# CUSTOM DEVELOPED COMPUTER NUMERICAL CONTROL INTERFACE

N. Jackimec

**MECH 4160 Graduation Thesis**

**Advisor:**

Dr. S. Balakrishnan

**Department of Mechanical and Manufacturing Engineering**

**University of Manitoba**

April 2010

II

# Abstract

Many CNC (Computer Numerical Control) milling machines are commercially available today, but none offers an open-source interface designed for research applications. The work reported in this thesis has retrofitted a 25-year-old manual milling machine, which has previously been modified into a CNC mill, but whose interface is no longer functioning, into a functional CNC mill. New components have been incorporated as necessary, to produce an open-source CNC mill capable of use in research and as a standard CNC mill. The original milling machine has inconsistent backlash due to cheaper methods of construction, ranging up to 0.049-inch per axis. The thesis has implemented software compensation, capable of producing most CNC paths with only 0.015-inch of inaccuracy with a 99%-confidence interval. Though the final mill does not have all the built in functionality of a commercial mill, it is designed with research application in mind, and as such, it is intended as a platform for future research to be mounted upon. The open-source interface will allow many forms of integration within a flexible manufacturing environment and easy future customization as necessary.

# Acknowledgments

I would like to thank:

My family and friends, for their continued support, encouragement, and understanding of my unusual hours and availability during my university education; it has helped in many ways.

My thesis advisor, Dr. Subramaniam Balakrishnan, for his academic and technical guidance and advice; particularly when I could not make sense of what was malfunctioning.

Mr. Ken Tarte, for his technical advice, assistance in sourcing equipment, and tolerance of the noise produced by the CNC mill as I worked on it.

# Table of Contents

# List of Figures

# List of Tables

# Nomenclature

**2½D –** The term 2½D (two-and-one-half-dimensional) describes a part, or the requirements of a machine to produce that part. Unlike 3D (three-dimensional) parts, which can contain non-straight paths along all three axes, a 2½D part can only contain non-straight path in 2 axes, and only straight path in the third. The capabilities of these axes are not interchangeable. Commonly, two-dimensional designs will be produced in the X and Y planes, and the Z plane will only be used to adjust the height, but not while the two-dimensional design is being produced.

**ActiveX –** ActiveX controls are small program modules that follow a Microsoft defined structure. They can perform a wide variety of functions, and are intended for use as reusable code objects for use with various platforms. They are similar to Microsoft's COM technology.

**Backlash –** Backlash is the measure of play in a mechanical system that exists between two meshed objects. Play in a mechanical system, is mostly of concern when the system reverses direction. The reversal of the driving-direction causes a momentary situation where the driving-object is not in contact with the driven-object, which causes uncertainty in locating the driven-object.

**Benchman –** Benchman is the brand name of a small tabletop CNC mill that is utilized in the Integrated Manufacturing Laboratory.

**C –** The C programming language is a mid-level, general-purpose programming

language.

**C++ –** The C++ programming language is a mid-level, general-purpose programming language, based upon the C programming language.

**CNC** machine **–** Computer Numeric Controlled machine utilize a computer for reading G-code programs. CNC machines typically read their program from an electronic storage location, either locally or from a network location. Utilizing a computer for axis control, CNC machines can produce much smoother paths through multi-axis interpolation than are commonly possible by manual efforts.

**COM – "**Microsoft COM (Component Object Model) technology in the Microsoft Windows-family of Operating Systems enables software components to communicate. COM is used by developers to create re-usable software components, link components together to build applications, and take advantage of Windows services." [1]

**Concatenated –** Concatenation is the process of joining two items together by placing one on the end of another (e.g. adding words onto the end of a sentence).

**Confidence Interval –** A confidence interval is a statistical term, which specifies what the maximum deviation a given percentage of data within the set under analysis will contain, about the mean value of the set. See Appendix D

**Delimiter –** A delimiter is a character used to specify that the end of an entry in a list has occurred, and the next entry is about to start (e.g. the spaces in a sentence indicate that one word has ended and another is about to begin).

**Drift –** Drift is the situation whereby no intentional external power is applied to the power amplifier and the axis is in motion. Transient voltages cause drift, possibility due to improper grounding, or radio frequency interference such as from lights or high-voltage AC equipment.

**G-code –** G-code is the common name for the RS274 programming language. It is comprised of 25 registers (A-N and P-Z), each of which can store a numerical value [2]. Different registers have different meanings and applications for the numerical value that they store.

**HMI –** Human-Machine Interface. "For simpler control systems, buttons and switches are quite suitable for operator interfaces. However as the number of operator options increases, or the interface becomes more complicated it may be preferable to replace many of the buttons, dials, and indicators with a Human Machine Interface (HMI). These units can be as simple as a single line of text and a couple of push buttons. More complicated units use large color monitors with touch screen capabilities. Ultimately these units are very powerful because the display contents can be changed to match the mode of operation." [3]

**Integer –** An integer number is a number containing no decimal places.

**Java** – The Java programming language is a mid-level programming language that runs within a dedicated platform independent Virtual Machine. Due to the platform independent Virtual Machine, Java is capable of running on almost any computer platform. The Virtual Machine that Java runs within is platform dependent. Virtual Machines' exist for most common computer platforms.

**Manual Milling Machine –** A milling machine is a tool that is capable of quickly removing large amounts of material from the part in process, by the use of rotating cutting teeth or tool inserts. In the case of a 2½D and 3D vertical milling machine, the part in process is mounted to the table of the machine, and the cutting tool rotating above and through the part's material. In 2½D and 3D vertical mill setup, the cutting tool is permanently affixed to rotate about the Z-axis, while only being able to translate along the Z-axis, while the table is only capable of translation along the X and Y-axes. In a manual milling machine the axes are located by the operator. The operator positions the axes by manually turning a crank or wheel connected to the leadscrew of that axis.

**NC-file –** A NC-file is the common storage format for CNC programs when stored electronically. A NC-file is a text file containing the coded instructions that represent the procedure to be completed. The instructions are commonly coded in RS274 (G-code), but some CNC manufactures will occasionally define proprietary codes specific to their machines. Often in a Windows operating system environment, these files will have the file extension of NC.

**PI control system –** Proportional Integral control system. See Appendix B.

**PID control system –** Proportional Integral Derivative control system. See Appendix B.

**Real –** A real number is a number that existing between +/- infinity, and containing an infinite amount of possible decimal places.

**Rotary Incremental Encoder –** Is an active input device, which outputs a series

of coded signals as its input shaft rotates. Commonly, these devices will output a pair of quarter-cycle offset, square wave pulses, an A and B signal, and a once per revolution signal, a Z signal.

**Signed –** A signed number is a number that has a sign associated with its value, either positive or negative.

**String –** In the context of a programming variable, a string variable is a variable that is capable of holding one or more character variables. Often the contents of string variables are displayed within quotation marks.

**VB.NET –** VB.NET Is Microsoft's newer version of the Visual Basic programming language, designed for compatibility with Microsoft's new .NET framework.

**VB6 –** Visual Basic version 6 is a Microsoft variation on the Basic programming language, specially designed for compatibility with other Microsoft products such as COM and ActiveX.

**VBA –** Visual Basic for Applications is a version of the Visual Basic language specifically designed for use within Microsoft Office, particularly for the purpose of automating task. The version of Visual Basic for Applications that was used during this research was Visual Basic version 6.5 [4].

**VC++ –** Visual C++ is a Microsoft variation on the C++ programming language, specially designed for compatibility with other Microsoft products such as ActiveX, DirectX, and .NET.

**White-Space –** White-space is a catch-all term in computers for any character that produces space within a document, and whose character does not normally

visibly print on the computer screen (e.g. tabs and spaces).

**Working Envelope –** the working envelope defines the maximum area or volume that a machine can move through during any movement.

