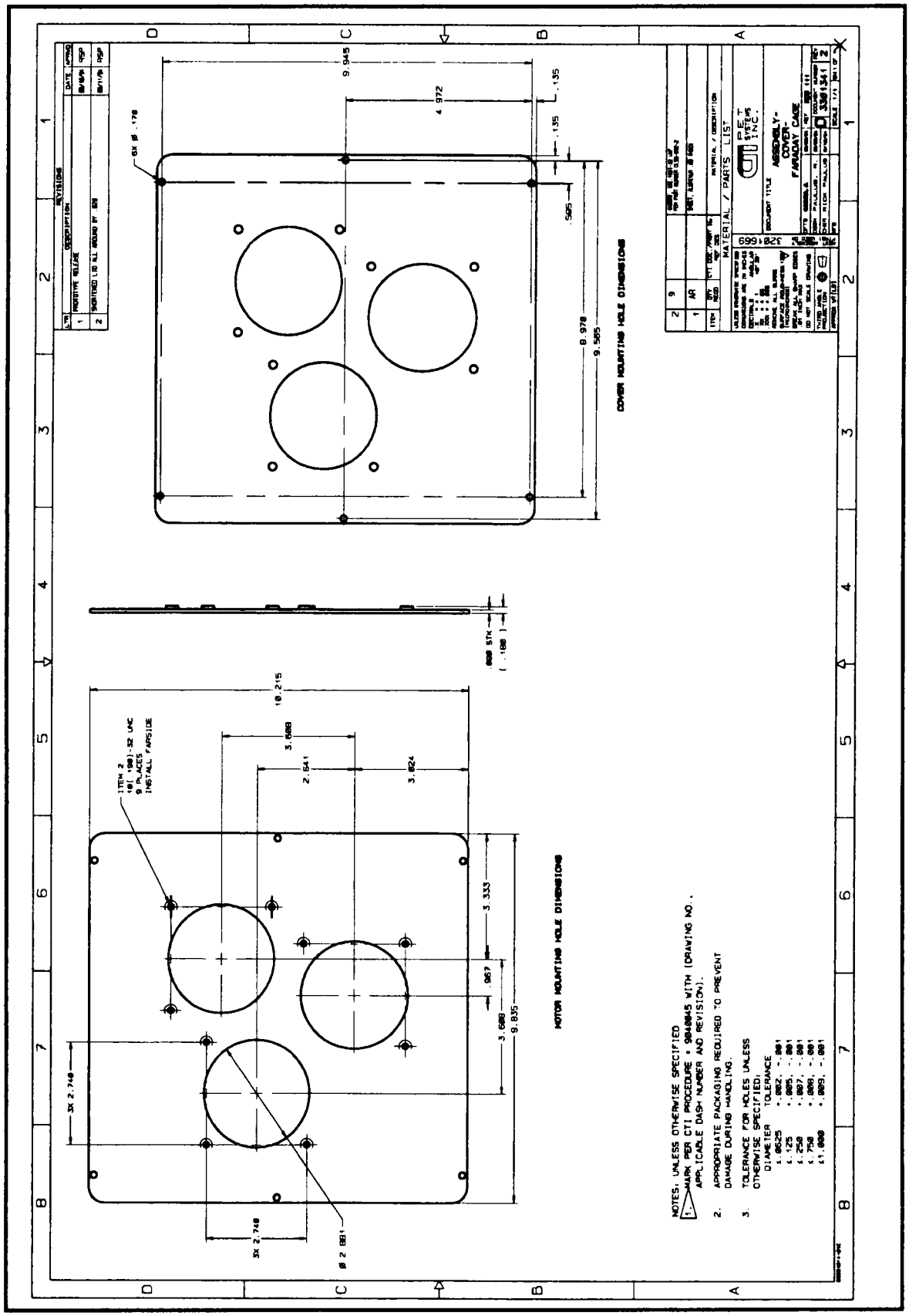
Brass Ball Sizing Calculations for 3/8" OD SST shaft					
ID (in)	0.25				
OD (in)	0.375				
a (ID/2)	0.125	Esst (psi)	30,000	(a/b)^2	0.444444
b (OD/2)	0.1875	mu (brass)	0.34	ap/Eh	0.000208
delta (int.)	0.0015			[ ]	1.832308
p	50			Uh	0.000382
Uh	0.000382				
ID (in)	0.25				
OD (in)	0.375				
a (ID/2)	0.125	Esst (psi)	30,000	(a/b)^2	0.444444
b (OD/2)	0.1875	mu (brass)	0.34	ap/Eh	0.000139
delta (int.)	0.001			[ ]	1.832308
p	33.33333			Uh	0.000254
Uh	0.000254				
ID (in)	0.25				
OD (in)	0.375				
a (ID/2)	0.125	Esst (psi)	30,000	(a/b)^2	0.444444
b (OD/2)	0.1875	mu (brass)	0.34	ap/Eh	0.000417
delta (int.)	0.003			[ ]	1.832308
p	100			Uh	0.000763
Uh	0.000763				

APPENDIX B

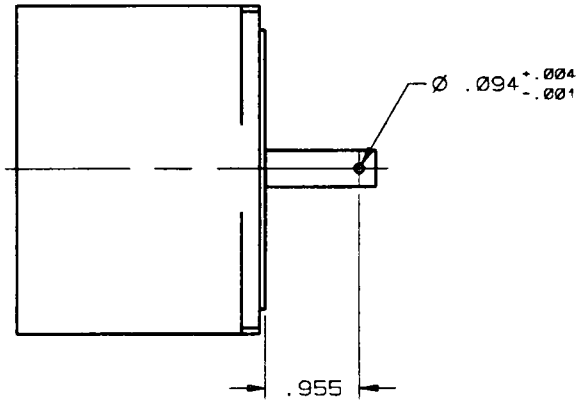
ENGINEERING DRAWINGS

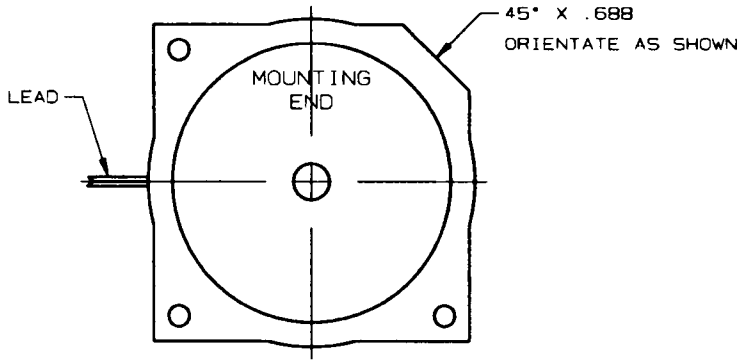

REVISIONS
LTR DESCRIPTION DATE APPRVD
1 PROTOTYPE RELEASE
2X 1.233
1.145
NOTES: UNLESS OTHERWISE SPECIFIED
1. MARK PER CTI PROCEDURE # 9040045 WITH (DRAWING NO., APPLICABLE DASH NUMBER AND REVISION).
2. APPROPRIATE PACKAGING REQUIRED TO PREVENT DAMAGE DURING HANDLING.
3. TOLERANCE FOR HOLES UNLESS OTHERWISE SPECIFIED:
DIAMETER TOLERANCE
≤.0625 +.002, -.001
≤.125 +.005, -.001
≤.250 +.007, -.001
≤.750 +.008, -.001
≤1.000 +.009, -.001
SEE SEPARATE PARTS LIST
MATERIAL / PARTS LIST
3201668
PET SYSTEMS INC.
ASSEMBLY-
MOTION, PRECISION-
ION SOURCE TOWER
3201667
1


REVISIONS
DESCRIPTION
PROTOTYPE RELEASE
NOTES: UNLESS OTHERWISE SPECIFIED
1. MARK PER CTI PROCEDURE # 9840845 WITH (DRAWING NO., APPLICABLE DASH NUMBER AND REVISION).
2. APPROPRIATE PACKAGING REQUIRED TO PREVENT DAMAGE DURING HANDLING.
3. TOLERANCE FOR HOLES UNLESS OTHERWISE SPECIFIED:
DIAMETER TOLERANCE
≤.0625 +.002, -.001
≤.125 +.005, -.001
≤.250 +.007, -.001
≤.750 +.008, -.001
≤1.000 +.009, -.001
SEE SEPARATE PARTS LIST
MATERIAL / PARTS LIST
ITEM
QTY. REQD
MATERIAL / DESCRIPTION
PET SYSTEMS INC.
DOCUMENT TITLE
ASSEMBLY-
CAGE, FARADAY-
TOWER, PRECISION MOTION
3201668
SCALE 1/1


COVER MOUNTING HOLE DIMENSIONS
MOTOR MOUNTING HOLE DIMENSIONS
NOTES: UNLESS OTHERWISE SPECIFIED
1. MARK PER CTI PROCEDURE # 9840845 WITH (DRAWING NO. - APPLICABLE DASH NUMBER AND REVISION).
2. APPROPRIATE PACKAGING REQUIRED TO PREVENT DAMAGE DURING HANDLING.
3. TOLERANCE FOR HOLES UNLESS OTHERWISE SPECIFIED:
DIAMETER TOLERANCE
PET SYSTEMS INC.
ASSEMBLY- COVER- FARADAY CAGE
MATERIAL / PARTS LIST

REVISIONS			
LTR	DESCRIPTION	DATE	APRVD
1	PROTOTYPE RELEASE	08/10/94	RSP





ALTERED ITEM DRAWING

NOTES: UNLESS OTHERWISE SPECIFIED

1. MARK PER CTI PROCEDURE • 9040045 WITH (DRAWING NO., APPLICABLE DASH NO. AND REVISION).
2. APPROPRIATE PACKAGING REQUIRED TO PREVENT DAMAGE DURING HANDLING.

ITEM	QTY REQD	CTI DOC./PART No. REF DES	STEP MOTOR, HYBRID APPLIED MOTION PRODUCTS PART NUMBER 4034-322 MATERIAL / DESCRIPTION

MATERIAL / PARTS LIST

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
DECIMALS .X = ±.1 .XX = ±.03 .XXX = ±.005
ANGULAR ±0°30'
REMOVE ALL BURRS
SURFACE ROUGHNESS 125 (MICROINCHES)
BREAK ALL SHARP EDGES .01 INCH MAX.
DO NOT SCALE DRAWING
THIRD ANGLE PROJECTION
APPROX WT(LB)

NEXT ASSEMBLY 3201669 FIRST USED ON

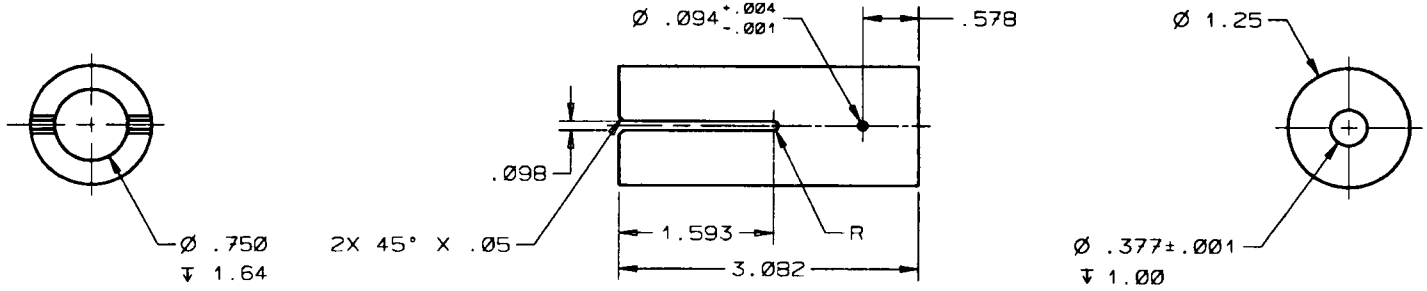
CTI PET SYSTEMS INC.

DOCUMENT TITLE
ADJUSTMENT MOTOR-
FARADAY CAGE

DFTG	GARCIA, J.	08/09/94	REF	RDS 111
CSGN	PAULUS, R.	08/09/94	B	DOCUMENT NUMBER 1403888 REV 1
CHKR	RICK PAULUS	8/10/94		
MFG			SCALE	1/1 SH 1 OF 1

B9691812-01C

REVISIONS			
LTR	DESCRIPTION	DATE	APRVD
1	PROTOTYPE RELEASE	08/10/94	RSP



NOTES: UNLESS OTHERWISE SPECIFIED

1. MARK PER CTI PROCEDURE • 9040045 WITH (DRAWING NO., APPLICABLE DASH NO. AND REVISION).
2. APPROPRIATE PACKAGING REQUIRED TO PREVENT DAMAGE DURING HANDLING.
3. TOLERANCE FOR HOLES UNLESS OTHERWISE SPECIFIED:

DIAMETER	TOLERANCE
≤.0625	+.002, -.001
≤.125	+.005, -.001
≤.250	+.007, -.001
≤.750	+.008, -.001
≤1.000	+.009, -.001

ITEM	QTY REQD	CTI DOC./PART No. REF DES	MATERIAL / DESCRIPTION
			DELRIN

MATERIAL / PARTS LIST

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
DECIMALS .X ± .1 .XX ± .03 .XXX ± .005 ANGULAR ±0°30'
REMOVE ALL BURRS
SURFACE ROUGHNESS 125 (MICROINCHES)
BREAK ALL SHARP EDGES .01 INCH MAX.
DO NOT SCALE DRAWING
THIRD ANGLE PROJECTION
APPROX WT(LB)

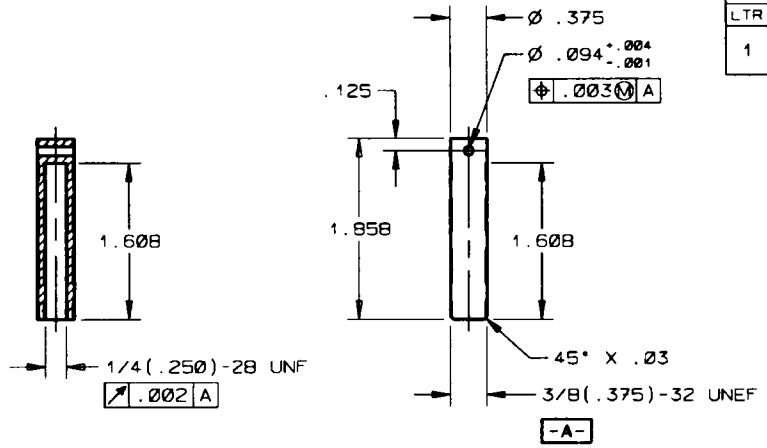
NEXT ASSEMBLY USED ON 3201668

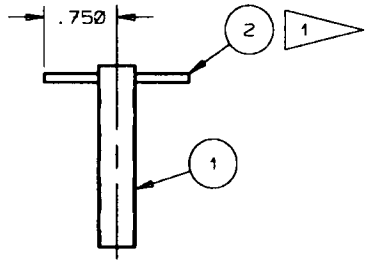
PET SYSTEMS INC.

DOCUMENT TITLE
COUPLING, DRIVE LINE-
FARADAY CAGE

DFTG	GARCIA, J.	08/09/94	REF	RDS 111
DSGN	PAULUS, R.	08/09/94	B	DOCUMENT NUMBER 1403889 REV 1
CHKR	RICK PAULUS	8/10/94		
MFG			SCALE	1/1 SH 1 OF 1

B9691012-01C





REVISIONS			
LTR	DESCRIPTION	DATE	APRVD
1	PROTOTYPE RELEASE	08/10/94	RSP

NOTES: UNLESS OTHERWISE SPECIFIED

1. INSTALL ITEM 2 INTO ITEM 1.
2. CLEAN AND DEGREASE ENTIRE PART AFTER ASSEMBLY.
3. BAG AND TAG WITH CTI PART NUMBER AND LATEST REVISION LEVEL.
4. APPROPRIATE PACKAGING REQUIRED TO PREVENT DAMAGE DURING HANDLING.