# Chapter 1
# Introduction

This research deals with the design of a custom-developed, computer numerical control interface. The custom-developed, computer numerical control interface in this thesis is orientated towards the practical application of the design of a functional **2½D CNC** mill for research and teaching purposes in the Integrated Manufacturing Laboratory. The mill being retrofit is a 25-year-old milling machine, as shown in Figure 1. It has been the focus of a CNC retrofit project in the past. However, the previous interface is no longer functioning. Remaining from the original retrofit are only the motors, their associated mounting brackets and power amplifiers, and the couplers that mechanically connect the motors to the mill.

The primary difference between a **manual milling machine** and a CNC mill is that in a CNC mill, motors connected to an electronic controller rotate the axes as opposed to the mill's operator. In this thesis, the controller is a desktop computer.

A CNC mill operates by a controller reading in a **NC-file** containing the desired geometry of the part to produce. Then, instructions arrive at each axis of the mill to move in a manner that will produce the desired finished piece. The degree of accuracy to which the axes can be coordinated and positioned determines the

level of complexity of parts, which the machine is capable of accurately producing.

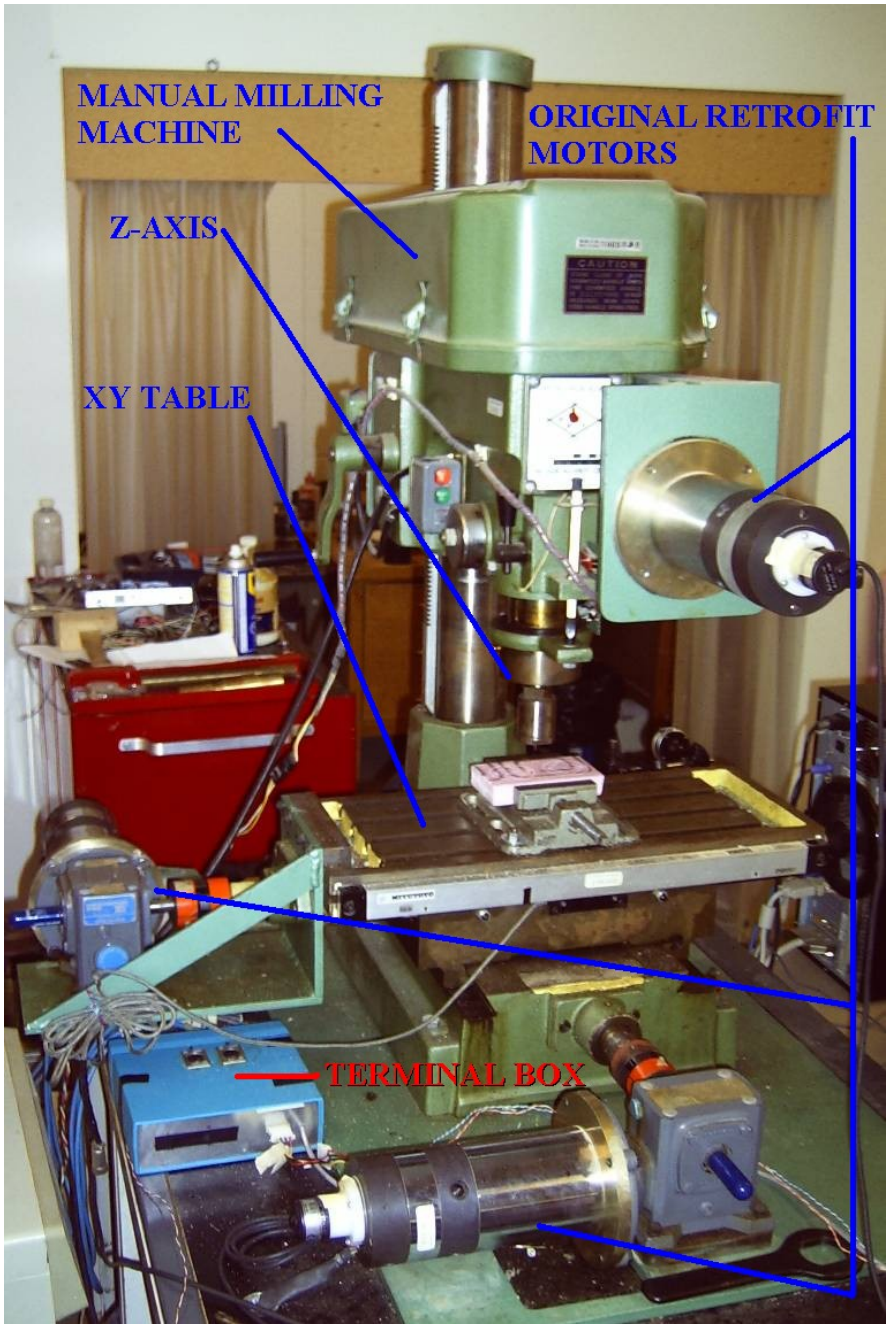


**Figure 1 - Retrofit Manual Milling Machine**

This project involves the design of a self-contained interface application, which is capable of operating the hardware of the CNC mill in a precise and accurate manner, given a standard NC-file input structure. Using the existing hardware

and additional hardware upgrades, the interface application will combine with the mechanical hardware to produce an open-source CNC mill.

## 1.1 Background

In the summer of 2008, working as a research assistant for Dr. Balakrishnan, work began on a motion control project to upgrade a previous custom-designed CNC mill, which utilized a previously developed interface for computerized operation. The previous-interface is not functional anymore, but the CNC mill has proven a valuable tool in completing many associated research projects. Part way through the summer, it was established that there was a sufficient amount of work to fulfill the requirements for an undergraduate thesis.

## 1.2 Purpose

The objective of this project is to utilize the existing hardware and integrating new components as necessary, to produce a 2½D CNC open-source milling machine for research and daily use, which is capable of running a standard NC-file, to produce predictable results at a reasonable feed rate.

## 1.3 Scope

The limiting scope of this thesis is the development of a functional 2½D milling machine for use in the Integrated Manufacturing Laboratory. This project intends to provide an open-source CNC mill orientated towards research applications, which is capable of linear and circular interpolation, which has full Imperial unit support, and which can accurately move at a reasonable feed rate. This CNC

interface provides support for several **G-code** commands including N, G, X, Y, Z, I, J, K and F, though the degree of functionality varies. Some support framework is in place in the source code for the following commands Q, R, S, T, D, H, M and P, but their functionality has not be implemented due to time limitations.

## 1.4 Significance

All feedback-based mechanisms utilize some form of a control system to operate precisely and accurately according to their design parameters. This project will demonstrate the development of a control system, which is calibrated for the particular use of controlling a manual milling machine for use as a CNC mill. The outcome of using a custom developed control interface to produce a CNC mill is that the final milling machine will be open source.

The significance of an open source CNC milling machine is that its calibration can be adjusted for any purpose. An open-source CNC mill interface has several positive implications. If external sensors are desirable for data collection, the mill is capable of collecting the required data, correlating it to when it occurred, and adjusting its characteristics to suite the situation, if desired. If better hardware becomes available at a later time, the newer hardware can be integrated and the interface re-calibrated, and the CNC mill can continue to be used with minimal downtime. In addition to all the flexibility that the custom developed control interface contributes, the mill is still capable of daily use as a metal-cutting-quality milling machine.