ITEM	QTY REQD	CTI DOC./PART No. REF DES	MATERIAL / DESCRIPTION
2	1		COIL SPRING PIN, CARBON STEEL .094 X 1.50 LONG McMASTER CARR PART NUMBER 91595A216 OR EQUIVALENT
1	A/R		BRASS

MATERIAL / PARTS LIST

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
DECIMALS: .X = ±.1, .XX = ±.03, .XXX = ±.005
ANGULAR ±0°30'
REMOVE ALL BURRS
SURFACE ROUGHNESS 125 (MICROINCHES)
BREAK ALL SHARP EDGES .01 INCH MAX.
DO NOT SCALE DRAWING
THIRD ANGLE PROJECTION
APPROX WT(LB)

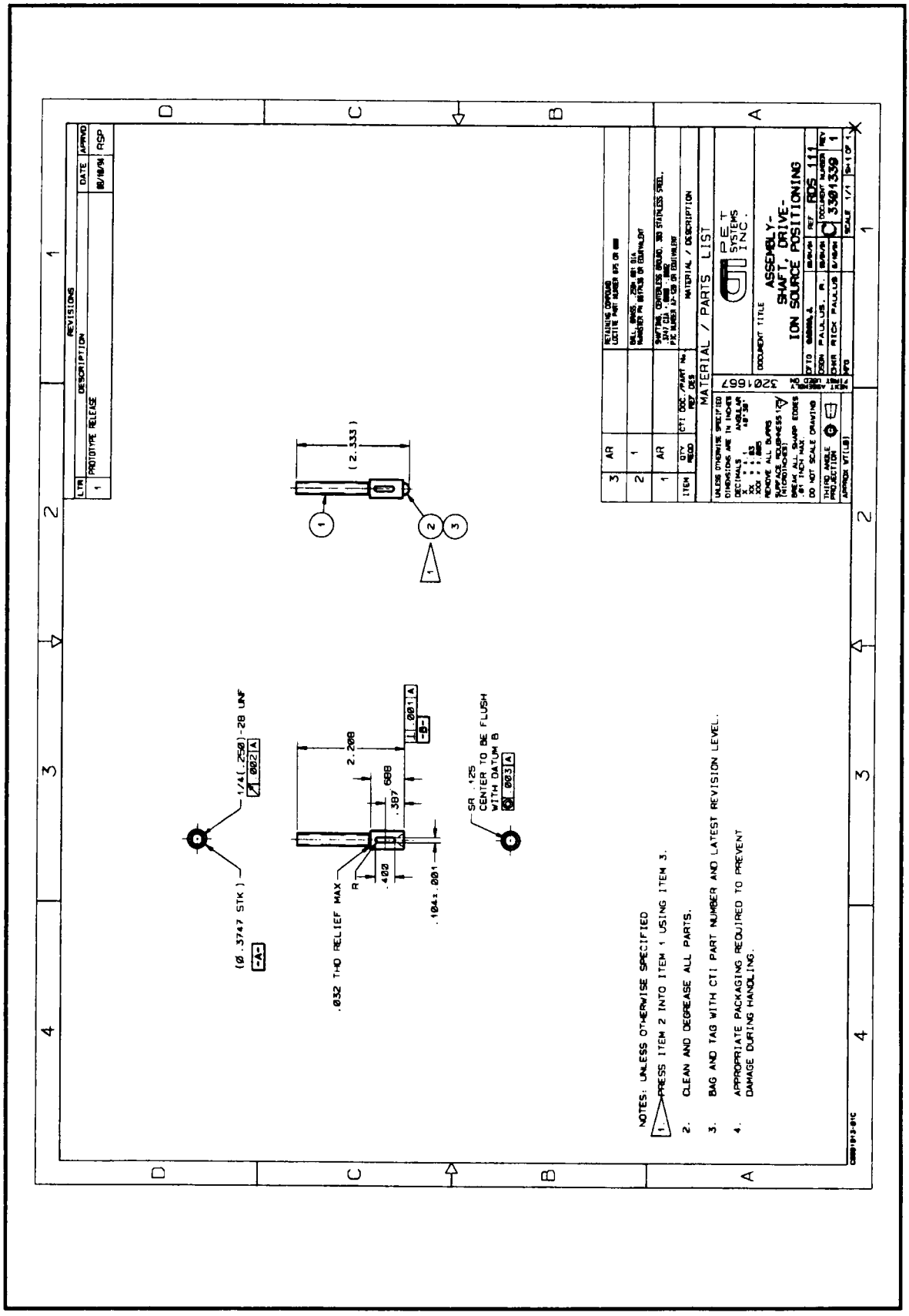
NEXT ASSEMBLY 3201667
FIRST USED ON

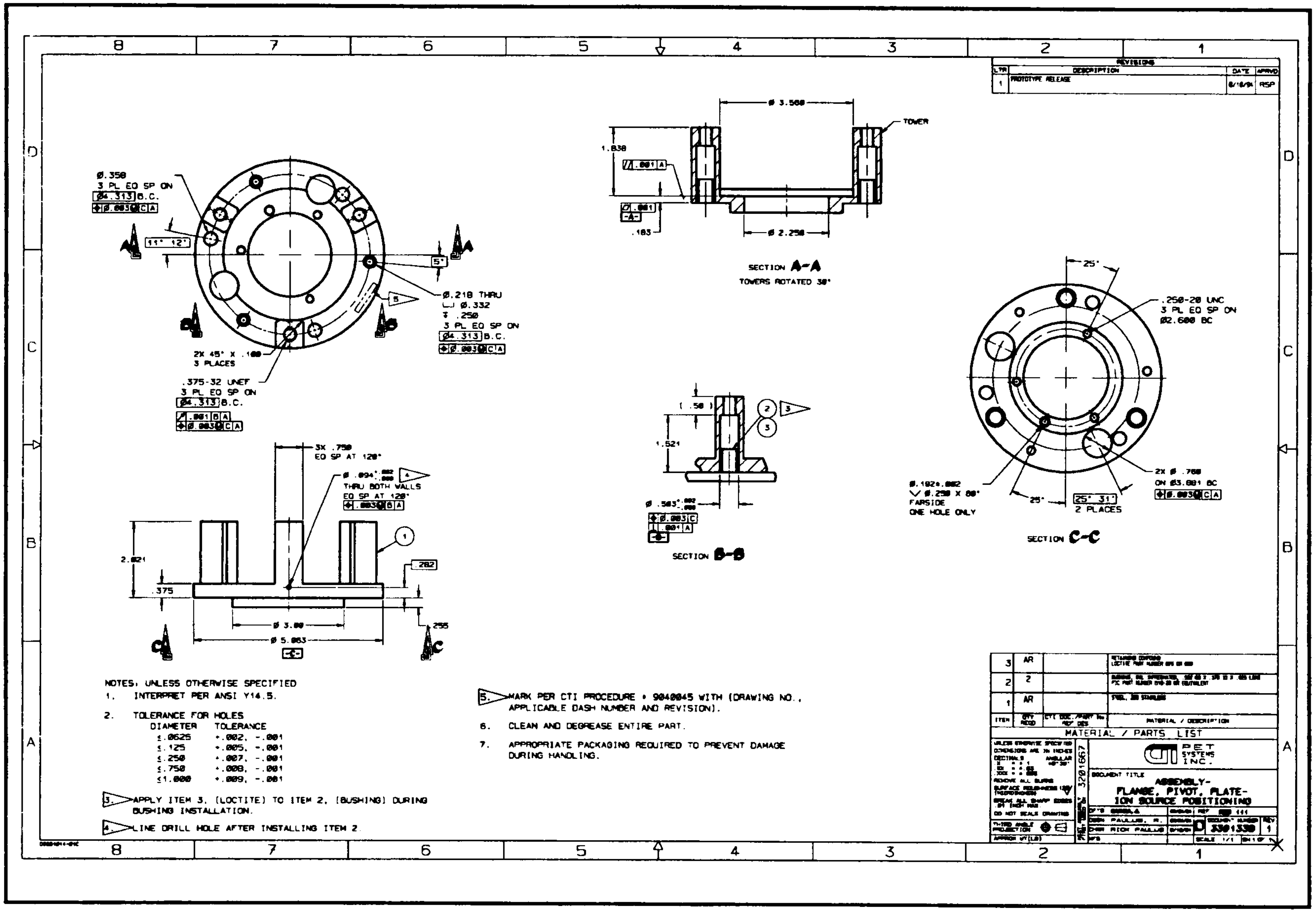
CTI PET SYSTEMS INC.

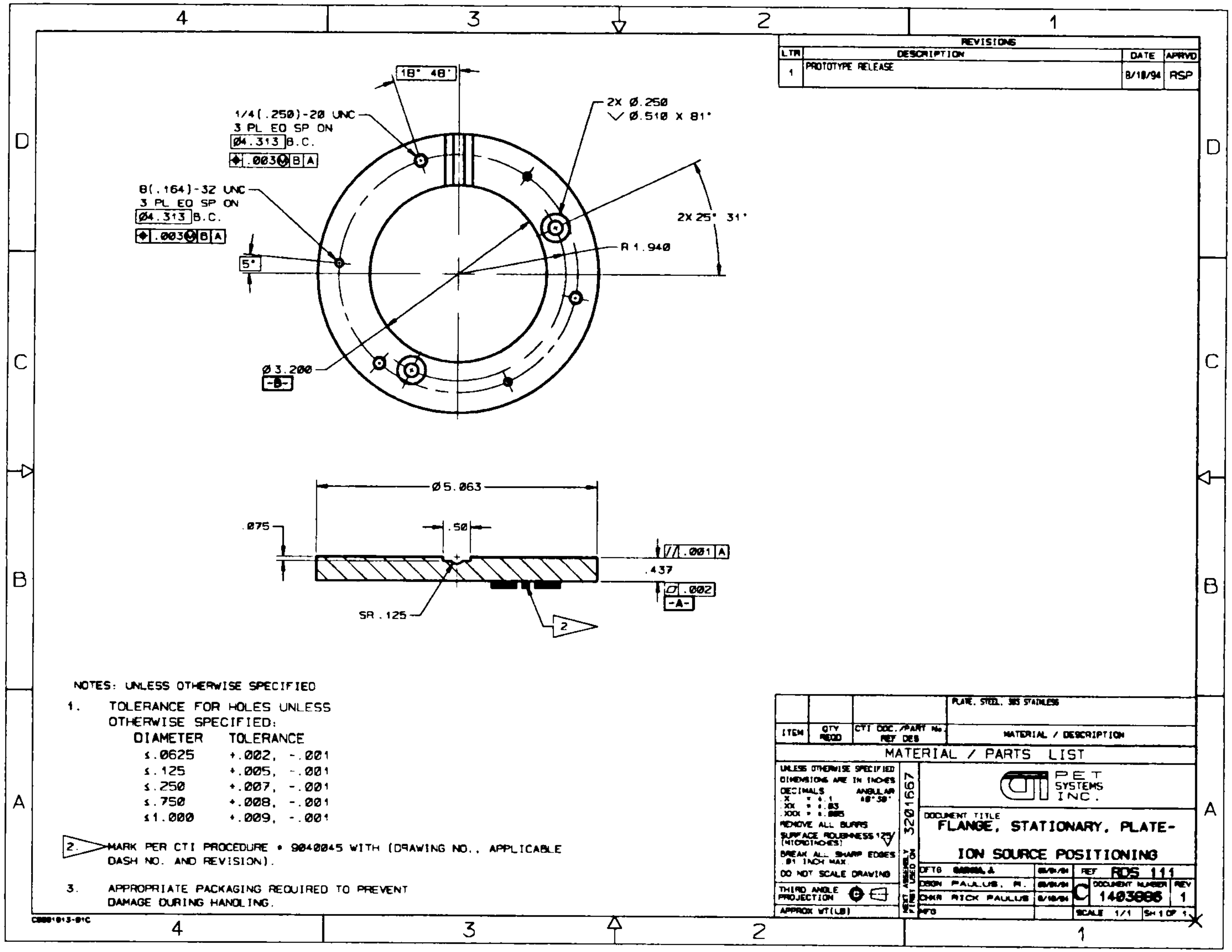
DOCUMENT TITLE
ASSEMBLY-
DIFFERENTIAL SCREW-
ION SOURCE POSITIONING

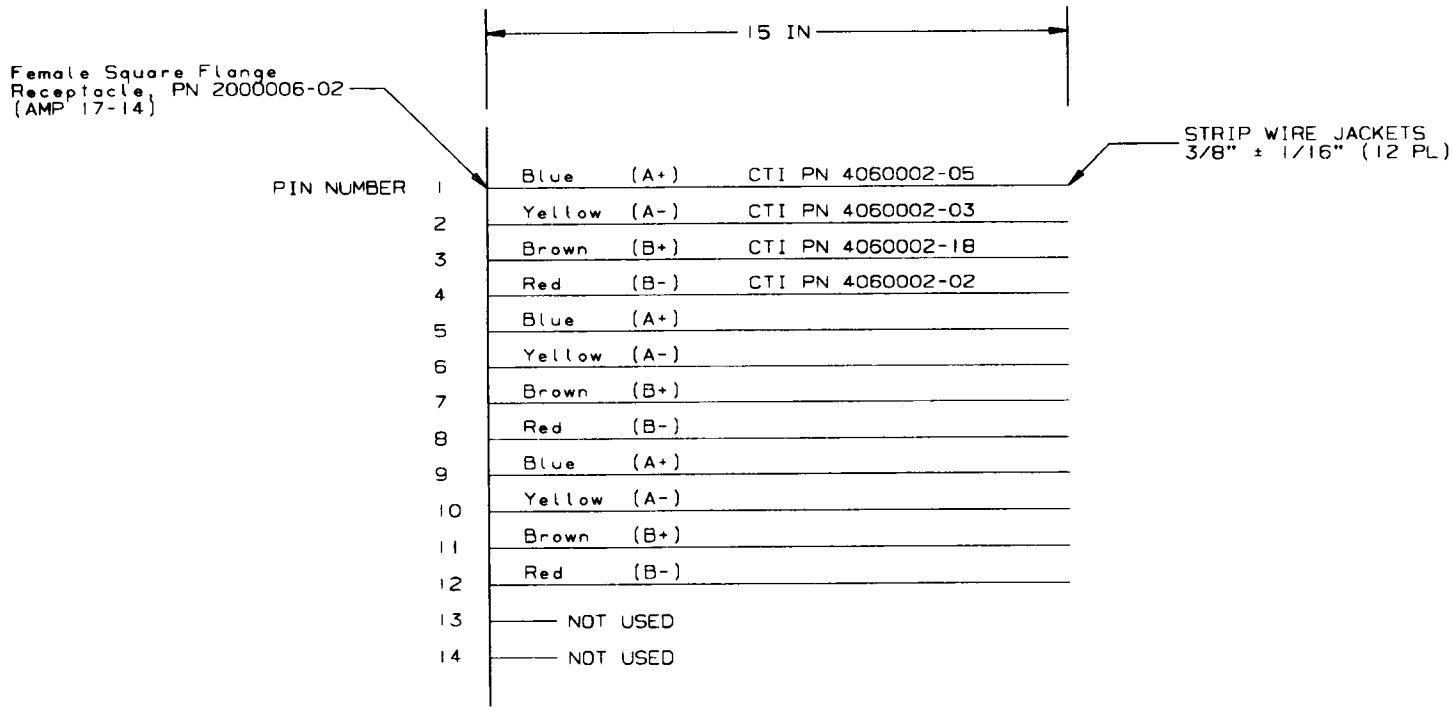
DFTG	GARCIA, J.	08/05/94	REF	RDS 111
DSGN	PAULUS, R.	08/05/94	B	DOCUMENT NUMBER 3301340 REV 1
CHKR	RICK PAULUS	8/10/94		
MFG			SCALE 1/1	SH 1 OF 1

B9691012-01C


REVISIONS
LTR
DESCRIPTION
DATE
APPRVD
1
PROTOTYPE RELEASE
10/18/94
RSP
MATERIAL / PARTS LIST
ITEM
QTY REQD
PART NO.
MATERIAL / DESCRIPTION
3
AR
2
1
1
AR
ASSEMBLY-
SHAFT, DRIVE-
ION SOURCE POSITIONING
PET SYSTEMS INC.
ROS 111
3361339
SCALE 1/1
SHEET 1 OF 1
(2.333)
1/4(.250)-28 UNF
.002 A
(Ø.3747 STK)
-A-
.032 THD RELIEF MAX
2.208
.688
.387
.402
.104±.001
.001 A
-B-
SR .125
CENTER TO BE FLUSH
WITH DATUM B
.003 A
NOTES: UNLESS OTHERWISE SPECIFIED
1. PRESS ITEM 2 INTO ITEM 1 USING ITEM 3.
2. CLEAN AND DEGREASE ALL PARTS.
3. BAG AND TAG WITH CTI PART NUMBER AND LATEST REVISION LEVEL.
4. APPROPRIATE PACKAGING REQUIRED TO PREVENT DAMAGE DURING HANDLING.


REVISIONS
LTR DESCRIPTION DATE APPRVD
1 PROTOTYPE RELEASE RSP
SECTION A-A
TOWERS ROTATED 30°
TOWER
SECTION B-B
SECTION C-C
.250-20 UNC
3 PL EQ SP ON
Ø2.600 BC
2X Ø .760
ON Ø3.001 BC
25° 31'
2 PLACES
Ø.350
3 PL EQ SP ON
Ø4.313 B.C.
Ø.218 THRU
Ø.332
.250
3 PL EQ SP ON
Ø4.313 B.C.
2X 45° X .100
3 PLACES
.375-32 UNEF
3 PL EQ SP ON
Ø4.313 B.C.
3X .750
EQ SP AT 120°
THRU BOTH WALLS
EQ SP AT 120°
FARSIDE
ONE HOLE ONLY
NOTES, UNLESS OTHERWISE SPECIFIED
1. INTERPRET PER ANSI Y14.5.
2. TOLERANCE FOR HOLES
DIAMETER TOLERANCE
≤.0625 +.002, -.001
≤.125 +.005, -.001
≤.250 +.007, -.001
≤.750 +.008, -.001
≤1.000 +.009, -.001
3. APPLY ITEM 3, (LOCTITE) TO ITEM 2, (BUSHING) DURING BUSHING INSTALLATION.
4. LINE DRILL HOLE AFTER INSTALLING ITEM 2.
5. MARK PER CTI PROCEDURE # 9040045 WITH (DRAWING NO., APPLICABLE DASH NUMBER AND REVISION).
6. CLEAN AND DEGREASE ENTIRE PART.
7. APPROPRIATE PACKAGING REQUIRED TO PREVENT DAMAGE DURING HANDLING.
MATERIAL / PARTS LIST
CTI PET SYSTEMS INC.
ASSEMBLY-
FLANGE, PIVOT, PLATE-
ION SOURCE POSITIONING
3201667
THIRD ANGLE PROJECTION
RICH PAULUS


REVISIONS
LTR | DESCRIPTION | DATE | APPRVD
1 | PROTOTYPE RELEASE | 8/18/94 | RSP
18° 48'
1/4(.250)-20 UNC
3 PL EQ SP ON
Ø4.313 B.C.
.003 B A
8(.164)-32 UNC
3 PL EQ SP ON
Ø4.313 B.C.
.003 B A
5°
2X Ø.250
Ø.510 X 81°
2X 25° 31'
R 1.940
Ø 3.200
-B-
Ø 5.063
.075
.50
.001 A
.437
.002
-A-
SR .125
2
NOTES: UNLESS OTHERWISE SPECIFIED
1. TOLERANCE FOR HOLES UNLESS OTHERWISE SPECIFIED:
DIAMETER TOLERANCE
≤.0625 +.002, -.001
≤.125 +.005, -.001
≤.250 +.007, -.001
≤.750 +.008, -.001
≤1.000 +.009, -.001
2. MARK PER CTI PROCEDURE # 9040045 WITH (DRAWING NO., APPLICABLE DASH NO. AND REVISION).
3. APPROPRIATE PACKAGING REQUIRED TO PREVENT DAMAGE DURING HANDLING.
PLATE, STEEL, 303 STAINLESS
ITEM | QTY REQD | CTI DOC./PART No REF DES | MATERIAL / DESCRIPTION
MATERIAL / PARTS LIST
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
REMOVE ALL BURRS
DO NOT SCALE DRAWING
THIRD ANGLE PROJECTION
APPROX WT(LB)
3201667
CTI PET SYSTEMS INC.
DOCUMENT TITLE
FLANGE, STATIONARY, PLATE-
ION SOURCE POSITIONING
DSGN PAULUS, R.
CHKR RICK PAULUS
REF RDS 111
C
DOCUMENT NUMBER
1403886
REV 1
SCALE 1/1
SH 1 OF 1



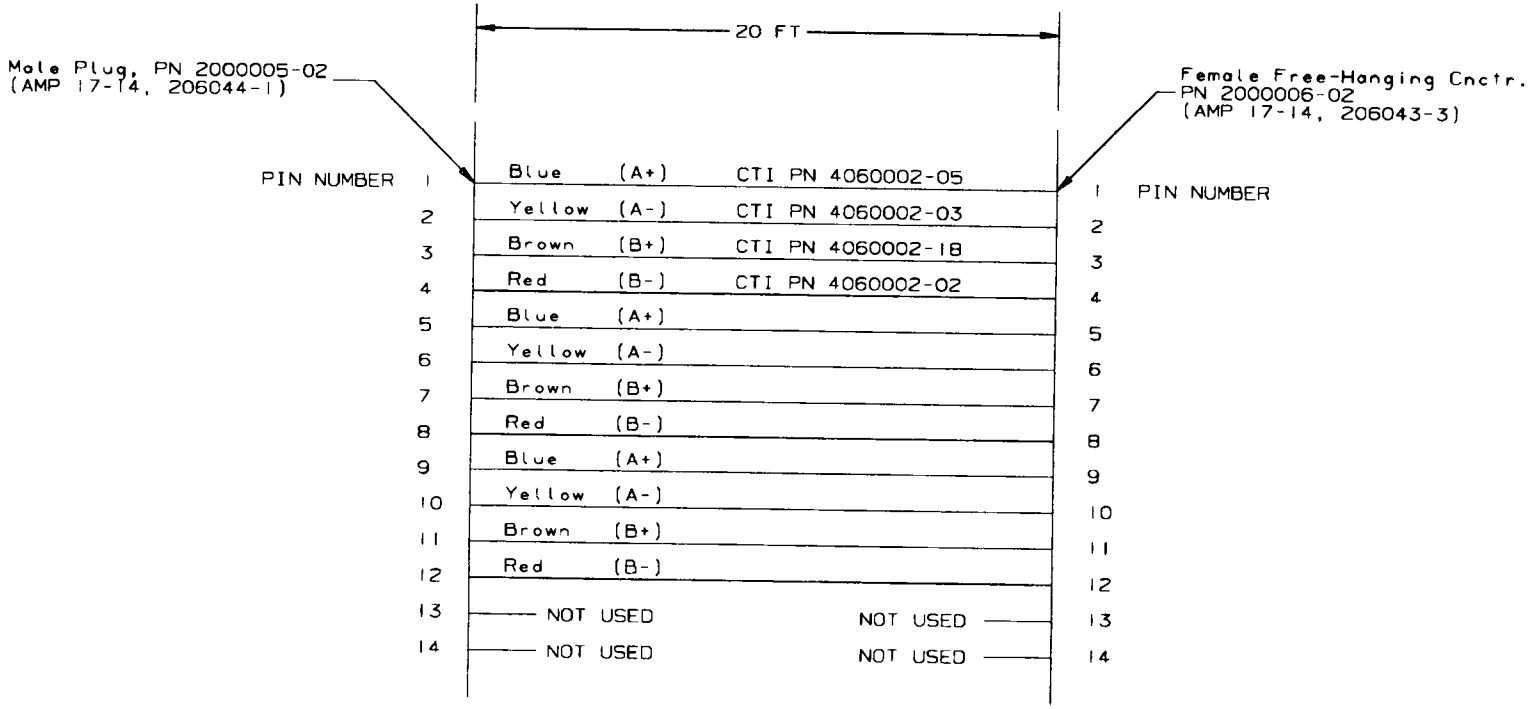
NOTES:

1. Use contact sockets, PN 2110014-04, 12 PL.
2. Use heat shrink tubing for strain relief if wires will not fit into back of connector.

TITLE: CABLE ASSY, BOX-DRIVER CARDS, 15 IN

DATE: 9-19-94

ENGINEER: RICK PAULUS x236



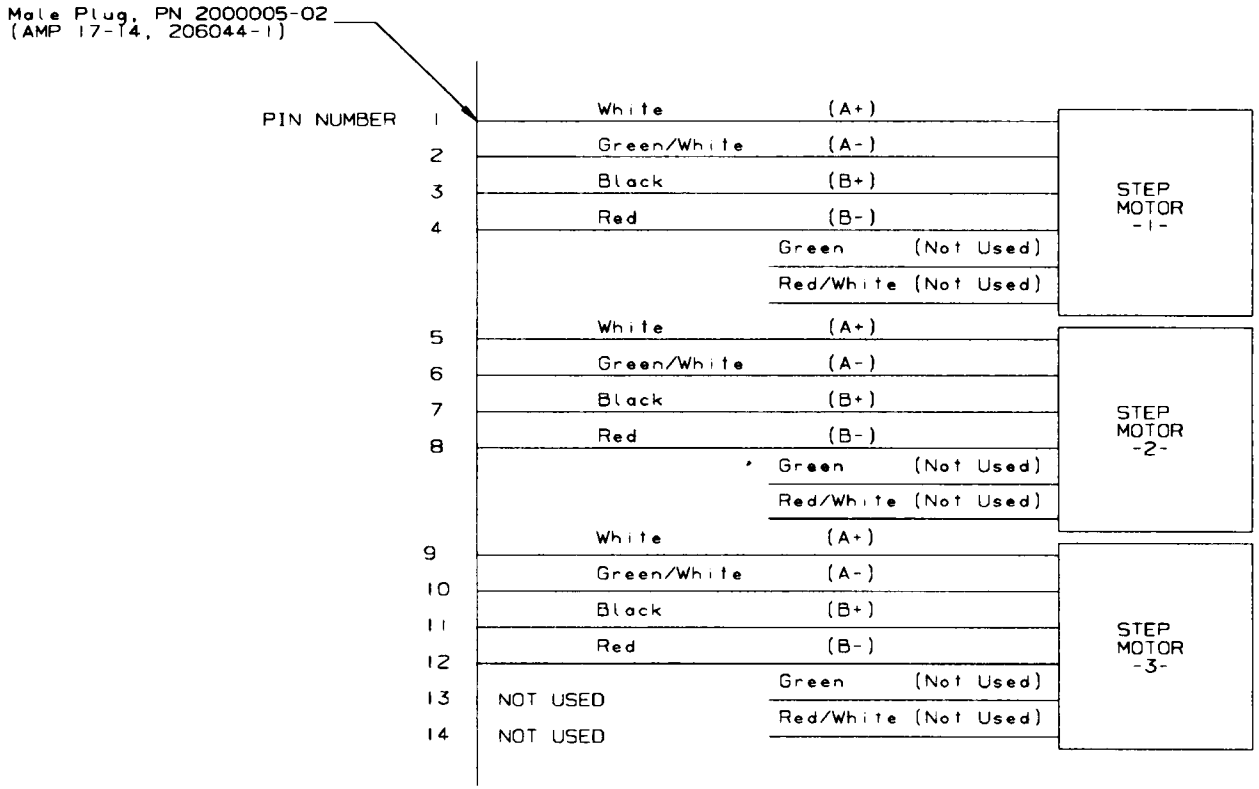
NOTES:

1. Use contact pins, PN 2110016-04, 12 PL.
2. Use contact sockets, PN 2110014-04, 12 PL.
3. Use braided sleeving around all wires to form cable assembly.
4. Use heat shrink tubing if all 12 conductors will not fit into back of connector.

TITLE: CABLE ASSY, MOTOR PULSING, 20 FT

DATE: 9-19-94

ENGINEER: RICK PAULUS x236



NOTES:

1. Use contact pins, PN 2110016-04, 12 PL.
2. Use heat shrink tubing for strain relief if wires will not fit into back of connector.