# Chapter 2
# Functional Design Aspects

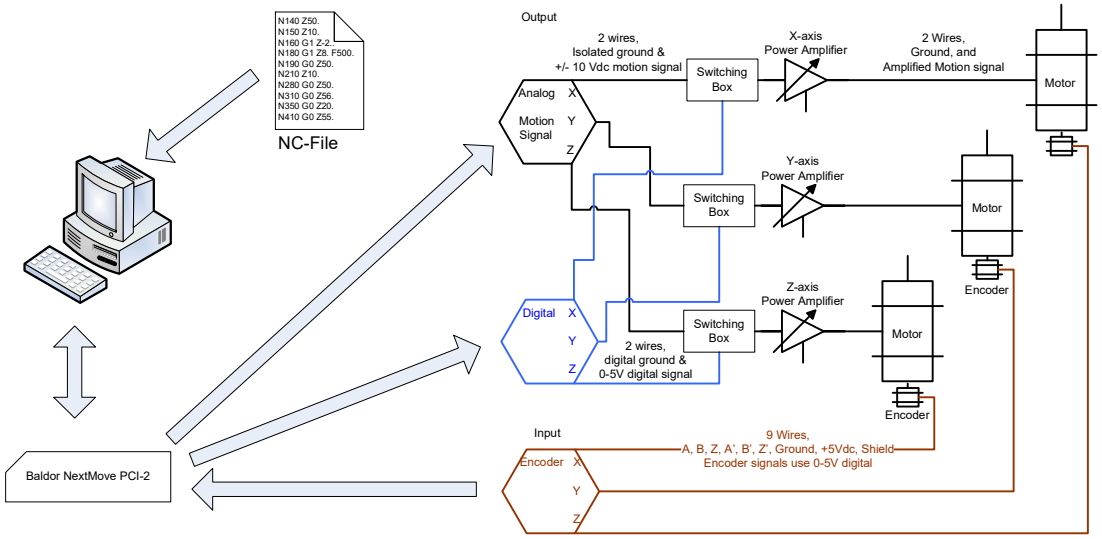


**Figure 2 - Design Summary**

The overall functionality of this project, centers on the combination of three closed-loop control systems, one per axis-motor, as shown in Figure 2, to locate the three open-loop axes. The purpose of these open-loop axes is to locate the tool tip to the desired position, as specified by an NC-file, relative to the mill table, and therefore the part to be machined as well.

## 2.1 System Overview

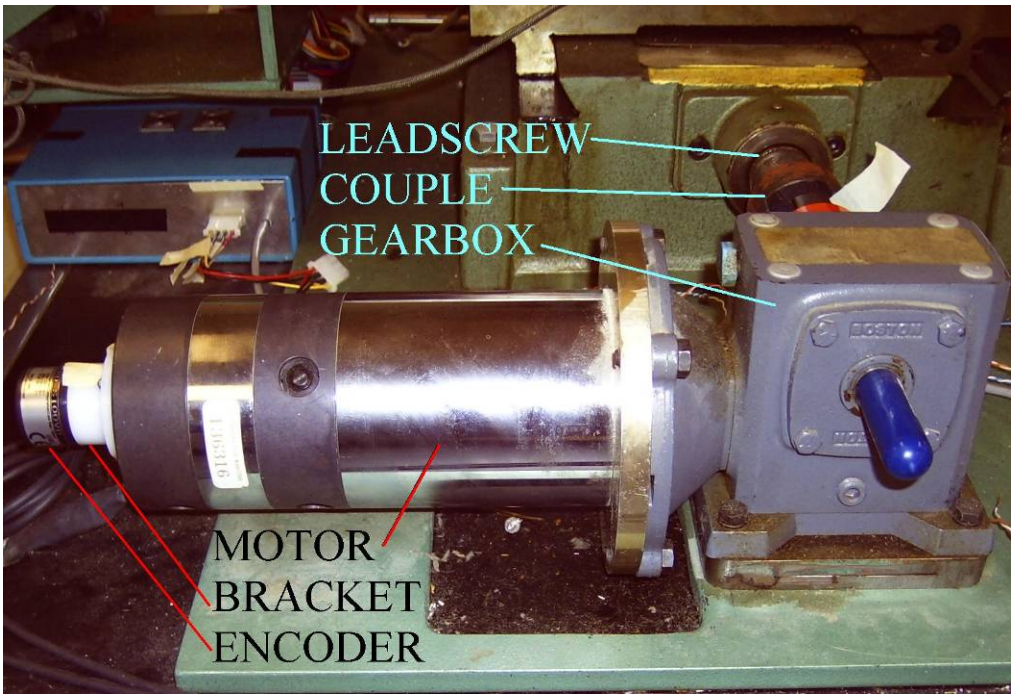


**Figure 3 - Servomotor Configuration**

Each axis consists of a DC motor and a feedback encoder, combining to create a servomotor that is mechanically coupled to the input shaft of one of the mill's axis leadscrews, as shown in Figure 3. As each motor rotates, the encoder attached at its rear sends positional displacement data back to the interface application. Using multiple iterations of the positional displacement data, velocity and acceleration are calculated. Using the full session history of the positional displacement data, the absolute theoretical position of each axis can be determined. If the actual position does not match the theoretical position, a modified DC voltage can be applied to the servomotor to adjust its rotational speed, thereby attempting to minimize the difference between future actual and theoretical positions.

## 2.2 Hardware

There are five major hardware component types in use in the closed-loop aspects of this project. No hardware relating to the open-loop control system, namely the mill's mechanical linkages, is discussed here. Discussion of the mill's mechanical linkages are inclusive to section 2.4, dealing with the control system.

### 2.2.1 **Motors and Power Amplifiers**

The three motors and power amplifier pairs that are used to control the motion of the mill's axes are remnant from the previous retrofit. Currently very little documentation regarding the specifications of either is available. All parameters regarding the motor's response characteristics were collected via an iterative process of motor excitation and encoder signal observation, until a reasonable set of parameters were known.

### 2.2.2 **Encoders**

Due to the requirements of the new interface card, the original encoders that were remnant from the previous retrofit needed to be replaced because they did not provide all of the data that the new interface card required. The new **rotary incremental encoders** were required to have not only the A, B, and Z outputs similar to the original encoders, but also their complements, the $\overline{A}, \overline{B}$ and $\overline{Z}$ signals [5]. This data, including quadrature positioning (see Appendix A), was made available by the installation of TRD-S100-VD [6] encoders on the rear of each motor. For incorporating of the new encoders, new mounting brackets were

designed and fabricated, and male DE-9 connectors were attached to the signal output wires of each encoder for connection to the new interface card [5].

### 2.2.3 **Servomotor Control Circuitry**

A requirement of the new interface card is the presence of disconnect circuitry for the control signal of each motor. The per axis control circuitry, shown in Figure 4, consists of a digitally controlled relay. The relay is capable of disconnecting the interface card from the power amplifier, when the interface application requires the motor to be stationary.

The disconnect circuitry was crucial for the initial minimization of the servomotors' **drift**.

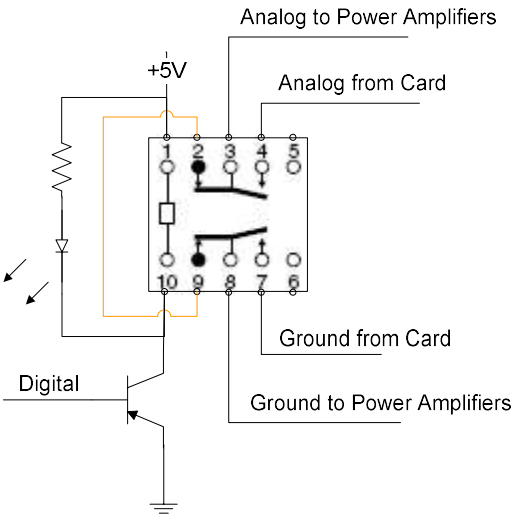


**Figure 4 - Control Circuitry for one Axis**

Any physical drift in the system would be undesirable in a CNC application, therefore it was decided that as high a speed as reasonably possibly would be desirable for the disconnect circuitry. Due to this requirement Panasonic TQ2–5V relays were selected, with a max switching latency of 3ms [7].

### 2.2.4 **Interface Card**

A primary factor that allowed this project to commence was the availability of the new interface card. The interface card is the bridge between the interface application and the sensors and actuators, which collect data from the mill and drive the mill's axes. Without the interface card, this project could not function.

The interface card and its accompanying software, a Baldor NextMove PCI 2 and its corresponding **ActiveX** control, have substantial built in functionality that are being leveraged in this project.