TITLE: CABLE ASSY, STEP MOTOR PIGTAIL

DATE: 9-19-94

ENGINEER: RICK PAULUS x236

APPENDIX C

SOFTWARE SOURCE CODE

&

MICROSECOND DELAY SOFTWARE REFERENCE INFORMATION

```
/*****************************************************************************
**                                                                          **
**      CONTROL PROGRAM FOR ION SOURCE POSITIONING MECHANISM                **
**                                                                          **
**      Engineer:       Rick Paulus                                         **
**      Date Started:   8-9-94                                              **
**      Revision:       9A                                                  **
**      Rev. Date:      10-10-94                                            **
**                                                                          **
*****************************************************************************
**                                                                          **
**      SUMMARY OF VARIABLES:                                               **
**                                                                          **
**      i,j,k:                  Count variables.                            **
**      gonogo, startstop:      User-query response variables.              **
**      max():                  Function used to calculate MaxSteps.        **
**                                  Returns maximum of two values.          **
**      val1,val2,maxval:       Variables in max() function.                **
**      initgap,xyap,zap:       Initial values for gap, xy-aperture, and    **
**                                  z-aperture.                             **
**      oldgap,oldxyap,oldzap:  Previous values for gap, xy-ap. and z-ap.   **
**                                  used for calculating incremental moves  **
**                                  from one relative position to the next. **
**      gapval,xyapval,zapval:  User-specified values for gap, xy-ap. and   **
**                                  z-ap.                                   **
**      x, y, z:                Calculated inputs to main algorithm.        **
**                                  Represent displacements in x,y,z of     **
**                                  anode relative to puller.               **
**      x0,y0,z0,c1,c2,c3:      Variables used in initial conversion of     **
**                                  user-specified gapval,xyapval,zapval    **
**                                  to x,y,z values.  Data from CAD values. **
**      alpha,beta:             Calculated angles based on x & y.           **
**      alphadeg,betadeg:       Degree equivalents to alpha,beta.           **
**      Hy,Hx:                  Calculated z-offsets for each set calc's.   **
**      z0:                     Initial value from plane passing through    **
**                                  center-line of o-ring to center-line    **
**                                  of "central region". (c-l puller slit)  **
**      PI:                     Trig. function.  (what you'd expect).       **
**      A1A2:                   Matrix used for coordinate transformation   **
**                                  calculate leg lengths.                  **
**      rad32:                  Calculated value for square-root of 3       **
**                                  divided by two.                         **
**      h:                      The initial gap between the upper and lower **
**                                  plates.  (key variable in calc's)       **
**      r:                      Radius of the bolt circle in which the legs **
**                                  are included.  (initial geometry)       **
**      c1,c2,c3:               Constants used in the initial conversion    **
**                                  of user-specified gap, xy-aperture and  **
**                                  z-aperture values into x,y, and z values.**
**      num_steps:              Defines the stepping mode that is used by   **
**                                  the driver cards.  (200 - full step;    **
**                                  400 - half step).  Half step mode is    **
**                                  used in this code revision.             **
**      difscrew:               Equivalent pitch value for differential     **
**                                  screw drive system.                     **
**      Zold1,Zold2,Zold3:      Intermediate variable used to represent     **
**                                  "old" leg lengths.                      **
**      LENG1,LENG2,LENG3:      Uncompensated leg lengths for legs 1,2,3.   **
**      LEGFIN1,'2,'3:          New leg lengths as a result of all calc's.  **
**      DLeg1,DLeg2,DLeg3:      Change in leg length values from previous   **
**                                  move to current move.                   **
**      Rev1,Rev2,Rev3:         Number of rotations each leg will undergo   **
**                                  to execute current move.                **
**      Step1,Step2,Step3:      Number of steps each motor will be pulsed   **
**                                  to execute current move.                **
**      Step1s,Step2s,Step3s:   Signs for rotating motors 1-3 (i.e. +or-)   **
**                                  + (1) CW rotation; - (0) CCS rotation.   **
**                                  Direction info written to PORT2.        **
**      Step1m,Step2m,Step3m:   Maximum step values for motors 1-3 for a    **
**                                  given calculation.                      **
**      MaxSteps:               Maximum number of steps for all 3 motors    **
**                                  for a given calculation.                **
```

```
**     Outvalue1,2:          Hex byte written to PORT1,2 for pulsing.     **
**                                                                        **
**                                                                        **
************************************************************************
**                                                                        **
**     SUMMARY OF PROGRAM EXECUTION:                                      **
**                                                                        **
**     This program is designed to drive 3 stepping motors from a "home"  **
**     position to a new position based on user-prescribed inputs.        **
**                                                                        **
**     The program converts variables for anode-to-puller gap, xy-aperture **
**     alignment, and z-aperture alignment into x,y,z values for input    **
**     into the main calculations.  Further, these x,y,z values are       **
**     converted into alpha,beta,z values which are used in two coordinate **
**     transformations.                                                   **
**                                                                        **
**     The bulk of the remaining calculations and function calls are      **
**     concerned with completing the new "leg length" calculations based on **
**     the user inputs.  Closed-form expressions for the lengths of the   **
**     three legs are used.  Refer to the thesis document for a complete  **
**     explanation and derivation.  "Leg length" refers to the gap between **
**     the moving upper plate and the stationary lower plate.  This gap, or **
**     "leg length" is changed using a differential screw drive system.   **
**     "Leg 1" refers to drive screw 1, "leg 2" to drive screw 2 and      **
**     "leg 3" to drive screw 3.                                          **
**                                                                        **
**     The key output of the mathematics is the step and direction infor- **
**     mation for each of the three motors.  These values are analyzed to **
**     determine the signs (+/-) and magnitudes of each of the three step **
**     motor moves.  These values are fed to the output portion of the    **
**     which pulses the three step motors in half-step mode through PORT1 **
**     and sets the direction bit for the three motors through PORT2.  Each **
**     write to output ports 1 and 2 uses a hex byte whose binary equiva- **
**     lent represents the actual bits which should be on (1) or off (0). **
**                                                                        **
**     The program gives the user several opportunities to re-input values **
**     or leave the program.  All three motors are pulsed to a "home"     **
**     position after querying user for pertinent move information.       **
**                                                                        **
************************************************************************
**                                                                        **
**     DELAY ROUTINE:                                                     **
**                                                                        **
**     In order to pulse the motors at the desired frequency (for speed and **
**     resonance considerations), a calibrated delay routine must be used. **
**     Several options are available.  A machine-INDEPENDENT routine is the **
**     most desireable solution since it would allow unlimited portability **
**     of the executable file among properly configured PCs.  A machine-  **
**     DEPENDENT routine is a less desireable, but effective solution     **
**     for a given routine.                                               **
**                                                                        **
**     During development a machine-INDEPENDENT routine was found via Ryle **
**     Design that has micro-second resolution.  A shareware version was  **
**     found to provide satisfactory timing resolution and accuracy. Code **
**     can be purchased for ~$100 (8/94).  This cost was deemed beyond the **
**     budget of this project.                                            **
**                                                                        **
**     Instead, a machine-DEPENDENT routine was created ("delay_manual")  **
**     and calibrated using an oscilloscope to provide a 1000 usec (1 msec) **
**     delay during writes to the motors.                                 **
**                                                                        **
**     Note the "delay_call" routine in the global definition section of  **
**     this code.  This block of code and the variables definition above it **
**     are building blocks for the MSCHRT (microsoft 'c' high-res timer)  **
**     code available through Ryle Design.  All the required calls and    **
**     variable definition are scattered throughout this code and are     **
**     indicated by the MSCHRT acronym in a comment preceding each section. **
**     Conversion of this code, which uses the machine-DEPENDENT delay,   **
**     to code which has machine-INDEPENDENT delay would be a matter of   **
**     simply removing the comment symbols around the MSCHRT calls and    **
**     deleting the two calls to "delay_manual" and the function "delay_  **
**     manual".                                                           **
```

```
 **                                                                          **
 ******************************************************************************/

#include <stdio.h>
#include <math.h>
#include <graph.h>
#include <conio.h>
#include <bios.h>
#include <time.h>
#include <pchrt.h>

#define BaseAdr 0x300            /*  Base address for PCIO12 card     */
#define PORT1 BaseAdr+0x0        /*  Address for PORT1                */
#define PORT2 BaseAdr+0x01       /*  Address for PORT2                */
#define IO_ctrl BaseAdr+0x03     /*  Mechanism for setting all ports  */
#define high 1
#define low 0
#define port_set 0x80
#define bit_1 0x001              /*  Hex values for writing to PORTs  */
#define bit_2 0x002
#define bit_3 0x004

/****  MSCHRT Delay Global Definitions ****
 ****  and Library Calls               ****
long unsigned   min_delay;
long unsigned   step_delay;
long unsigned   delay_request;
long unsigned   hits;
long unsigned   elapsed;
float           delay_error;
float           resolution;
tdelay_type     dp;

void delay_call(long unsigned how_long)
     {
       long unsigned   hits,elapsed;
       tdelay_type     dp;
       t_ask_delay(how_long,&dp);
       t_reset(1);

       if (delay_request<54000L)
          {
            t_entry(1);
            t_do_delay(&dp);
            t_exit(1);
          }
       else
          {
            t_entry(1);
            t_do_delay_wints(&dp);
            t_exit(1);
          }
     }
*/


main()
{

    /****  Initialize all variables and arrays  ****/

       int i, j, k, gonogo, ststop, num_steps;
       long int Step1s, Step2s, Step3s;
       long int Step1m, Step2m, Step3m, MaxSteps;
       long int start_step;
       long int count, delay_manual(), max(), max1;
       long int delay, home_delay, delay_sign;
       unsigned char Outvalue1;
       unsigned char Outvalue2;
       long unsigned how_long;
```

```
        double initgap, initxyap, initzap;
        double oldgap, oldxyap, oldzap;
        double gapval, xyapval, zapval;
        double x, y, z, x0, y0, z0, c1, c2, c3;
        double alpha, alphadeg, beta, betadeg;
        double Hy, Hx, PI;
        double difscrew,Zold1,Zold2,Zold3;
        double start_move;
        double correction_factor;

        double A1A2[4][4];
        double h, r, r1, r2, r3, rad32;

        double LENG1,LENG2,LENG3;
        double LEGFIN1,LEGFIN2,LEGFIN3;
        double Dleg1,Dleg2,Dleg3;
        double Rev1,Rev2,Rev3,Step1,Step2,Step3;

/****  Initialize the gap and aperture values (mm) at the
         anode-puller interface.  Values are taken off of nominal
         stack of values from CAD geometry  ****/
        initgap  =  0.10977 * 25.4;       /*  All Values in mm  */
        initxyap = -0.00882 * 25.4;
        initzap  =  0.0           ;
        x0       = -0.45271 * 25.4;
        y0       =  0.05712 * 25.4;
        z0       = 16.721   * 25.4;       /* Dist o-ring plane to c-1 central reg.*/
        c1       =  0.16690 * 25.4;
        c2       = -0.44390 * 25.4;
        c3       = 16.721   * 25.4;
        r        = 1.954    * 25.4;
        r1       = 2.1558   * 25.4;
        r2       = 2.1571   * 25.4;
        r3       = 2.1554   * 25.4;
        rad32    = (sqrt(3.0)/2.0);
        h        = -.15     * 25.4;       /* Initial gap is .150 in */
        gapval   =  0.0           ;
        xyapval  =  0.0           ;
        zapval   =  0.0           ;
        PI       = 4 * atan(1)    ;       /* Generate value for PI */
        difscrew = ((32.0-28.0)/(32.0*28.0))*25.4 ;  /* equiv. pitch dif screw */
                                                      /* in mm */
        num_steps= 400;                  /* Assumes 400 steps/rev */
                                        /* 1/2-step mode for motors */
        home_delay = 100;          /* Arbitrarily sets delay for homing routine */
        delay_sign = 10000;
        start_move = 0.1 * 25.4; /* Distance to move from home posn to start posn */
        start_step = start_move/difscrew*num_steps; /* #steps to get to start posn */
        correction_factor = 1.0 + (1.0/(7.3));

        Zold1  = -h;
        Zold2  = -h;
        Zold3  = -h;

        oldgap  = initgap;
        oldxyap = initxyap;
        oldzap  = initzap;

/****  Set I/O card as all ports OUTPUT only ****/
        outp(IO_ctrl, port_set);
        outp(PORT2, 0x07);       /* Moves the motors back one step */
        outp(PORT1, 0x00);       /* Open collector inverts logic */

    /****  Request new gap and aperture values from user       ****
            First clear screen so output looks good every time ****/

        start:
        _clearscreen(_GCLEARSCREEN);
        printf("                WELCOME TO THE ION SOURCE POSITIONING CODE\n\n");
        printf("                     READY TO MAKE DESIRED MOVES\n\n");
        printf("Please enter the desired delay in microseconds for pulsing the
motors.\n");
```

```
        printf("Delay should be between (20 - 100000).\n");
        scanf("%ld",&delay);

        printf("\nThe current anode-puller gap is %3.1lf mm.\n", oldgap);
        printf("Enter new value for gap (.2 -- 5.8 mm):  ");
        scanf("%lf",&gapval);

        printf("\nThe current x-y aperture alignment is %3.1lf mm.\n", oldxyap);
        printf("Enter new value for x-y aperture alignment (-2.7 -- 3.2 mm):  ");
        scanf("%lf",&xyapval);

        printf("\nThe current z aperture alignment is %3.1lf mm.\n", oldzap);
        printf("Enter new value for z aperture alignment (-2.0 -- 2.0 mm):  ");
        scanf("%lf",&zapval);

    /****  Calculate x, y, and z values from user inputs  ****/
        y = (c1 - gapval  - y0)  ;       /* values in mm */
        x = (xyapval + c2 - x0)  ;
        z = (zapval  + c3 - z0)  ;

/* printf("\n\nx = %lf\ty = %lf\tz = %lf\n",x,y,z); */

    /****  Check to see that input values are within acceptable ranges  ****/
        if((gapval  < .2)   || (gapval > 5.9)) {goto error;}
        if((xyapval < -2.8) || (xyapval > 3.3)){goto error;}
        if((zapval  < -2.1) || (zapval > 2.1)) {goto error;}
        if((delay == 0)) {goto error;}

    /****  Confirm inputs with user  ****/
        printf("\n\nYou have entered the following new values:\n");
        printf("\tRequested delay:\t%ld usec.\n",delay);
        printf("\tNew gap value:\t\t%3.1lf mm.\n",gapval);
        printf("\tNew xy-aperture value:\t%3.1lf mm.\n",xyapval);
        printf("\tNew z-aperture value:\t%3.1lf mm.\n",zapval);

    /****  Query user if they want to change entries; if so
          then loop back up to the input section.  ****/
        printf("\nAre these values acceptable?  (y or n)\n");
        scanf("%s", &gonogo);
        if ((toupper(gonogo)) == 'N')
                {
                printf("\t\t\t****  You have chosen No  ****");
                printf("\t\t\t\tPress Any Key to Continue\n\n");
                        while (!kbhit()) {}
                goto start;
                }

    /****  Calculate alpha and beta values from user inputs.  Recall that
            code will operate in radians.  Note definition of alphadeg and
            betadeg.  This is to put angles in degrees.  ****/

        alpha = atan((-1.0*y)/(z0*sqrt(1-((y*y)/(z0*z0)))));
        alphadeg = 180 / PI * alpha;

        beta = atan(x/(z0*sqrt(1-((x*x)/(z0*z0)))));
        betadeg = 180 / PI * beta;

    /****  Calculate Hx and Hy z-offsets  ****/
        Hx = x * tan(beta/2)  ;
        Hy = (-1.0 * y) * tan(alpha/2) ;

/* printf("\n\nalpha (rad) = %lf\tbeta (rad) = %lf\n",alpha,beta);
   printf("alpha (deg) = %lf\tbeta (deg) = %lf\n",alphadeg,betadeg);
   printf("\nHx = %lf\t\tHy = %lf\n",Hx,Hy);
   scanf("%s",&gonogo);  */

    /****  Generate matrix for coordinate transformation  ****/
    A1A2[1][1] = cos(beta);
    A1A2[1][2] = 0;
    A1A2[1][3] = sin(beta);
    A1A2[2][1] = sin(alpha) * sin(beta);
    A1A2[2][2] = cos(alpha);
```

```
    A1A2[2][3] = -(sin(alpha) * cos(beta));
    A1A2[3][1] = -(cos(alpha) * sin(beta));
    A1A2[3][2] = sin(alpha);
    A1A2[3][3] = cos(alpha) * cos(beta);

    /****  Calculate Leg Lengths  ****/

    LENG1 = (A1A2[3][1]*(r1/2) - A1A2[3][2]*rad32*r1)/A1A2[3][3] - h;
    LENG2 = (A1A2[3][1]*(r2/2) + A1A2[3][2]*rad32*r2)/A1A2[3][3] - h;
    LENG3 = (-A1A2[3][1]*r3) / A1A2[3][3] - h;

/* printf("\nLENG1 = %lf\n",LENG1);
   printf("\nLENG2 = %lf\n",LENG2);
   printf("\nLENG3 = %lf\n",LENG3);
   scanf("%s",&gonogo);  */


    /***************     FINAL LEG LENGTH CALCULATIONS    ***************
     ****  Calculate the final leg length by subtracting the calculated
           Hx, Hy, and z values from the values calculated above.  ****/

       LEGFIN1 = (LENG1 - Hx - Hy - z) ;
       LEGFIN2 = (LENG2 - Hx - Hy - z) ;
       LEGFIN3 = (LENG3 - Hx - Hy - z) ;

/* printf("\nLEGFIN1 = %lf\n",LEGFIN1);
   printf("\nLEGFIN2 = %lf\n",LEGFIN2);
   printf("\nLEGFIN3 = %lf\n",LEGFIN3);
   scanf("%s",&gonogo);  */

/****  Calculate the difference between the required leg length
          (new value) and the old leg length.  ****/

       Dleg1 = (Zold1 - LEGFIN1);
       Dleg2 = (Zold2 - LEGFIN2);
       Dleg3 = (Zold3 - LEGFIN3);

       Rev1 = Dleg1 / difscrew;          /*  Converts linear displ. into   */
       Rev2 = Dleg2 / difscrew;          /*  revolutions using difscrew    */
       Rev3 = Dleg3 / difscrew;          /*  value for screw equiv. pitch. */

       Step1 = Rev1 * num_steps;         /*   Assumes full step mode*/
       Step2 = Rev2 * num_steps;
       Step3 = Rev3 * num_steps;

    /****  Reset screen and print meaningful output  ***/

       _clearscreen(_GCLEARSCREEN);
       printf("                    RESULTS FROM THIS CALCULATION");
       printf("\n\n");
       printf("-- User-Specified Parameters --\n\n");
       printf("Old Gap:\t\t%4.3lf\tNew Gap:\t\t%4.3lf\n",oldgap,gapval);
       printf("Old XY-Aperture:\t%4.3lf\tNew XY-Aperture:\t%4.3lf\n",oldxyap,xyapval);
       printf("Old Z-Aperture:\t\t%4.3lf\tNew Z-Aperture:\t\t%4.3lf",oldzap,zapval);
       printf("\n\n");
       printf("LEG 1 Movement:\n");
       printf("\tDisplaced\t%5.4lf\tmm\n",Dleg1);
       printf("\tTurned\t\t%7.3lf\trevs\n",Rev1);
       printf("\tPulsed\t\t%4.0lf (%5.4lf)\tsteps\n",Step1,Step1);

       printf("LEG 2 Movement:\n");
       printf("\tDisplaced\t%5.4lf\tmm\n",Dleg2);
       printf("\tTurned\t\t%7.3lf\trevs\n",Rev2);
       printf("\tPulsed\t\t%4.0lf (%5.4lf)\tsteps\n",Step2,Step2);

       printf("LEG 3 Movement:\n");
       printf("\tDisplaced\t%5.4lf\tmm\n",Dleg3);
       printf("\tTurned\t\t%7.3lf\trevs\n",Rev3);
       printf("\tPulsed\t\t%4.0lf (%5.4lf)\tsteps\n",Step3,Step3);

    /****  EXECUTE MOVE and PULSE MOTORS  ****/
```

```
    printf("\n\nWould You Like to Execute this Move?  ");
    scanf("%s",&gonogo);
    if((toupper(gonogo)) == 'Y')
        {
            _clearscreen(_GCLEARSCREEN);
            printf("\t\t---  NOW PULSING THE MOTORS TO NEW POSITION  ---\n\n");
            printf("\t\tMotor1 Data: %5.1lf\tsteps\n",Step1);
            printf("\t\tMotor2 Data: %5.1lf\tsteps\n",Step2);
            printf("\t\tMotor3 Data: %5.1lf\tsteps\n",Step3);

/****  Determine magnitude of Step Values  ****/
            Step1m = abs(Step1);
            Step2m = abs(Step2);
            Step3m = abs(Step3);

/****  Determine signs of Step Values  ****/
            Step1s = (Step1<0.0) ? low : high;
            Step2s = (Step2<0.0) ? low : high;
            Step3s = (Step3<0.0) ? low : high;

            printf("\n\n\t\tMag  Step1: %d",Step1m);
            printf("\n\t\tSign Step1: %d",Step1s);
            printf("\n\t\tMag  Step2: %d",Step2m);
            printf("\n\t\tSign Step2: %d",Step2s);
            printf("\n\t\tMag  Step3: %d",Step3m);
            printf("\n\t\tSign Step3: %d",Step3s);

/****  Find Maximum of Step1, Step2, & Step3  ****/
            max1     = max(Step1m, Step2m);
            MaxSteps = max(max1, Step3m);
            printf("\n\nMax Steps: %d\n",MaxSteps);

/****  Output Direction Byte to Port2  ****
 ****  Bit 1 is sign for Step1,        ****
 ****  Bit 2 is sign for Step2,        ****
 ****  Bit 3 is sign for Step3         ****/

            Outvalue2 = 0x0;
            Outvalue2 = Step1s * bit_1 + Step2s * bit_2 + Step3s * bit_3;
            Outvalue2 = ~Outvalue2;

            outp(PORT2,Outvalue2);

            /* Use machine-DEPENDENT delay for step direction change*/
            count = delay_manual(delay_sign);

/****  Set Outvalue1 = 0x0  ****/
            Outvalue1 = 0x0;

/****  Set up Loop to Write a single byte to PORT1 which contains all
      the information to pulse all 3 motors simultaneously  ****/
            for (i=0; i<=MaxSteps; ++i)
              {
                 Outvalue1 = 0x0;
                /* Determine which bits still need to be pulsed */
                 if(i < Step1m) /* Select bit 1*/
                      {
                      Outvalue1 = Outvalue1 | bit_1;
                      }
                 if(i < Step2m) /* Select bit 2*/
                      {
                      Outvalue1 = Outvalue1 | bit_2;
                      }
                 if(i < Step3m) /* Select bit 3*/
                      {
                      Outvalue1 = Outvalue1 | bit_3;
                      }

                 /* Invert the Output for open collector driver */
                 Outvalue1 = ~Outvalue1;
                 outp(PORT1,Outvalue1);
```

```
                    /* Delay requested amount */
/*                     t_start();                                  For Use with
                         delay_call(delay_request);               MSCHRT delay
                         t_stop();                                software.*/
/*                        printf("\nDelay Value %ld\n",how_long);        */

                   /* Use machine-DEPENDENT delay */
                    count = delay_manual(delay);

                    /* Turn off selected bits */
                    Outvalue1 = 0x0;

                    /* Invert the Output for open collector driver */
                    Outvalue1 = ~Outvalue1;
                    outp(PORT1,Outvalue1);

                    /* Delay requested amount using MSCHRT routine*/
/*                     t_start();
                         delay_call(delay_request);
                         t_stop();  */

                   /* Delay using Machine-DEPENDENT delay */
                    count = delay_manual(delay);
                  }
    /****  Set current values to old values for next set of calc's ****/
       oldgap   = gapval;
       oldxyap  = xyapval;
       oldzap   = zapval;
       Zold1    = LEGFIN1;
       Zold2    = LEGFIN2;
       Zold3    = LEGFIN3;
           }

       printf("\nDo You Wish to Make Another Set of Calculations?  ");
       scanf("%s",&ststop);
       if((toupper(ststop)) == 'N') {goto end;}

       goto start;

      error:
       printf("\n\nYOU HAVE ENTERED AN OUT-OF-RANGE VALUE!!!");
       printf("\n\n\nDo You Wish to Enter New Values?  ");
       scanf("%s", &gonogo);
       if((toupper(gonogo)) == 'Y'){goto start;}

end:;

}

    /****  Function to find the maximum value between two inputs  ****/
long int max(long int val1, long int val2)
       {
       long int maxval;
         if((val1) > (val2)) {maxval = val1;}
         else {maxval = val2;}
       return maxval;
       }

    /****  Routine to accomplish delay for motor pulsing.  This is     ****
     ****  a machine-DEPENDENT delay, calibrated for an 8MHz AT-style  ****
     ****  PC.                                                         ****/
long int delay_manual(long int delay)
       {
       int i;
       long int loop;
       long int count;
           loop = delay * 1;     /*  Multiplier for manual tweaking  */
           count = 0;
           for(i=0 ; i<=loop; ++i)
               {Count = i;}
            return count;
       }
```

Ryle Design

Purveyors of Big Science Since 1987

PO Box 22, Mt. Pleasant MI 48804 USA

FAX TRANSMISSION

From:	**Thomas Leathley**	**Date:**	**08.23.94**
Voice:	**517.773.0587**	**Time:**	**1230 local**
Fax:	**517.773.0587**	**Pages:**	**04**

To: **Rick Paulus**
Company: CTI
Voice:
Fax: **615.966.8955**

Thank you for your interest in our precision timing tools. Enclosed are complete specifications for PC Timer Tools V5.0 and PC Timer Objects V1.0. Both will provide you with extensive timing capabilities in a PC/MSDOS environment. PC Timer Tools is for the "traditional" C and Turbo Pascal developer, while PC Timer Objects is for the object oriented developer using C++ or Turbo Pascal Objects. Each toolkit is $89.95 each, or $129.95 purchased together.

All our toolkits include complete source code, a royalty-free object code distribution license, and a 30 day "No Questions Asked" return policy. We accept VISA, MasterCard, and American Express.

Please contact us with any additional questions or comments. Best of luck with your present and future projects.

Sincerely yours,

Thomas Leathley
President

MSCHRT

Microsoft C High Resolution Timer Toolbox

Version 3.00

Shareware Evaluation Version

Ryle Design

P.O. Box 22

Mt. Pleasant, Michigan 48804

(517) 773-0587

Cserv 73047,1765

Theory

MSCHRT works by directly interfacing with channel 0 of the 8253 timer chip found in all compatible PCs. Channel 0 counts from 65535 to 0, when it triggers BIOS interrupt 8 and begins counting from 65535 again. Interrupt 8 is the hardware timer interrupt, and it increments the DOS time of day data word in the BIOS data area, checks for diskette motor activity and shuts down the motor if needed, and finally calls BIOS interrupt 1C (hex), which is known as the software timer tick. At boot time, 1C contains an IRET instruction, meaning there is no routine installed in that vector. Since the 8253 has a period of 838 nanoseconds, the hardware timer tick occurs every 54.925 milliseconds, or 18.2 times a second, which is the standard software time resolution of the PC. The 8253 count can be read by accessing the appropriate IO port, and by reading the 8253 count along with the low word DOS time of day count, we can construct a high resolution time stamp, which is the base unit of time for MSCHRT. The only modifications we make to the PC environment is to change the mode of channel 0 to give the output count a 50 percent duty cycle, which does not impact any other PC operations. Thus we have a simple, unobtrusive, highly accurate time reference with 838 nanosecond resolution. Once a highly accurate time reference is available, we can use it to calibrate a simple function that contains an inner and outer loop to generate precision delays.

The accuracy of MSCHRT is as accurate as the 8253 timer chip and the ability of the software to measure the overhead in retrieving the 8253 and DOS time of day data. With the advent of advanced chips in the 80286 and 80386 class, the effects of instruction prefetch queues, pipelining, and cache memory become significant and the same sequence of instructions may take different amounts of time to execute depending on the interaction of the processor's performance optimization mechanisms. The end result is twofold: MSCHRT's calibration routines may experience inaccuracies, and software under test may give different timing results on different runs. Normally the "jitter" introduced by these conditions is on the order of a few microseconds.

A major source of apparent erratic timer behavior comes from the interaction of asynchronous software and hardware interrupts that may occur during a timed interval. While the PC/MSDOS environment is essentially single-threaded, there is enough "background" activity occurring in the form of timer and keyboard interrupts that these interrupts may occur during an timed interval, and the time these interrupts take to complete are naturally counted in the timed interval. For very short duration timing a MSCHRT timer mode with interrupts disabled is available, but that may not be suitable for longer duration timings, or when concurrent timer activity is required. Be aware that few results in the PC environment will be exactly repeatable, and that some variance in the timing results is unavoidable.

The high resolution delay function of MSCHRT is particularly affected by background interrupts and processor performance optimization mechanisms. The initial delay calibration will yield delays with a variance of from one to three percent of the requested delay interval.

Additional functions are available to optimize the delay calibration to a specific delay interval, and using these functions can reduce the delay error to between zero and one percent, and thus they are recommended for critical applications.

MSCHRT Function Descriptions

The following is a brief synopsis of the functions found in MSCHRT. This chapter is meant to give the user a brief overview, and is not meant to be a definitive reference. It is essential that the user review the complete function reference in the Function Reference chapter before using any MSCHRT function. Some functions may not work as they would appear at first inspection, and in order to achieve high performance, MSCHRT has a minimum of internal error checking to prevent such catastrophes as NULL pointer references.

MSCHRT Timer Logic

Initialization functions

t_request	Requests the number of timers needed.
t_start	Initializes the MSCHRT timer logic.
t_stop	Shuts down MSCHRT and frees allocated memory.

Internal functions

t_get	Gets MSCHRT time stamp with interrupts enabled.
t_hires_entry	Gets MSCHRT time stamp with interrupts disabled. 8253 count is reset.
t_hires_exit	Gets MSCHRT time stamp and restores interrupts
t_diff	Calculates elapsed time difference between two MSCHRT time stamps.
t_calib	Calibrates MSCHRT timer logic, resets all timers, and calls the delay calibration function.
t_bios_entry	Receives timer start request from interrupt service routine.
t_bios_exit	Receives timer stop request from interrupt service routine.
t_bios_load_desc	Loads interrupt function description file
t_hook_int	Installs timer calls in specified interrupt vector.
t_unhook_int	Removes timer calls from specified interrupt vector.
t_delay_calib	Calibrates delay logic.

t_request is called by the user to specify the number of timers needed for generic timer operations if the user wishes to override the default number of timers provided (10). Since each timer data structure is allocated dynamically, the number of timers available is restricted by the amount of heap available and the constraints of the memory model you are using. If t_request is used in your program, it must be called before t_start.

PC Timer Tools Version 5.0

New version of the high resolution timer toolbox now available.
The definitive timing tool for profiling, data acquisition, and process control.

Toolkit Description

Anyone trying to time or schedule *any* type of event in a PC/MSDOS environment will find out very quickly that the 54.925 ms resolution of the DOS clock does not provide anything close to the resolution most developers need. Accelerating the PC's timer tick to gain higher resolution is a development project in and of itself, and any number of side effects from this scheme makes this technique generally unsuitable for code to be distributed into unknown execution environments. Fear not - **PC Timer Tools** from Ryle Design gives you a wide variety of functions for timing, delays, interrupt profiling, timer tick interrupt management, asynchronous thread scheduling, and stand alone execution profiling, with up to one microsecond resolution. With complete source code and a royalty-free distribution license, PC Timer Tools is the ideal timer toolbox for software developers working in a wide variety of environments.

Theory of Operation

The timing kernel achieves high resolution not by accelerating the timer tick interrupt (the common method) but rather by directly accessing the 8253/8254 timer controller present in all 100% compatible PC systems. By reading the timer controller's count and combining that with the four byte DOS time of day data words, we have a simple, unobtrusive, microsecond resolution timestamp. Delays can be generated by using the precision timing system to calibrate a set of loops, which are run as needed to achieve the delay requested.

Services Provided

A set of low level timing kernel services are available to be used by the other timing services, or by the software developer who needs a particular timing function not provided. Timers are available to time events in stopwatch fashion with microsecond, millisecond, or single second precision. The number of timers active is limited only by available heap. Report services provide formatted output of timer results. Delay functions generate user specified delays with microsecond, millisecond, or single second accuracy. Alarm services allow the user to specify a period of time to elapse and provide a means to check the alarm periodically for expiration, while performing other processing functions. BIOS interrupt profiling services provide a means to precisely profile the activities of user specified interrupts. A timer is dedicated to each of the 255 different values the AH register may have when an Interrupt is invoked, and complete formatted reports are generated showing interrupt function activity. Using these functions it is very easy to "eavesdrop" on MSDOS and BIOS activity. The timer tick management functions allow the user to run the timer tick interrupt at a rate other than 18.2 Hz, useful for a running a background ISR to service I/O ports. The thread scheduler allows up to 8 user written functions to execute at desired intervals with millisecond resolution. Finally, two utilities allow timing and interrupt profiling of executable programs. These are handy tools for interrupt "snooping" and general program execution timing.

Function List

The following is a complete list of the functions contained in PC Timer Tools V5, grouped by category. All functions are supported in both C and Turbo Pascal versions of the timing libraries. Full source code to all functions is included.