## 2.3 Software

There are two major components to the software in use, the ActiveX control and the interface application. The ActiveX control came with the interface card, and the interface application, which is custom developed for this project.

### 2.3.1 **ActiveX Control**

The ActiveX control is a standalone application file, which the host Windows based operating system executes. The ActiveX control allows whatever software is referencing it, to have read and write access to the memory components of the interface card, and to call certain functions from within the ActiveX control itself. Two vital functions that are being leveraged from the ActiveX control are the functions VectorA and HelixA. These two functions are explained in further detail in sections 2.3.2.2.

### 2.3.2 **Interface Application**

The interface application is a custom developed **VBA** application. Its major task is the coordination of all aspects of the CNC. It is responsible for configuring the interface card, and performing the actions specified by the input NC-file. The interface application creates an array of commands to execute based on the input NC-file and monitors the per axis motion of the mill as they execute. If any corrective motion is required, the interface application specifies the compensation.

#### 2.3.2.1 **File Input and Array Grooming**

The application receives an input file via a standard file-selection dialog-box. Upon selection, the application reads the file line by line. The content of each line is **concatenated** to the array of lines, in the file viewing-window of the interface application's **HMI**. Any lines deemed as not a comment, undergo further processing to create the command array. The definition of a comment is any line not starting with the character 'N'.

When a line is deemed as not a comment, the line is broken into a set of fields, by using the **white-space** as a **delimiter**, and each field is processed individually. Each field starts with a character A-Z, excluding the character O, followed by a **signed-real**-number, some of which are consistently **integers**.

The command array is composed of nine **string** array fields per command, designated by N, G, X, Y, Z, I, J, K and F. String arrays are required to differentiate between blank entries and entries having a value of zero.

Each array field from the current line is assigned to its corresponding field within the command array. If multiple G commands exist within the array of fields from the current line, they are broken into multiple commands within the command array. The multiple commands that are created contain a modified N field from the N field present in the original NC-file. The N fields effectively contain a line numbering scheme to assist in sequential execution, though not all of the sequential numbers are necessarily present. The modified N field, has a sequentially incrementing single decimal, concatenated to the original integer N field.

For example, if a single line from the NC-file contains “N100 G40 G20 G90 M30” it would be stored in three separate commands as shown in TABLE 1 (all nine fields are present for each command, but only the three relevant ones are shown in this example).

**TABLE 1 - SAMPLE COMMAND ARRAY**

| | N field | G field | M field |
|---|---|---|---|
| Command 1: | “100” | “90” | |
| Command 2: | “100.1” | “20” | |
| Command 3: | “100.2” | “40” | “30” |

As shown in TABLE 1, the order that multiple G fields are assigned to commands within the command array follows a right to left sequence, based on the order that they appeared in the original NC-file. The last G field for the current line is grouped with the remainder of the fields from that line. No documentation is available to support this heuristic, though based on multiple

NC-files compiled in the Integrated Manufacturing Laboratory for the **Benchman** CNC mill, this appears to be the only consistent sequence to follow.

After processing the entire NC-file, grooming of the command array begins. This entails filling in all the information that the NC-file compiler assumes the CNC interface would remember from previous commands. In the situation where the motion type is not specified the motion type is assumed as the same as the previous motion that occurred. In the situation where a command does not contain a feed rate, the information is retrieved from the first previous occurrence of a command with the same motion type. For commands where an axis is stationary while other axes move, the unspecified X, Y and Z coordinates are specified based on previous motion commands.

One at a time, the commands are analysed. Replacement of the missing information for the current command under analysis is achieved by first ensuring that a G field is present. If no G field is present within the command line, and any motion information is present (i.e. X, Y, Z, I, J, K, S or F fields), the missing G field is supplied from the first previous command with a G field. All other missing information is filled in using a similar process; commands are analysed for missing components (e.g. G00 and G01 need X, Y, Z and F, and G02 and G03 need X, Y, Z, F, I and J) and the information is supplied from the first previous occurrence that has a G field matching the G field of the current command. In the case of G00 codes, which seldom have a specified feed rate (F field) present, the grooming process assigns the maximum per axis feed rate.

The grooming process ensures that each command in the command array is a complete command that can execute independent of the rest of the command

array. This ensures that during the program execution, all relevant information is present and only the current command is required, and therefore the much larger command array does not need processing.

#### 2.3.2.2 **Command execution**

During the execution of the command array, only one command within the command array is accessed at any one time. Since each command, contains all information required for the execution of that command, all commands are processed in a similar fashion.

Regardless of the application interface not supporting M fields, M field's presence and content are used by the application interface to decide how the command array executes. Depending on the G and M fields' contents, the application interface will execute different fields in different sequences. If the M field is not present, or is equal to 0, 1, 2, 5, or 30 (these relate to stopping the tool spindle), the G field will execute first, followed by the M field. For all other situations, the M field will execute first, then followed by the G field [8]. Final X, Y and Z coordinates are all present within each command; therefore, the interface application is capable of specifying an absolute single vector for the interface card to have the mill move through.

In the case of drift having occurred since the last command executed, the present command specifies the final tool position in all three axes, and therefore the error from the drift is minimized instead of being carried through the rest of the program.

Since all commands within the command array contain their own feed rate, all motion commands can be grouped into two types, linear and circular interpolation. This requires only two standardized motion output commands.

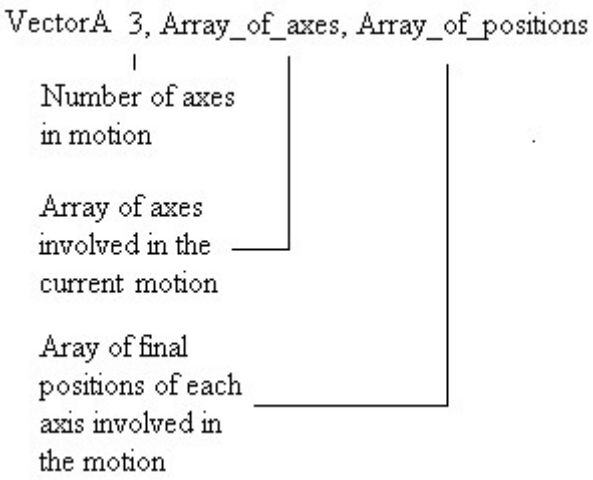


**Figure 5 - Anatomy of VectorA command**

For all linear interpolations, the VectorA command is used, and its anatomy is shown in Figure 5. VectorA accepts the absolute final positions of the vector motion path. Due to VectorA utilization of lowest feed rate from all axes involved in each motion path as the vector's feed rate, prior to the command array's execution, each axis can be assigned the same maximum feed rate [9].

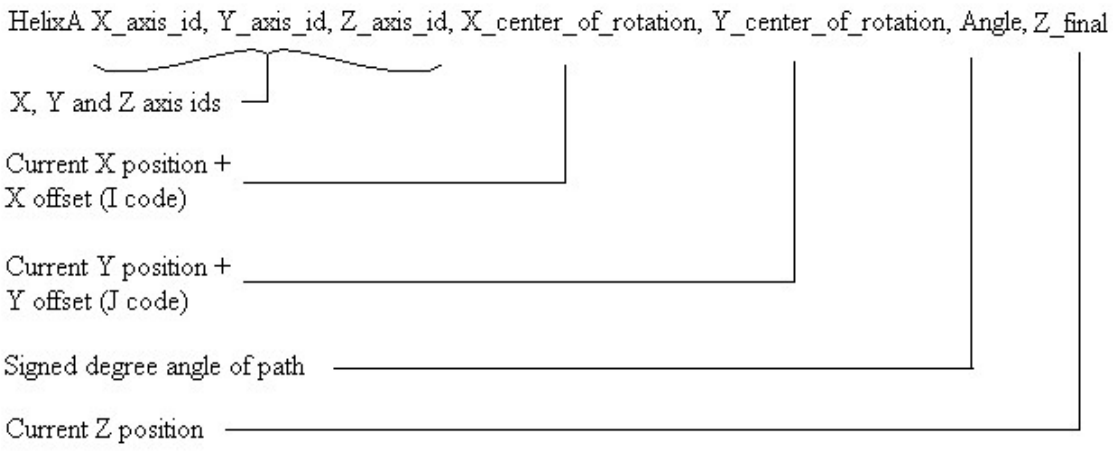


**Figure 6 - Anatomy of HelixA command**

For all circular interpolations, the HelixA command is used, and its anatomy is shown in Figure 6. HelixA is used to specify the arc that satisfies the circular interpolation specified. A helical path is required because it keeps all three axes active. By specifying the final Z coordinate, theoretically, the same as the starting Z coordinate; ensures that the Z-axis will be in the proper final position at the end of the arc's path. Similar to the VectorA command, HelixA uses the smallest feed rate of all axes in motion.

Since only two motion types exist in the interface application, a heuristic is used to associate the proper signed angle for circular interpolation, based on the supplied final coordinates, for distinguishing between clockwise and counter-clockwise interpolation.

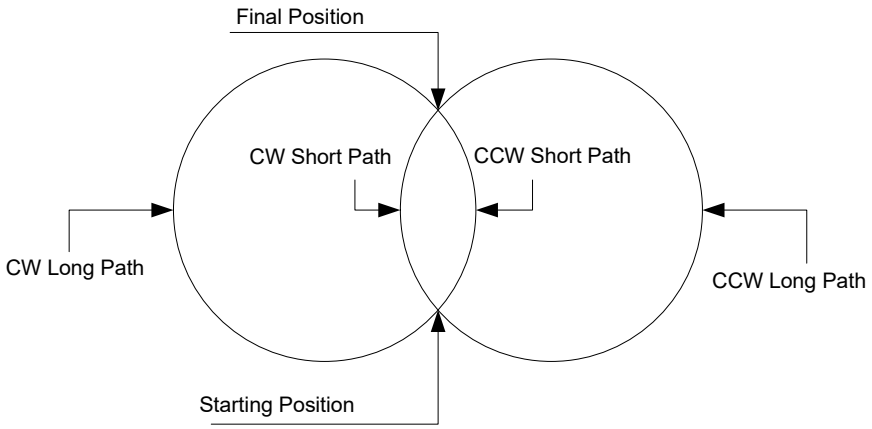


**Figure 7 - Circular Interpolation, Four Paths**

For all possible clockwise and counter-clockwise circular interpolations, four paths exist between the initial and final position. These four paths are illustrated in Figure 7. Depending on the specified I and J codes, only one of the paths will be appropriate for the specified center of interpolation and interpolation direction.

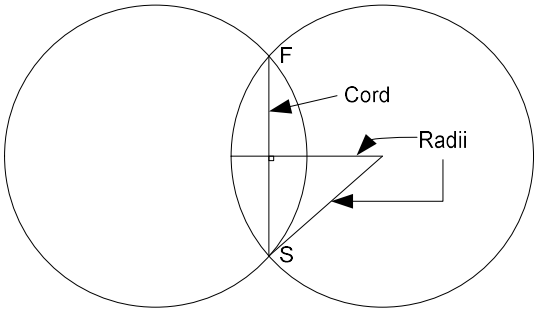


**Figure 8 - Circular Interpolation, Defining Cord Length**

**Equation 1 - Cord Length**

$$\text{cord length} = 2A = \sqrt{\left(x_F - x_S\right)^2 + \left(y_F - y_S\right)^2}$$

Since both the current and final positions are known, the cord length of the arc path is calculated as the vector length between the two points, as shown in Figure 8 and Equation 1. With the center point of the circular interpolation and the starting point known, the radius is calculated as the vector length between them as shown in Equation 2.

**Equation 2 - Radius of Interpolation**

$$\text{radius} = \text{C} = \sqrt{\left(x_S - x_{\text{Center}}\right)^2 + \left(y_S - y_{\text{Center}}\right)^2}$$

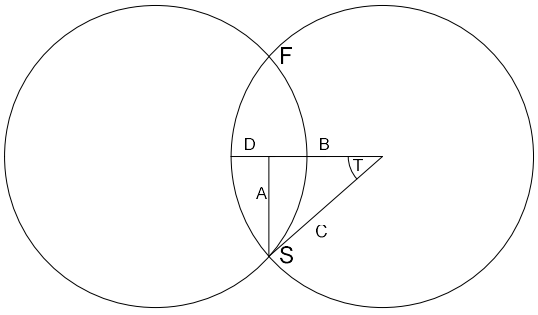


**Figure 9 - Circular Interpolation, Defining Minor Angle**

**Equation 3 - Arc Angle**

$$\tau = \arcsin\left(\frac{A}{C}\right)$$

Using half of the cord length (A) and the radius of interpolation (C), a right angle triangle is formed, as shown in Figure 9. With the right angle triangle, angle $\tau$ is solved for by using the arcsine of the quotient A over C, as shown in Equation 3. The angle $\tau$, is also the arc angle corresponding to one half of the cord length, and therefore the full arc angle is twice $\tau$.

From the X and Y location of the center of interpolation relative to the starting point, the angle of the vector between these two points relative to the X-axis can be calculated with arctangent, as shown in Equation 4. The two possible situations, for clockwise and counter-clockwise, are shown in Figure 10, denoted as u1 and u3, mutually hereafter referenced to as ux, for which ever is present. Similarly, the angle of the vector, between the starting and final positions, and the X-axis, denoted as u2, can be calculated with arctangent, as shown in Equation 5. All three of these angles are illustrated in Figure 10, showing their relative locations.

**Equation 4 – Magnitude of ux**

$$\arctan(ux) = \left(\frac{y_{Center} - y_S}{x_{Center} - x_S}\right)$$

**Equation 5 - Magnitude of u2**

$$\arctan(u2) = \left(\frac{y_F - y_S}{x_F - x_S}\right)$$

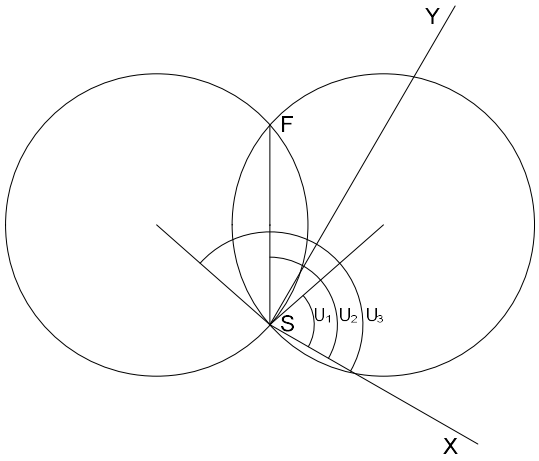


**Figure 10 - Circular Interpolation, Path Selection**

If the difference between the u2 and ux is positive, ux is less than u2 and therefore the center point is on the right of the cord. If the difference between the u2 and ux is negative, ux is greater than u2 and therefore the center point is on the left of the cord. Once the location of the center point is known, either the arc angle or 360 minus the arc angle can be specified as the path angle, with the appropriate sign being based on clockwise or counter-clockwise motion.

In the situation where the difference between u2 and ux is zero, the path is 180 degrees and the sign is specified by the direction of interpolation, either clockwise or counter-clockwise. The six previously mentioned possible paths are summarized in Equation 6.

**Equation 6 - Angle Heuristic**

$$\text{path angle} = \begin{cases} \textit{clockwise} & \begin{cases} u2 > ux & \left(360^0 - \text{arc angle}\right) \\ u2 < ux & -\left(360^0 - \text{arc angle}\right) \\ u2 = ux & -180^0 \end{cases} \\ \textit{counter} - \textit{clockwise} & \begin{cases} u2 > ux & -\left(\text{arc angle}\right) \\ u2 < ux & \left(\text{arc angle}\right) \\ u2 = ux & 180^0 \end{cases} \end{cases}$$

In the situation where the interpolation finishes at the same location that it started, the arc angle is known to be 360 degrees, with the appropriate sign being specified from the direction of interpolation, positive for counter-clockwise, and negative for clockwise.