Timer Kernel

t_request	t_start
t_stop	t_get
t_diff	t_bigdiff
t_hires_entry	t_hires_exit
t_calib	

Timers

t_alloc	t_free
t_entry	t_exit
t_suspend	t_resume
t_reset	t_ask_timer
t_ask_elapsed	t_ask_count
t_check_timer	t_debug_on
t_debug_off	

Reports

t_cvt_time	t_cvt_bigtime
t_report	t_name
t_rname	t_fname
t_set_report	

BIOS Interrupt Profiling

t_bios_start	t_bios_stop
t_bios_suspend	t_bios_resume
t_bios_set_user	t_bios_hard_user
t_bios_set_file	t_bios_hard_file
t_bios_set_path	t_bios_ask
t_bios_reset	t_bios_report
t_bios_rname	t_bios_fname
t_bios_entry	t_bios_exit
t_bios_load_desc	

Delays

t_delay_calib	t_do_bigdelay
t_ask_delay	t_do_delay
t_do_delay_wints	t_calc_delay_ff
t_get_delay_ff	t_set_delay_ff
t_min_delay	t_res_delay

Alarms

t_alarm_request	t_alarm_start
t_alarm_alloc	t_alarm_free
t_alarm_set	t_alarm_check
t_alarm_stop	

Timer Tick Management

t_ttm_init	t_ttm_start
t_ttm_1C_set	t_ttm_stop

Thread Scheduler

t_sched_init	t_sched_set
t_sched_enable	t_sched_disable
t_sched_cancel	t_sched_reset
t_sched_delete	t_sched_synchronize
t_sched_off	

Executable Utilities

Exetime	Intpro

Compiler Support

PC Timer Tools version 5.0 supports the following compilers:

- Turbo C 2.0
- Turbo C++ 1.0, Second Edition 3.0
- Borland C++ 2.0, 3.0, 3.1, 4.0
- Turbo Pascal 5.0, 5.5, 6.0, 7.0
- Microsoft C 5.1, 6.0, 7.0
- Intel 386/486 C Code Builder 1.0, 1.1
- Zortech C++ 3.0

Complete library source code (in both C and Pascal) and a developer's distribution license is included.

Version 5 Update Summary

- selectable resolution (usec, msec, sec) for timers, alarms, and delays.
- timer report has new sort options
- timers and alarms may be allocated, used, and discarded dynamically
- improved delay performance on 486s
- timer error checking enhanced and debug mode added
- multiple asynchronous thread scheduler
- new support for Intel and Zortech C/C++ compilers
- updated BIOS interrupt profile descriptions

What About Windows?

Many of our current PC Timer Tools users are migrating to the Windows environment, and we have frequently been asked to provide a Windows compatible version of our toolkit. Until recently, a version compatible with Windows Enhanced mode has not been possible, due to the software virtualization of the 8253 timer hardware in Windows Enhanced mode. It appears that Windows 3.1 will provide the access to the timer hardware PC Timer Tools needs to work in Enhanced mode, and we have begun development of a high resolution timer toolbox suitable for that environment. We will advise current users of PC Timer Tools when a Windows version is available.

Need Some OOP Timer Tools?

If OOP is the *flavor du jour* in your current development efforts, you may want a full OOP implementation of PC Timer Tools, and **PC Timer Objects** is the answer to your needs. Please see page 5 for a complete product description.

PC Timer Tools V5 Now Shipping

If you have specific questions regarding your current development project and how it might use PC Timer Tools, contact us by voice, email, or fax and we'll be glad to respond to your application specific questions as they relate to our timer toolbox. We're here to help!

PC Timer Tools V5.0

$89.95 / $35 Update

PC Timer Objects Version 1.0

An OOP implementation of PC Timer Tools for C++ and Turbo Pascal Objects

Functional Overview

Software developers needing timing services that go beyond the 54 ms accuracy of the PC's DOS-BIOS timing functions have long turned to Ryle Design's **PC Timer Tools** for high resolution timing, precision delays, synchronous alarms, and multiple thread scheduling. **PC Timer Objects** brings these services to an OOP environment, with full support for Turbo C++, Borland C++, Microsoft C++ 7.0, Zortech C++ 3.0, and Turbo Pascal Objects. Although the functionality is similar, PC Timer Objects is not just an OOP wrapper placed around our PC Timer Tools code, but rather a complete rewrite of our timer kernel designed to take advantage of C++ and Turbo Pascal Object language features. With our C++ classes and Turbo Pascal objects, you can implement this library in a wide variety of timing, delay, and scheduling applications, or use our base classes and objects to develop very sophisticated functions that suit your unique timing requirements.

How It Works

Like PC Timer Tools, PC Timer Objects talks directly to channel 0 of the 8253/8254 timer controller found in all 100% compatible PCs. PC Timer Objects reads the 838 nanosecond 8253 timer "tick" count and combines this with the DOS time of day data words (which is a count of the number of times the 8253 timer count has overflowed) to generate a microsecond resolution time stamp with no adverse hardware or software side effects. This low-level timing engine is then used to time events and to calibrate the precision delay functions. The timing engine contains logic to calibrate itself to the PC's processor environment at run-time so that a single executable file using PC Timer Objects will provide accurate timing across the entire Intel 80x86 platform without machine specific tuning.

Services Available

PC Timer Objects provides the following timing services:

- precision timestamp services
- selectable resolution (usec, msec, sec) event timers, alarms, and delays
- multiple thread scheduler with 1 ms resolution
- event timer reporting functions generate reports on timer activity, ideal for profiling.

PC Timer Objects continues the Ryle Design tradition of **full source code** (in *both* C++ and Turbo Pascal) and a **royalty-free object code distribution license** included, so commercial developers may distribute our object code in bona fide end-user applications.

If precision timing, delays, alarms, or thread scheduling are required in an OOP implementation, PC Timer Objects is the high resolution choice. Call us with specific questions regarding your current or future projects and their timing requirements, and see how our precision timing functions can solve your problems today.

PC Timer Objects Class / Object Library Description

Timestamp Classes / Objects

The timestamp classes/objects provide a "snapshot" of the system timing hardware and low level functions to calculate time differences.

Classes / Objects:
usec_timestamp (usec resolution)
msec_timestamp (msec resolution)
sec_timestamp (sec resolution)

Member functions / object methods
get
operator - (C++)
diff (Turbo Pascal)

Timer Classes / Objects

"Stopwatch" style timers. Timers may be started, stopped, suspended, resumed, and each timer contains counters for both cumulative elapsed time and number of activations. Multiple timers may be active concurrently.

Classes/Objects
hrt_timer (base class / object)
usec_timer (usec resolution)
msec_timer (msec resolution)
sec_timer (sec resolution)

Member functions / object methods
count
current (virtual)
current_secs (virtual)
debug_off
debug_on
elapsed
elapsed_secs (virtual)
getname
last_elapsed (virtual)
last_elapsed_secs (virtual)
report
reset (virtual)
resolution (virtual)
resume (virtual)
setname
start (virtual)
state
stop (virtual)
suspend (virtual)
timestring (virtual)

Alarm Classes / Objects

Foreground synchronous alarms are initialized with a time value and then checked periodically for expiration. Alarms are ideal for handshake timeouts and animation synchronization.

Classes/Objects
hrt_alarm (base class / object)
usec_alarm (usec resolution)
msec_alarm (msec resolution)
sec_alarm (sec resolution)

Member functions / object methods
cancel
check(virtual)
reset(virtual)
set (virtual)

Delay Classes / Objects

Precision delays, completely self calibrating to the host PC runtime hardware environment.

Classes/Objects
hrt_delay (base class / object)
usec_delay (usec resolution)
msec_delay (msec resolution)
sec_delay (sec resolution)

Member functions / object methods
doit (virtual)
doit_critical (usec_delay only)
min_delay (usec_delay only)
optimize (usec_delay only)
optimize_critical (usec_delay only)
request (virtual)

Thread Class / Object

Schedules functions to execute in the "background" while "foreground" processing continues. Not a substitute for a true multitasking operating system, but an effective means to implement scheduled, asynchronous servicing of I/O cards in communications, data acquisition, and process control applications. The scheduling resolution is 1 millisecond.

Classes/Objects
hrt_thread

Member functions / object methods
reset
resume
scheduler_on
scheduler_off
set
suspend
synchronize_threads

Compiler Support

Turbo C++ 1.0, 2nd Edition, 3.0
Borland C++ 2.0, 3.0, 3.1, 4.0
Microsoft C++ 7.0
Zortech C++ 3.0 (real & 286 protected)
Turbo Pascal 5.5, 6.0, 7.0
Borland Pascal 7.0

Toolkit Includes

Full source code (in both C++ and Turbo Pascal), royalty-free object code distribution license, our 30 day "No Questions Asked" return policy, and a complete reference manual with example programs ready to compile and run.

PC Timer Objects Version 1.0

$89.95

General Information and Official Fine Print

Our 30 Day "No Questions Asked" Return Policy

Your satisfaction is always our primary objective. All of our products may be returned within 30 days of purchase for *any reason* as long as the product is in good condition and you have destroyed any backup copies you may have made. You will receive a full refund (no sneaky restocking fees here) less your original shipping charges. If returning a product, please call first for a return authorization number.

Technical Support

Technical support for our products is available by any of the following methods:

1. Telephone to 517-773-0587 during normal business hours Monday-Friday Eastern Standard Time (-5 hours GMT).
2. Fax to 517-773-0587 24 hours a day, 7 days a week.
3. Compuserve email to 73047,1765. We check our CIS email daily.
4. Technical support BBS (517-772-2399) 12/24/9600, 8N1.

Payment Terms For USA Customers

We accept VISA, MasterCard, American Express, and checks. Please do not send cash. COD and net terms (purchase orders) are available to qualified businesses and institutions - please inquire.

Payment Terms For Customers Outside the USA

All orders from outside the USA must be prepaid by VISA, MasterCard, American Express, or a check drawn in US dollars payable through a US bank branch. Please do not send cash. If paying by check, the check must be MICR encoded - these are the machine readable routing numbers that go across the bottom of the check. As our bank charges us $50 to cash a check that is not MICR encoded, we cannot accept checks without MICR encoding. If in doubt, ask your bank when you have the check prepared.

How To Order

You may order by telephone or fax to 517-773-0587, by mail to PO Box 22, Mt. Pleasant, MI 48804 USA, or by Compuserve email to 73047,1765. Orders are shipped within 24 hours of receipt.

Our Customers Include ...

American Airlines, Atomic Energy Canada, Conoco, Digital Equipment, Du Pont, Dun & Bradstreet, E & J Gallo, EDS, Georgia Tech, Hughes Aircraft, International Paper, Intuit, Jet Propulsion Laboratory, Max Planck Institute, Matsushita, Microsoft, NASA, Northrup, Phoenix Technologies, Sandia Labs, Sony, Texas Instruments, USGS, US Navy, Zenith, and hundreds of other companies and universities worldwide. Our total installed userbase numbers in the thousands.

Ryle Design Product Order Form - Pricing Effective May 1, 1993

Mail to Ryle Design, PO Box 22, Mt. Pleasant, MI 48804 USA or fax to 517-773-0587
Telephone orders with VISA or MasterCard may call 517-773-0587 10am to 4pm EDT

Name_____

Company _____

Address _____

_____

_____

Phone _____ Fax _____ Email_____

BGI Printer Driver Toolkit Version 1.2 **($129.95 each, $40 update, serial #_____)** ... $_____

BGI For Windows Version 1.1 **($129.95 each, V1.0 users update *free*)** ... $_____

Save! BGI Printer Driver Toolkit 1.2 & BGI For Windows 1.0 **(both for $189.95)** ... $_____

PC Timer Tools Version 5.0 **($89.95 each, $35 update - serial #_____)** ... $_____

PC Timer Objects Version 1.0 **($89.95 each)** ... $_____

Save! PC Timer Tools V5.0 and PC Timer Objects V1.0 **(both for $129.95)** ... $_____

Shipping **(USA Ground $5, USA 2 Day Air $7, USA Next Day Air $12, Overseas Air $10)** ... $_____

Subtotal ... $_____

Michigan residents add 4% sales tax ... $_____

Total enclosed or charged ... $_____

____ Check or money order enclosed payable to RYLE DESIGN

____ Charge my Visa/MasterCard/American Express #_____

Card expiration date _____ Signature _____

APPENDIX D

MOTOR & MOTOR DRIVER CARD

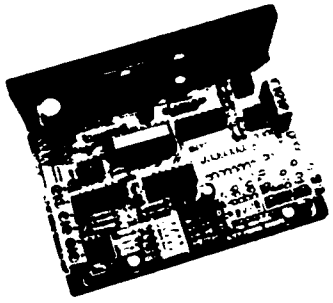
TECHNICAL SPECIFICATIONS

Applied Motion Products

404 Westridge Dr. • Watsonville, CA 95076
(408) 761-6555 • (800) 525-1609
FAX (408) 761-6544

Step Motor Driver

Full & Half Step
2A, 35V Bipolar Chopper



Features

- 12–35 VDC motor supply (including ripple)
- single power supply
- 125 mA – 2 amps/phase motor current
- DIP switch selectable current from 16 levels
- jumper selectable full and half step
- thermal protection
- optional heat sink
- screw terminal connectors
- inaudible PWM amplifiers
- optoisolated inputs
- automatic idle current reduction
- drives 4, 6 or 8 lead step motors, sizes 14–23

Description

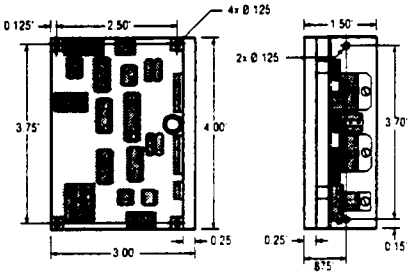
The 2035 step motor driver contains a full and half step phase sequencer, two switching amplifiers and optoisolation circuits. This driver also includes an automatic feature to lower motor current anytime the motor is left at rest for more than one second.

The amplifiers regulate motor current by chopping at a constant, inaudible frequency. Phase current is selected from 16 levels by a DIP switch. Full or half step operation is also selected by the DIP switch.

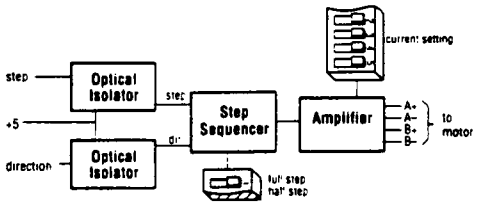
The 2035 is a PC board mounted on an aluminum angle heat transfer chassis. Mating heat sink available for stand alone operation.

Options

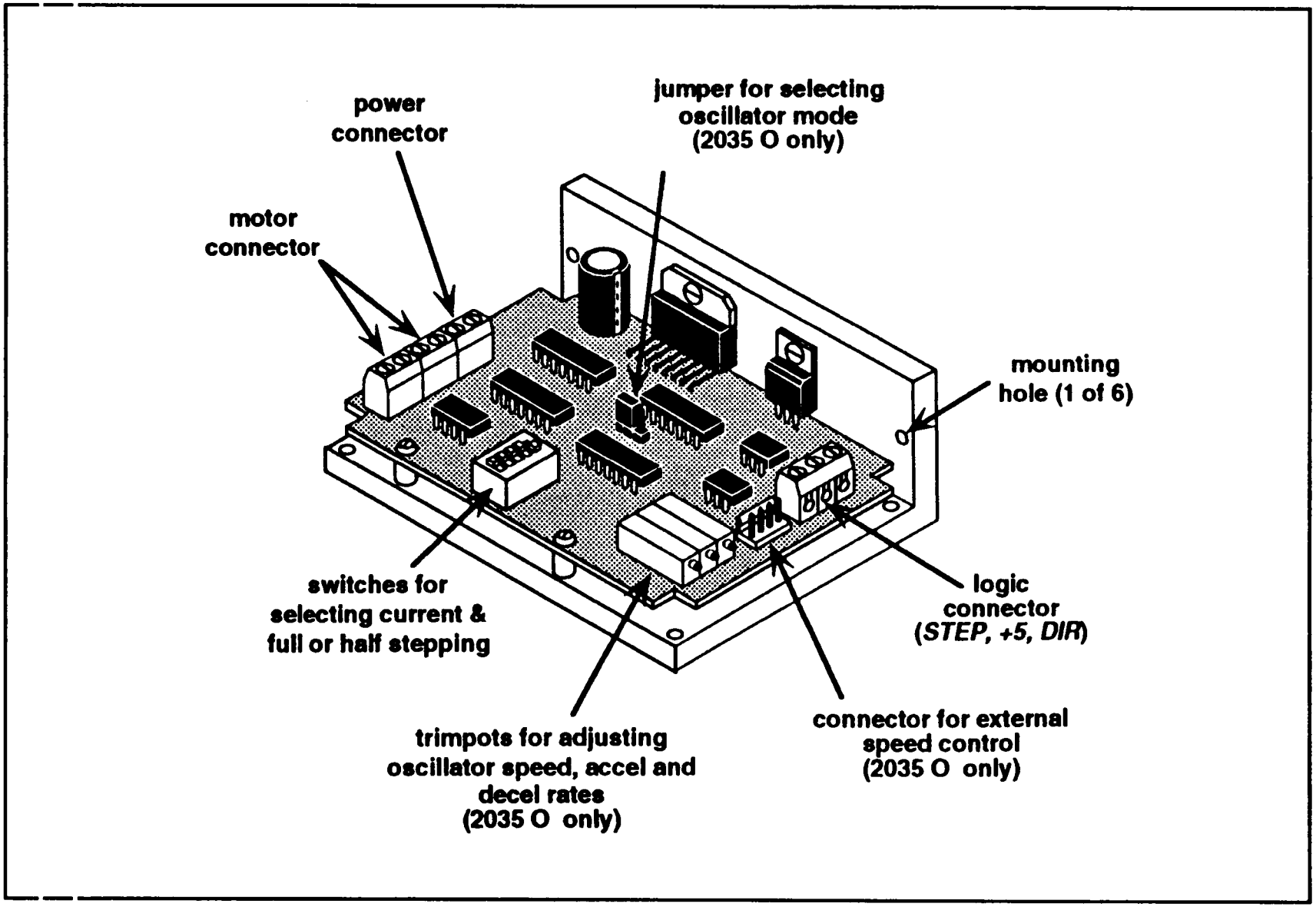
- on board oscillator (Model 2035 O)
- add on heat sink (Model 2035 H)



MECHANICAL OUTLINE



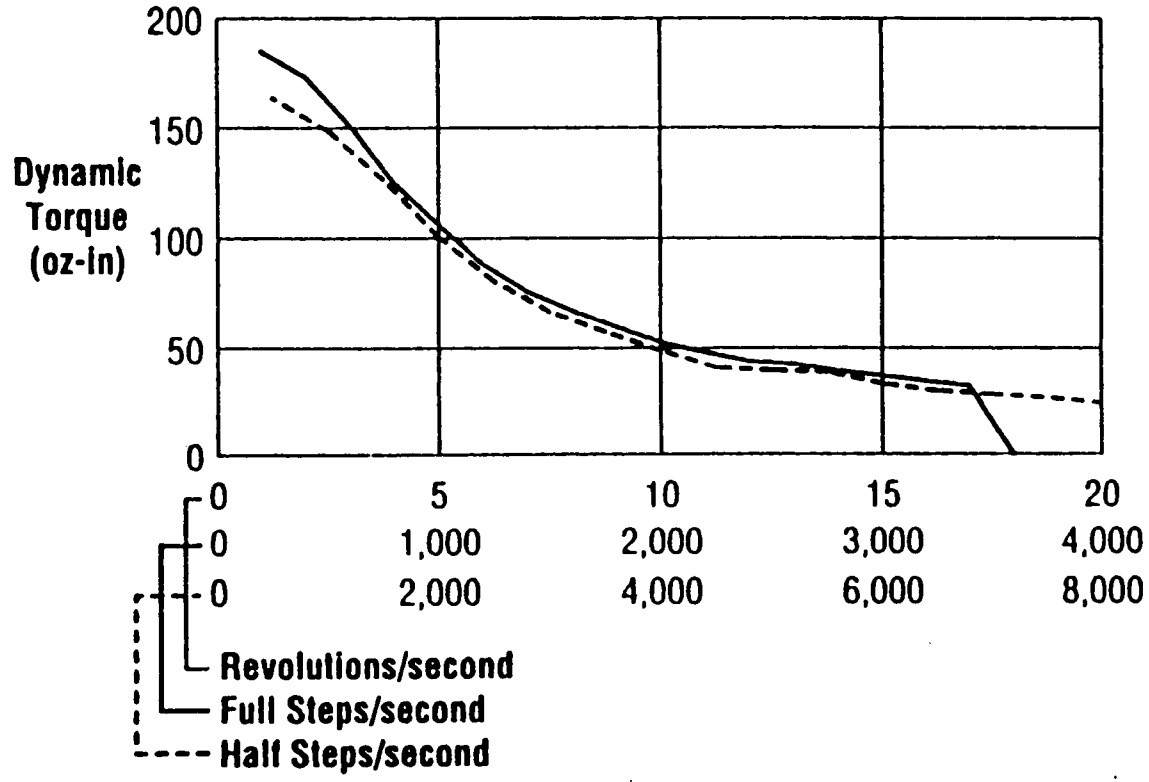
BLOCK DIAGRAM


power connector
jumper for selecting oscillator mode (2035 O only)
motor connector
mounting hole (1 of 6)
switches for selecting current & full or half stepping
logic connector (STEP, +5, DIR)
trimpots for adjusting oscillator speed, accel and decel rates (2035 O only)
connector for external speed control (2035 O only)

4034-322 MOTOR

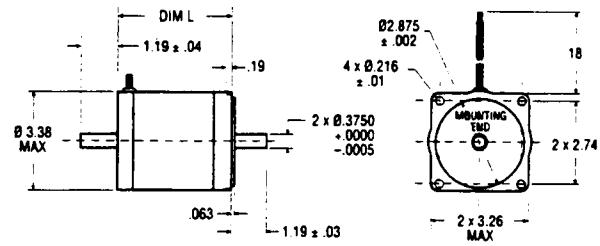
Motor Connection: Center Tap to End

Drive Setting: 35 VDC • 1.6A/Phase



Applied Motion Products

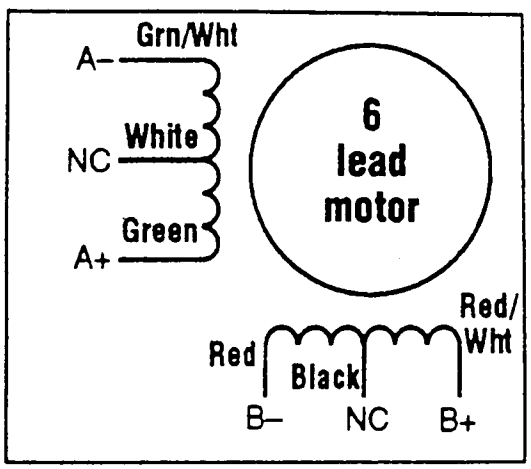
404 Westridge Dr. • Watsonville, CA 95076
(408) 761-6555 • (800) 525-1609
FAX (408) 761-6544



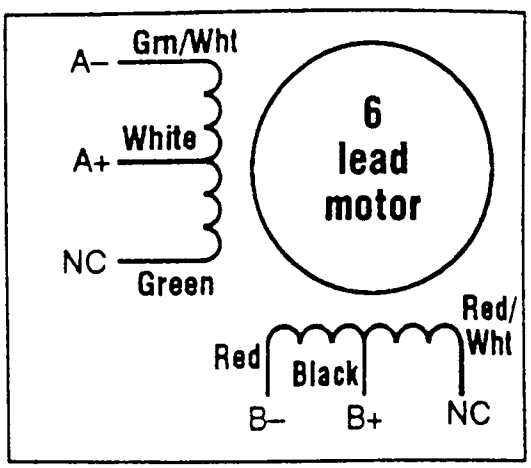
Part #	Motor Length (inches)	Holding Torque (oz-in)	Leads	Step Angle	Volts	Amps	Ohms	mH	Rotor Inertia (oz-in²/G-CM²)	Motor Weight (Lbs.)
→ 4034-322	2.5	150	6	1.8	5.8	1.6	3.6	11.5	3.66/670	3.00
4034-324	2.5	150	6	1.8	2.5	3.1	.82	3.0	3.66/670	3.00
4034-326	2.5	150	6	1.8	1.8	4.7	.38	1.25	3.66/670	3.00
5034-348	2.5	150	8	1.8	2.1	3.42	.64	2.5	3.66/670	3.00
4034-329	3.7	300	6	1.8	3.0	4.0	.75	3.0	6.72/1230	5.40
4034-331	3.7	300	6	1.8	2.5	4.6	.55	2.76	6.72/1230	5.40
5034-349	3.7	316	8	1.8	2.5	5.00	.50	2.9	6.72/1230	5.40
4034-334	5.1	450	6	1.8	4.3	3.5	1.22	7.0	10.2/1870	7.70
4034-336	5.1	450	6	1.8	2.6	5.5	.48	2.5	10.2/1870	7.70
4034-338	5.1	450	6	1.8	2.5	7.0	.35	1.7	10.2/1870	7.70
4034-339	5.1	450	6	1.8	1.5	11	.14	.7	10.2/1870	7.70
5034-350	5.1	472	8	1.8	2.5	5.90	.42	2.6	10.2/1870	7.70

OTHER LENGTHS AND WINDINGS AVAILABLE UPON REQUEST

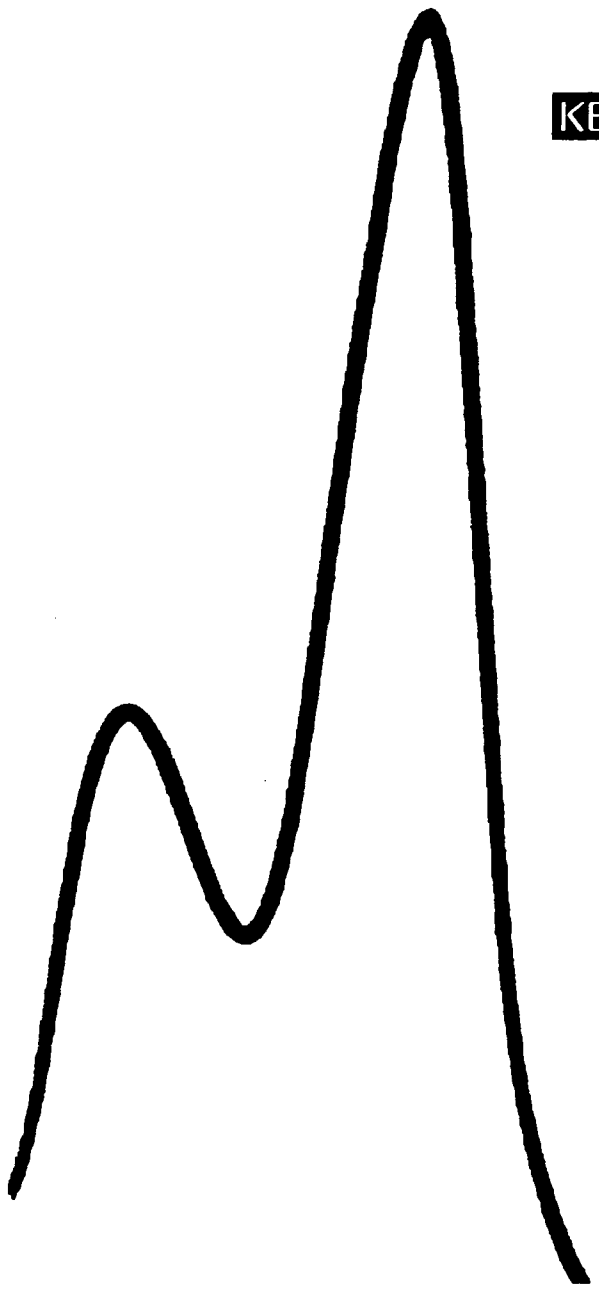
*Part numbers listed are for single shaft. To order double shaft add 'D' to the end.



6 Leads Series Connected



6 Leads Center Tap Connected



KEITHLEY METRABYTE

PIO-12

USER'S GUIDE

Table A-2. Environmental Specifications

Feature	Specification
Bus-loading power consumption	170 mA
Operating temperature range	0 to 50° C
Stoarge temperature range	-40 to 100° C
Humidity	0 to 90% noncondensing
Dimensions	5 inches L x 4.25 inches H x 0.75 inches (12.7 cm x 10.8 cm x 1.9 cm)
Weight	4 oz. (113 g)

Table 2-1. Values of Base Address Switch Positions

Switch Position	Address Line	Value When Switch Is OFF	
		Decimal	Hexadecimal
1	A9	512	200
2	A8	256	100
3	A7	128	80
4	A6	64	40
5	A5	32	20
6	A4	16	10
7	A3	8	8
8	A2	4	4

A

Specifications

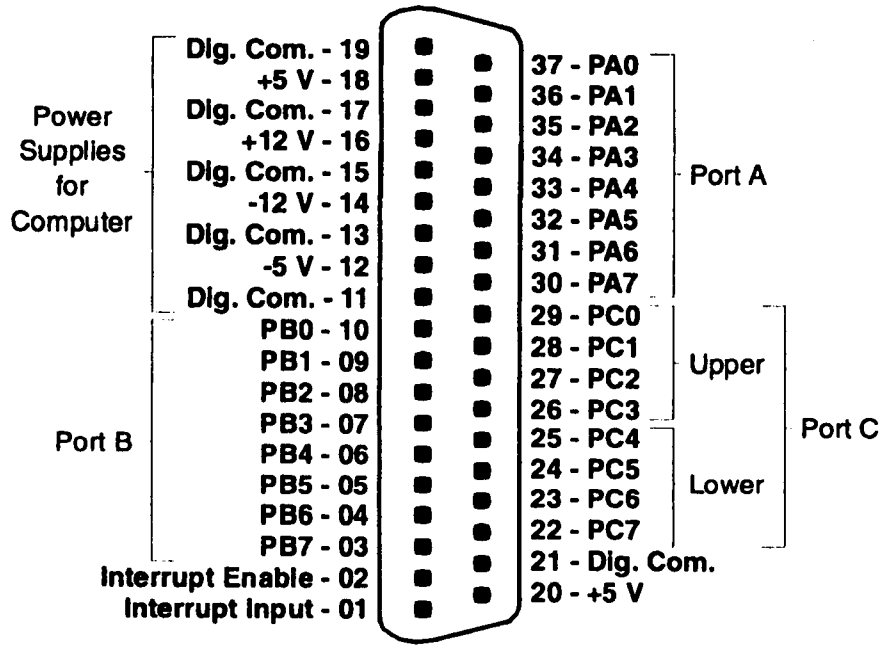
Inputs and outputs of the PIO-12 are TTL/DTL-compatible. Outputs can drive one standard TTL (74 series) load or four LSTTL (74LS) loads. CMOS compatibility can be achieved by connecting a 10 kΩ pullup resistor from the PI)-12 output to +5 V. Table A-1 and Table A-2 list PIO-12 specifications.

Table A-1. Input and Output Specifications

Logic Inputs and Outputs	Minimum	Maximum
Input logic low voltage	-0.5 V	0.8 V
Input logic high voltage	2.0 V	5.0 V
Input load current for the PA, PB, and PC ports ($0 < V_{in} < 5$ V)	-10 µA	10 µA
Input low current interrupt inputs	—	-0.4 mA
Input high current interrupt inputs	—	20 µA
Output low voltage for PA, PB, and PC ports (I_{sink} = 1.7 mA)	—	-0.45 V
Output high voltage for PA, PB, and PC ports (I_{source} = 200 µA)	2.4 V	—

Cabling and Wiring

This chapter describes the cabling and wiring required for attaching accessories and I/O lines to the main I/O connector of your PIO-12 boards. The main I/O connector of the PIO-12 is a 37-pin D-type. Pin assignments for this connector are shown in Figure 3-1.