## 2.4 Control System

The motion of each axis is governed by a first order **PID control system** (see Appendix B). The interface card's built-in PID control system requires accurate constants, based on physical parameters, for accurately governing the motion of each axis.

Since only a first order system is available from the interface card, the interface application has been designed to utilize trapezoidal motion profiles to minimize the physical jerk on the machine. Due to the use of trapezoidal motion profiles, the Ziegler–Nichols tuning method (see Appendix C) is inappropriate [10,11], and the system needed to be manually tuned. Based on an understanding of the anticipated use as a CNC mill, the PID control system is represented by a **PI control system**, because no high frequency position changes are anticipated. By iterative excitation and monitoring of the encoder feedback, the required system constants have been configured for each axis-motor. The final PI control system is capable of maintaining all three axes-motors within an equivalent of 0.0005-inch of output motion, operating at up to a maximum feed rate of 18-inches per minute.

# Chapter 3
# Design Challenges and Solutions

During this research unexpected challenges arose that required specific attention for the entire project to proceed. This chapter outlines these challenges and the solutions that allowed the project to progress.

## 3.1 Software Language Selection

The programming-language selection proved to be a much more involved process than initially expected. Initially, programming was assumed possible in **C++.** After further research, it became apparent that the interface card only supported a narrow selection of communication options. Of the available communication options, ActiveX was almost exclusively recommended over all other options by Balder Support [12]. Therefore, the decision was made that all software development would proceed with the requirement of ActiveX support for communicating with the interface card.

The ActiveX requirement placed a hindrance on the software development process, due to most stand-alone languages such as **C**, C++, and **Java**, requiring considerable **COM** knowledge to setup and adjust ActiveX controls [13,14]. Initial attempts to establish hardware communication via C++, proved unsuccessful and the language selection process moved to more high-level languages including **VC++** and **VB.NET**.

Both of VC++ and VB.NET possessed an easy method of establishing hardware communication. Upon further attempts to develop the interface application, it was concluded that neither of these high-level languages were acceptable, due to their high degree of inherent interactions with the host Windows operating system [15].

In working on unrelated activities, VBA demonstrated its capabilities as a standalone language. Upon further research, it was discovered that VBA is actually Visual Basic version 6.5 [4], a successor to **VB6** [16], which is one of the languages that is recommended by Baldor in some of its earlier documentation [17]. Based on its easy interaction with ActiveX controls [18] and lack of required operating system interaction, VBA's capabilities and Baldor's existing documentation, VBA proved to be a near ideal choice for the creation of the interface application.

## 3.2 Grounding

An unresolved concern is regarding the grounding of the entire system (computer, spindle motor, power amplifiers, and external power supply). It has been observed that upon changing the power outlet used, or changing the main power extension cord, upon sending an analog voltage to any of the power amplifiers, in any combination that involves the X or Y power amplifiers, will cause problems. All analog voltages will drop to the +/- 10-millivolt range instead of being in the +/- 10-volt range, and the current draw will approach 30-milliamps instead of maintaining under 1-milliamp. All nine permutation of interchanging the analog voltage signals from the interface card to the power amplifiers have

been attempted; in all cases, the situation always centered on the use of the X and Y amplifiers.

The assumption has been made that this a floating-ground issue between the three power amplifiers, and the situation has been temporally resolved by continuing to use the cord and outlet combination that have proven trouble free to date.

## 3.3 Relay Flicker

As shown in Figure 4, on page 8, an external 5-volt DC power supply is required to power the relays. The external power supply is required because of the electrical current requirements to energize the coils of all three relays simultaneously, has shown to typically cause the voltage being supplied from the interface card, to drop below the energization threshold of the relays. The result of the drop in voltage cause the relay's coils to de-energize, thereby decreasing the current draw from the interface card, and raising the interface card's voltage output back above the energization threshold of the relays, and re-energizing the relay's coils. This situation causes a high frequency flickering effect on the output of the relays.

The addition of a PNP transistor controlling the relay's ground connection removed the flicker. The external 5-volt power supply connected to the relays, energizes the relay's coils when a signal from the interface card's digital output energizes the transistor, completing the circuit.

## 3.4 Drift

Prior to the installation of disconnect relays between the interface card and the power amplifiers, drift was a major concern. After the installation of the relays, the drift decreased significantly, but was still present.

Later it was discovered that the previous use of the terminal box, which housed the male screw-on connections for the power amplifier's cables, had a common positive connection between all of the amplifiers, as opposed to the more typical common ground. Original wiring focused on the assumption that the common wiring was for the ground leads.

After rewiring the terminal box, ensuring all analog grounds were isolated back to the interface card, and correcting the wiring for the ground and demand leads from the interface card to the power amplifiers, all observations of drift have ceased.

## 3.5 Backlash

The **backlash** values that have been collected for the X and Y-axis are shown in TABLE 2. The Z-axis did not appear to have backlash, presumably due to gravity or a mechanical backlash compensation mechanism (e.g. spring tensioning coil).

The collection of backlash values was achieved by placing a dial gauge on the axis under observation. That axis was then moved until the dial gauge clearly indicated motion. The motion of the axis under observation was then reversed, and incrementally moved until the dial gauge indicated that the entire backlash

had been removed. Working in 0.005 and 0.0005-inch increments, the backlash for that location on the leadscrew was recorded.

**TABLE 2 - BACKLASH MEASUREMENTS FOR THE X AND Y AXIS IN INCHES**

| X | Y |
|---|---|
| 0.0400 | 0.0360 |
| 0.0480 | 0.0380 |
| 0.0490 | 0.0360 |
| 0.0460 | 0.0410 |
| 0.0480 | 0.0365 |
| 0.0485 | 0.0355 |
| 0.0485 | 0.0355 |
| 0.0470 | 0.0355 |
| 0.0485 | 0.0350 |
| 0.0475 | 0.0360 |
| 0.0480 | 0.0350 |
| 0.0490 | 0.0350 |

Backlash samples were taken at 12 locations along the travel of each axis under observation. Based on the sets of backlash data collected, the analysis shown in TABLE 3 was conducted. The consequence of this data analysis shows that if the mean backlash for that axis is applied to that axis whenever a change of direction occurs, the actual final position of that axis will be within the absolute difference of the mean and max value of the data set for that axis, from its theoretical position.

**TABLE 3 - ANALYSIS OF BACKLASH DATA SETS**

| | X | Y |
|---|---|---|
| Max of the backlash data set | 0.0490 | 0.0410 |
| Mean of the backlash data set | 0.0473 | 0.0363 |
| Absolute difference between Mean and Max | 0.0017 | 0.0048 |

The application of a constant backlash offset (the mean of the backlash data set for that axis) for all motions, is coordinated by the interface card, and takes place over a defined period.

Ideally, the period over which the backlash is removed should be as small as possible. Given the size of the constant backlash offset, it is necessary for the period for backlash removal to be 1500ms. The long period of backlash removal is required to avoid loosing stability within the control system, due to the backlash compensation demanding the axis move at a rate above its modeled characteristics, thereby causing either instability within the PI control system or the positional error to exceed its predefined maximum limit.

## 3.6 Circular interpolation

A concern realized during the testing phase, is due to the range of possible output values from the inverse trigonometric functions and their implications on the circular interpolation heuristics.

The circular interpolation heuristics are based on the assumption that all angular values are between 0 and $2\pi$. In some situations, particularly those represented by Figure 7, on page 15, being rotated a quarter-revolution in either direction, caused the circular interpolation heuristic to fail. This was due the range limitation on the inputs to the inverse trigonometric functions.