APPENDIX E

OPEN COLLECTOR CALCULATIONS

Open Collector design requirement

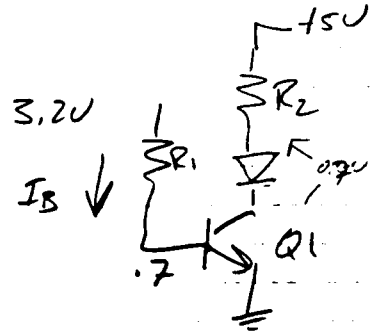
- $I_C \geq 5mA$ when $V_{in} = 3.2V$ (TTL min).
- Calculate I_C for Q1 in active region with $V_{in} = 3.2V$
- $I_C = I_B \cdot \beta$; $\beta = 50$

(1) $I_{C,active} = \frac{V_{in} - V_{BE}}{R_1}$ must be $> 5mA$

(2) $I_{C_{active}} \cdot R_2 > \frac{5V - V_d - V_{CE,SAT}}{R_2}$

If 2 is true, Q1 is in saturation and $V_{CE} \approx 0.4V$ and Required current drive is available

TTL High



$$I_B = \frac{3.2 - 0.7}{10K} = \frac{2.5V}{10K} = 0.25\ mA$$

Q1 $\beta \leq 50 = \frac{I_C}{I_B}$

$I_C = 12.5\ mA$ if Q1 is in active region
($12.5\ mA >$ Required $5\ mA$ — SPEC is MET)

However $I_{C,max} = \frac{V_{CC} - V_d - V_{CE,SAT,Q1}}{R_2}$

$$= \frac{5V - 0.7 - 0.2}{10K} = \frac{4.1V}{10K}$$

$$= 0.41\ mA$$

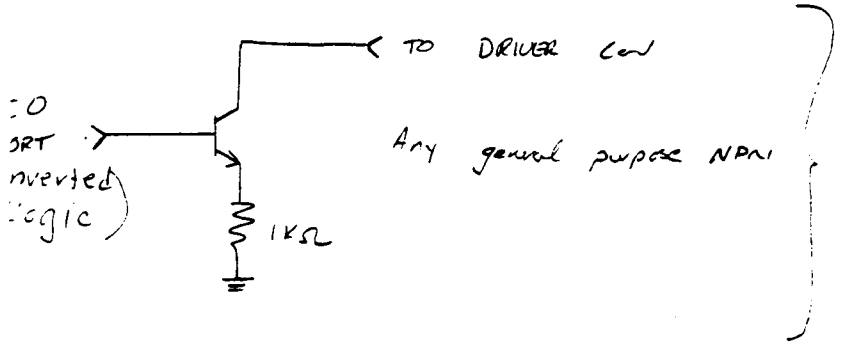
So far

$0.41\ mA < 12.5\ mA$

So Q1 is saturated

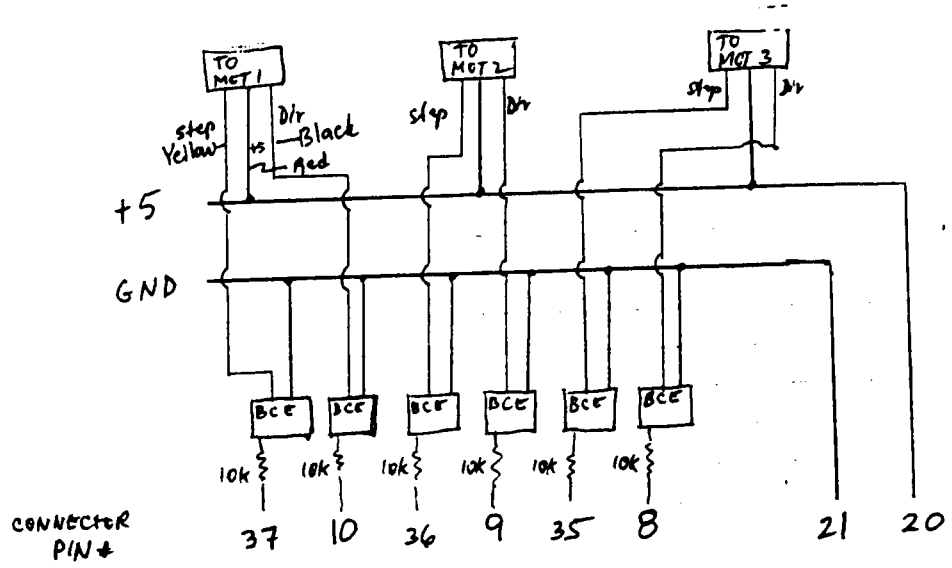
$V_{OL} \leq 0.4V$ and open collector driver is acceptable.

Need to invert output signal from PCIO-12 because of inversion inherent in ~~a~~ ~~the~~ Add-on output circuit is required ~~by~~ ~~to propriate~~ ~~s~~ because PIO12 is unable to ~~propriate~~ ~~s~~ ~~the~~ ~~input~~ meet the current sinking rights of the driver card.

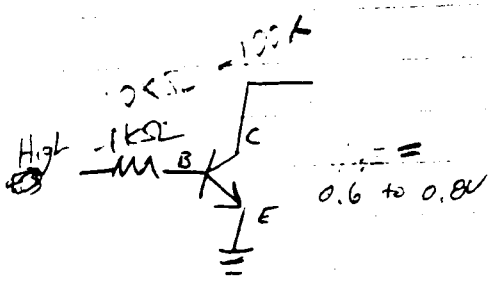


Home Check —
~~457390~~ 27AHLM385
~~B68B~~ Z-1.2
} Zener diode

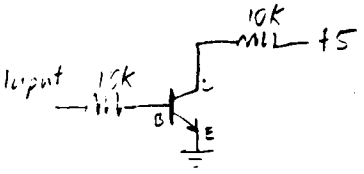
Use ~ (tilde) operator to invert the ~~signa~~ logic. X = ~X



BCE is ~~NAC~~ misloading! Package reads B E C !



No. _



Channel	Input	Initial	Verify Input	Output (C, before ICK)
1	High	0	4	0
	Low		0	4
2	High	0	4	0
	Low		0	4
3	High	0	4	0
	Low		0	4
4	High	0	4	0
	Low		0	4
5	High	0	4	0
	Low		0	4
6	High	0	4	0
	Low		0	4

Proto board performs INVERSION!

power-up:
Verified +5 across ground and +5 bus bar
Measured following delay values

Delay Value	Period	Freq
1	199 µs	5.03 KHz
5	315 µs	3.18 kHz
10	467 µs	2.14 kHz
20	761 µs	1.31 KHz
50	1643 µs	609 Hz
100	3124 µs	320 Hz
200	6.05 ms	165 Hz
500	15.13 ms	66.1 Hz
1000	29.96 ms	33.4 Hz
5000	138.45 ms	7.2 Hz
10000	292.8 ms	3.4 Hz
15000	432.2 ms	2.3 Hz
20000	585.5 ms	1.7 Hz

Using Tektronix 2445A 150 MHz
Probe-TEK P6133, 10 MΩ, 12.5pF, 10x
Tied ground to PC frame

—	25	912.5 µs	1.10 KHz
	30	1057 µs	946 Hz
—	35	12085 µs	827 Hz
	40	1365 µs	733 Hz
	45	1494 µs	669 Hz

** SEE SUMMARY CHART NEXT PAGE

Need to invert Outport Calls!

Equipment :

1) 16N_8C.EXE code running on 8 MHz, AT-style PC. Out connected to Junction box where open-collector output board and 3 driver cards are mounted.
→ Checking counts on the "STEP" input to each d

2) EG&G ORTEC Model 871 Timer and Counter plugged Standard NIM power supply chassis.
→ Counter input is on "POSITIVE"

Procedure :

Setting Delay = 20 in 16N_8C.EXE each time and ran identical trials for meters 1, 2, and 3.
started executing program once each time (from DOS prompt)

		Meter 1	Meter 2	Meter 3	
Theoretical		6772	8922	5730	Gap = 5.8,
Actual	1	6770	8921	6072	ZAP = 2
	2	6771	8922	5729	
	3	6771	8922	5845	
	4	6771	8922	5730	
	5	6771	8922	5730	

indicates direction

		(-) 6588	(-) 8436	(-) 5981	Gap = .2, XYMAP
Theoretical					ZAP = -
Actual	1	6588	8436	5981	Delay = 1
	2	6588	8436	5981	
	3	6588	8436	5981	
	4	6588	8436	5981	
	5	6588	8436	5981	

		6772 + 13360	8922 + 17359	5730 + 11711	2-stage Calc.
Theoretical		20132	26281	17441	
Actual	1	20131	26280	17441	
	2	20131	26280	17441	
	3	20131	26280	17441	

Gap 5.
XYr 3.2
ZA. 2,

Page No. _

- Added pre
- Pulse @ 20

		Motor 1	Motor 2	Motor 3	
1 Rev (z = -.1136)	Go return	centered "	centered "	slightly before " "	← Adjusted after 1st
1 Rev	Go return	centered " cent.	centered " cent.	centered " cent.	- Pulsed @ 50
2 Rev (z = -.2272)		cent. "	cent. "	cent. "	
3 Rev z = -.3408		slightly past cent.	cent "	cent. "	
4 Rev z = -.4544		slightly past ~cent.	cent. cent	slightly past ~cent.	
5 Rev z = -.568		sl. past - cent. cent. cent	cent — cent cent.	sl past - cent. Cent cent.	
6 Rev z = -.6816		sl. past cent.	sl. past cent.	sl. past cent.	
7 Rev z = -.7952		sl. past ~cent.	sl. past cent.	sl. past cent.	
8 Rev z = -.9088		sl. past ~cent.	sl. past og cent.	sl. past cent.	
9 Rev z = -1.0224		sl. past ~cent	sl. past og cent	sl. past cent	
10 Rev z = -1.136		sl. past cent. sl. past	sl past og cent. sl. past og	sl. past cent. sl. past	
(15.027) 15 Rev z = -1.704		sl. past og cent. sl. past og	sl. past og cent. sl. past og	sl. past og cent. sl. past og	

To Pa

APPENDIX F

CONTACT STRESS SPREADSHEET ANALYSIS

CONTACT STRESS ANALYSIS

From the development in Chapter Five, several equations are used in the contact stress analysis. First, consider a quantity Δ which is a function of Young's Modulus (E) and Poisson's ratio (ν) for the two mating bodies:

$$\Delta = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \qquad (5\text{-}3)$$

Using this expression for Δ, the expressions for contact pressure (p_o) and the radius of the area of contact (a_o) may be stated as follows:

$$p_o = 0.578 \cdot \sqrt[3]{\frac{F(1/R_1 + 1/R_2)}{\Delta^2}} \qquad (5\text{-}4)$$

and

$$a_o = 0.908 \cdot \sqrt[3]{\frac{F\Delta}{1/R_1 + 1/R_2}} \qquad (5\text{-}5)$$

where:
F is the loading force;
R_1 is the radius of the sphere;
R_2 is the radius of the mating part (flat plate in this case so $R_2 = \infty$ in this case);

For the mechanism under investigation, the following values[5] were used to calculate Δ, p_o and a_o:

$F = 57/3 = 19$ lb	$E_1 = 15 \cdot 10^6$ psi	$\nu_1 = 0.34$
$R_1 = .125$ in	$E_2 = 28 \cdot 10^6$ psi	$\nu_2 = 0.30$

Spreadsheet outputs showing the effects of different R values with different load values are shown on the following two pages.

Ebrass	1.50E+07										
Esst	2.80E+07										
mu(brass)	0.34										
mu(sst)	0.3										
Delta	9.146E-08										
				JUVINALL FORMULAS							
				CONTACT STRESS VALUES (psi)							
	Load (lb)										
R (in)	20	19	18	17	16	15	14	13	12	11	10
0.063	490754	482434	473818	464876	455575	445879	435742	425110	413918	402085	389512
0.125	309156	303915	298486	292853	286995	280886	274500	267803	260752	253298	245377
0.188	235930	231930	227788	223489	219018	214356	209483	204372	198991	193302	187258
0.250	194756	191454	188035	184486	180795	176947	172924	168705	164263	159568	154578
0.313	167835	164990	162043	158985	155805	152489	149022	145386	141558	137511	133211
0.375	148626	146107	143497	140789	137973	135036	131966	128746	125356	121773	117965
0.438	134111	131838	129483	127039	124498	121848	119078	116172	113114	109880	106444
0.500	122688	120609	118454	116219	113894	111470	108936	106278	103479	100521	97378
				CONTACT AREA VALUES (in^2)							
	Load (lb)										
R (in)	20	19	18	17	16	15	14	13	12	11	10
0.063	0.0044	0.0043	0.0043	0.0042	0.0041	0.0040	0.0039	0.0038	0.0037	0.0036	0.0035
0.125	0.0056	0.0055	0.0054	0.0053	0.0052	0.0050	0.0049	0.0048	0.0047	0.0045	0.0044
0.188	0.0064	0.0062	0.0061	0.0060	0.0059	0.0058	0.0056	0.0055	0.0054	0.0052	0.0050
0.250	0.0070	0.0069	0.0068	0.0066	0.0065	0.0064	0.0062	0.0061	0.0059	0.0057	0.0056
0.313	0.0075	0.0074	0.0073	0.0071	0.0070	0.0068	0.0067	0.0065	0.0064	0.0062	0.0060
0.375	0.0080	0.0079	0.0077	0.0076	0.0074	0.0073	0.0071	0.0069	0.0068	0.0066	0.0064
0.438	0.0084	0.0083	0.0081	0.0080	0.0078	0.0077	0.0075	0.0073	0.0071	0.0069	0.0067
0.500	0.0088	0.0087	0.0085	0.0083	0.0082	0.0080	0.0078	0.0076	0.0074	0.0072	0.0070

Ebrass	1.50E+07										
Esst	2.80E+07										
mu(brass)	0.34										
mu(sst)	0.3										
Delta	9.146E-08										
			SPOTTS FORMULAS								
			CONTACT STRESS VALUES (psi)								
	Load (lb)										
R (in)	20	19	18	17	16	15	14	13	12	11	10
0.063	485130	476906	468388	459548	450355	440770	430749	420238	409174	397477	385048
0.125	305613	300432	295066	289497	283706	277667	271355	264734	257764	250395	242565
0.188	233226	229272	225177	220928	216508	211900	207082	202030	196710	191087	185112
0.250	192524	189260	185880	182372	178723	174920	170943	166772	162381	157739	152806
0.313	165912	163099	160186	157163	154019	150741	147314	143720	139936	135935	131684
0.375	146923	144433	141853	139176	136391	133489	130454	127271	123920	120377	116613
0.438	132574	130327	127999	125583	123071	120452	117713	114841	111817	108621	105224
0.500	121282	119226	117097	114887	112589	110192	107687	105060	102294	99369	96262

VITA

Richard Semmes Paulus was born in Maryville, Tennessee on March 3, 1966, to Thomas J. and Susan S. Paulus. He is the youngest of three children, all of whom are engineers. He received his primary and secondary education in the Knox County school system in East Tennessee, graduating from Bearden High School in 1984. The following September Mr. Paulus entered The University of Tennessee, Knoxville in the college of Mechanical Engineering and in December, 1988 received the degree of Bachelor of Science in Mechanical Engineering.

Following his undergraduate work Mr. Paulus worked for two years with the Hewlett-Packard Company in Colorado Springs, Colorado as a Manufacturing Development Engineer focusing on design margin improvement of oscilloscopes. In May 1991, he continued his professional career with Scientific-Atlanta, Incorporated in Atlanta, Georgia as a Senior Manufacturing Engineer supporting machine shop and mechanical assembly of large satellite tracking products.

In August, 1993 Mr. Paulus reentered The University of Tennessee, Knoxville in the college of Mechanical Engineering and in December, 1994 received the degree of Master of Science in Mechanical Engineering. His thesis work entitled, "Development and Evaluation of an Improved Positioning Scheme for a Cyclotron Ion Source" was conducted over a six-month period at CTI, Incorporated in Knoxville, Tennessee.

Mr. Paulus is married to the former Audrey van Blommesteyn. They have a daughter, Caroline. They enjoy outdoor activities including camping, hiking, and biking.