The inverse trigonometric functions have their range limited to an interval of $+/-\pi$. To resolve this, secondary heuristics offset any 0 to $-\pi$ values that occur, by a $2\pi$ increment, bringing the angular measurements back into the 0 to $2\pi$ range.

Another concern that arose about the circular interpolation is in regards to locking up to the final position. Since in the initial circular interpolation sequence, only the X and Y-axis were active, and therefore capable of moving, if any error

were present in the Z location; the system could never reach the final position of the circular interpolation, because it was unable to correct the error on the Z-axis.

To allow the system to reach the final circular interpolation position when error is presenting on the Z-axis, a helical interpolation is used to represent the motion of the circular interpolation. By using a helical interpolation instead of a circular interpolation, and specifying the final Z location, which ideally is the current Z location, if any error is present on the Z-axis, it will be compensated for, and the program will not freeze waiting for the circular interpolation to finish a move that it is incapable of completing. In testing this resolved the program freezes that were caused by minor errors in the Z-axis motion.

# Chapter 4
# Performance Analysis

The analyses of the functional capabilities, of the finalized CNC mill are outlined in this chapter. The requirements of the test-part will be discussed, as well as the analysis of the parts produced.

## 4.1 Initial Test-Part

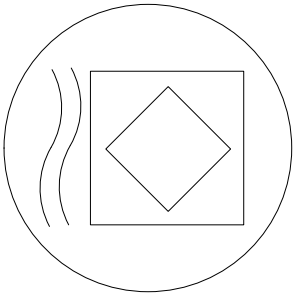

**Figure 11 - Initial Test-Part**

The initial test-part, whose overhead view is shown in Figure 11, was anticipated to be between 4-8-inch$^2$. It comprised of linear and circular cut paths and was meant to test the functionality of the machine. Due to clamping limitations, table travel considerations, and the resulting effect that these would have on the size of the details present on the test-part, a revised test-part was designed.

## 4.2 Revised test part

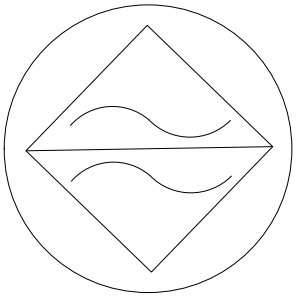

**Figure 12 - Finalized Test-Part**

The revised test-part, shown in Figure 12, allows for larger details within a similar layout to the original test-part design. After finalizing the diameter of the outer circle at 2-inches, this still allowed approximately ¼-inch displacement between the maxima and minima points along the contoured paths inside the diamond. Since this size will utilize roughly 10% of the mill's approximate 45-inch$^2$ **working envelope** in the XY plane, this scale is anticipated to provide adequate detail for data collection of the capabilities of the finalized CNC mill.

One limitation that is present in the finalized test-part that was not noticed until the analysis phase is the lack of ability to measure the precision of single axis motion. In the original test-part design, two sets of parallel line comprised the central square. These parallel lines would have allowed for the analysis of single axis motion in the XY plane. In the finalized test-part, only one line is created by single axis motion, therefore no relative or absolute reference can be used, while measuring with hand-tools, to verify the straightness of this line.

## 4.3 Testing

Testing is conducted on three parts produced on the research mill, and one part from the Benchman mill from the Intergraded Manufacturing Laboratory. The data collected is analysed for accuracy, and the part from the Benchman mill is used for contrast. Precision calculations are preformed using statistical analysis on the data from the three parts produced on the research mill.

### 4.3.1 **Measured Distances**

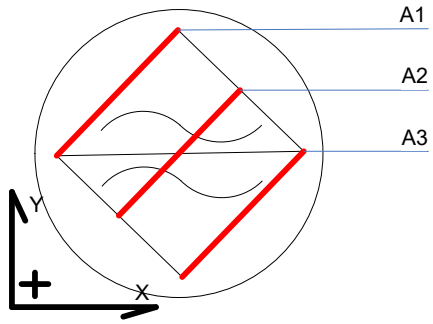


**Figure 13 - Linear Interpolation, set A**

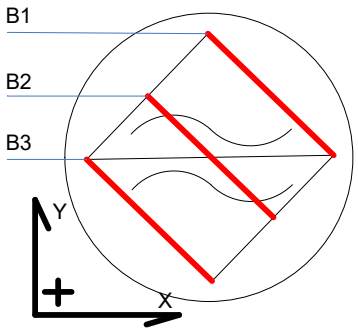


**Figure 14 - Linear Interpolation, set B**

Data set A and B, shown in Figure 13 and Figure 14, are used to assess the research mill's capabilities for multi-axis linear interpolation in the XY plane, and indirectly linear backlash compensation at the four corners.

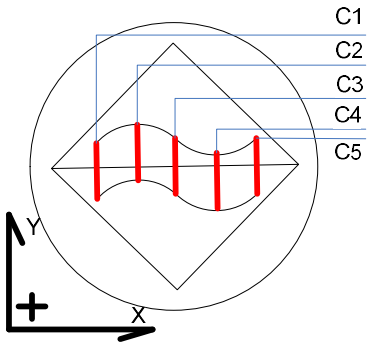


**Figure 15 - Circular Interpolation, set C**

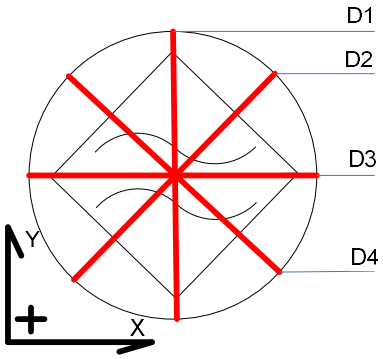


**Figure 16 - Circular Interpolation, set D**

Data collected from sets C and D, shown in Figure 15 and Figure 16, are used to assess the research mill's capabilities for circular interpolation. Set C particularly address the non-linear path precision in the XY plane, as the two curves are theoretically the same distance apart at all points along their path. Set D particularly address large-angle circular interpolation precision and backlash compensation. Points D2 and D4 occur after all backlash is removed, and points D1 and D2 are the location where the backlash compensation is visibly present. All points in the D set ideally should be the same length.

**TABLE 4 - MEASURED DISTANCES**

| | Part 1 | Part 2 | Part 3 | Benchman |
|---|---|---|---|---|
| A1 | 1.182 | 1.194 | 1.200 | 1.188 |
| A2 | 1.182 | 1.192 | 1.198 | 1.178 |
| A3 | 1.184 | 1.195 | 1.191 | 1.180 |
| | | | | |
| B1 | 1.182 | 1.184 | 1.195 | 1.190 |
| B2 | 1.183 | 1.183 | 1.186 | 1.183 |
| B3 | 1.188 | 1.186 | 1.186 | 1.186 |
| | | | | |
| C1 | 0.426 | 0.434 | 0.445 | 0.421 |
| C2 | 0.436 | 0.436 | 0.438 | 0.429 |
| C3 | 0.424 | 0.421 | 0.432 | 0.402 |
| C4 | 0.436 | 0.438 | 0.438 | 0.429 |
| C5 | 0.428 | 0.430 | 0.445 | 0.420 |
| | | | | |
| D1 | 2.013 | 2.009 | 2.032 | 2.010 |
| D2 | 1.973 | 1.981 | 1.985 | 1.990 |
| D3 | 2.003 | 2.000 | 2.060 | 2.005 |
| D4 | 1.984 | 1.981 | 1.992 | 2.005 |

All distances are measured using Vernier dial callipers, having a resolution of 0.001-inch, with an unknown accuracy, and the minimum distance between the points is used in the analysis. The measured distances are shown in TABLE 4. Due to the diamond profile on the test-part being designed with four equal sides, the A and B data sets are grouped together, hereafter referred to as set AB.

## 4.4 Accuracy

As shown in TABLE 5, based on the means of the measured distances of the three research mill parts and the Benchman part, an absolute discrepancy is calculated per location, and the means of the absolute discrepancy are calculated. As shown by the AB mean of the absolute discrepancy, for multi-axis linear interpolation motion in the XY plane, the research mill appears to have an average deviance from the designed-path of approximately 0.012-inch, whereas

the Benchman mill has an average approximate designed-path deviance of 0.016-inch.

**TABLE 5 - ACCURACY ANALYSIS**

| | Mean of three parts | Benchman | As Designed | Research Mill Absolute Discrepancy | Benchman Absolute Discrepancy |
|---|---|---|---|---|---|
| A1 | 1.1920 | 1.188 | 1.2000 | 0.0080 | 0.012 |
| A2 | 1.1907 | 1.178 | 1.2000 | 0.0093 | 0.022 |
| A3 | 1.1900 | 1.180 | 1.2000 | 0.0100 | 0.02 |
| A Mean | 1.1909 | 1.1820 | 1.2000 | 0.0091 | 0.0180 |
| B1 | 1.1870 | 1.190 | 1.2000 | 0.0130 | 0.01 |
| B2 | 1.1840 | 1.183 | 1.2000 | 0.0160 | 0.017 |
| B3 | 1.1867 | 1.186 | 1.2000 | 0.0133 | 0.014 |
| B Mean | 1.1859 | 1.1863 | 1.2000 | 0.0141 | 0.0137 |
| AB Mean | 1.1884 | 1.1842 | 1.2000 | 0.0116 | 0.0158 |
| C1 | 0.4350 | 0.421 | 0.4219 | 0.0131 | 0.0009 |
| C2 | 0.4367 | 0.429 | 0.4219 | 0.0148 | 0.0071 |
| C3 | 0.4257 | 0.402 | 0.4219 | 0.0038 | 0.0199 |
| C4 | 0.4373 | 0.429 | 0.4219 | 0.0154 | 0.0071 |
| C5 | 0.4343 | 0.420 | 0.4219 | 0.0124 | 0.0019 |
| C Mean | 0.4338 | 0.4202 | 0.4219 | 0.0119 | 0.0074 |
| D1 | 2.0180 | 2.010 | 2.0000 | 0.0180 | 0.01 |
| D2 | 1.9797 | 1.990 | 2.0000 | 0.0203 | 0.01 |
| D3 | 2.0210 | 2.005 | 2.0000 | 0.0210 | 0.005 |
| D4 | 1.9857 | 2.005 | 2.0000 | 0.0143 | 0.005 |
| D Mean | 2.0011 | 2.0025 | 2.0000 | 0.0184 | 0.0075 |

For non-linear parallel paths in the XY plane, the research mill has an average deviance from the designed-path of approximately 0.012-inch, whereas on the Benchman mill this is approximately 0.007-inch. For large angle circular interpolation in the XY plane, the research mill has an average deviance from the designed-path of approximately 0.018-inch, whereas the Benchman mill's designed-path deviance is approximately 0.008-inch.

## 4.5 Precision

The **confidence intervals** (see Appendix D) for the data sets AB, C and D are respectively 0.0035, 0.0045, and 0.0177, based on a 99%-confidence level. This corresponds to the predictions that 99% of the paths produced on the research mill:

- Involving precise tool locating using multi-axis linear interpolation in the XY plane will be within 0.0035-inch of the average location.
- Involving precise tool locating using circular interpolation in the XY plane will be within 0.0045-inch of the average location.
- Involving large angle circular interpolation in the XY plane will be within 0.0177-inch of the average location.

This means that for most precision work not relating to large-angle circular interpolation in the XY plane, it is assumed that, for the majority of the research mill's use, it will have a consistency of approximately 0.005-inch. It is assumed that, for the majority of the research mill's use for large-angle circular interpolation, it will have a consistency of approximately 0.018-inch.

## 4.6 Analysis Summary

Based on the raw data presented in TABLE 4, it is concluded that:

- 99% of the multi-axis linear paths produced in the XY plane with the research mill will be within 0.0116 +/- 0.0035-inch of the designed location. This corresponds to only 1% of the multi-axis linear paths produced in the XY plane to deviate by more than 0.0141-inch.

- 99% of the non-linear paths produced in the XY plane with the research mill will be within 0.0119 +/- 0.0045-inch of the designed location. This corresponds to only 1% of the non-linear paths produced in the XY plane to deviate by more than 0.0164-inch.
- 99% of the large-angle non-linear paths produced in the XY plane with the research mill will be within 0.0184 +/- 0.0177-inch of the designed location. This corresponds to only 1% of the large-angle paths produced in the XY plane to deviate by more than 0.0361-inch.

The accuracy of this analysis is based upon two of the test-parts produced on the research mill being cut in hard plastic, and one being cut in pink-insulation foam board, and the one test-part being produced on the Benchman mill, being cut in pink-insulation foam board. Due to the easily deformable nature of the pink-insulation foam board, the quality of the measurements taken is a possible source of error in this analysis.

No analysis on the Z-axis could be completed due to the use of a 1/16 end mill for the milling operation. Only realized during the analysis phase, the 1/16 end mill produced a tool path narrower than the depth gauge of the Vernier dial calliper, and therefore no data could be collected.

# Chapter 5
# Conclusion

Given the theatrical and analytically based performance expectation outlined in sections 2.4 and 4.6, it is my belief that the custom developed CNC interface used for controlling this mill, assuming the previously mentioned limitations are appropriately considered, will be capable of fulfilling a majority of the CNC milling tasks assigned to it.

As this is an open-source CNC mill, what has been established during this project is only meant to be a reliable platform for future research to utilize as a starting point. There are many aspects of the interface application that can be expanded on to provide a more robust set of CNC functions, but they have been deemed outside the scope of this thesis.

# References


[1] Microsoft Corporation. (2010). "*COM: Component Object Model Technologies*" [Online]. Available: http://www.microsoft.com/com/default.mspx [January 24, 2010].

[2] T.Kramer, F.Proctor, and E.Messina. (2000). "3 Input: The RS274/NGC Language," in *The NIST RS274NGC Interpreter - Version 3* [Online], Gaithersburg, Maryland: Intelligent Systems Division, National Institute of Standards and Technology. Available: http://www.isd.mel.nist.gov/personnel/kramer/pubs/RS274NGC_3.web/RS274NGC_33a.html#999262 [February 8, 2010]

[3] H.Jack. (2009, Aug. 31). "*29.1 Introduction*" [Online]. Available: http://hughjack.com/book_plcs/plc_hmia2.html [February 08, 2010].

[4] Microsoft Corporation. *Microsoft Visual Basic 6.5 version 1024* About File. United States of America: Microsoft Corporation, 2003.

[5] Baldor UK Ltd. (2007, Apr.). "*NextMove PCI-2 Motion Controller - Installation Manual - MN1933*" [Online]. Available: http://www.baldor.com/downloads/manuals/_downloads/MN1933_04-07.pdf [January 21, 2010].

[6] Automationdirect.com. (2007, Nov.). "*Light Duty Incremental Encoders*" [Online]. Available: http://web6.automationdirect.com/static/specs/encoderld.pdf [July 13, 2009].

[7] Matsushita Electric Works Ltd. (2005, Feb.). "*Low Profile 2 Form C Relay, TQ Relays*" [Online]. Available: http://pewa.panasonic.com/pcsd/product/sign/pdf_cat/tq.pdf [July 12, 2009].

[8] M.Fitzpatrick, *Computer Numeric Control Simplified*. United States of America: McGraw-Hill, 1989, pp. 13-24.

[9] Baldor UK Ltd. *Mint ActiveX Help, HF0002A0* Help File. Bristol, United Kingdom: Baldor UK Ltd, 2003.

[10] "Measurements and Control", Class notes for Mech 3430, Mechanical and Manufacturing Engineering, University of Manitoba, Winter, 2008.

[11] "Mechatronics System Design", Class notes for Mech 4900, Mechanical and Manufacturing Engineering, University of Manitoba, Fall, 2008.

[12] J.Brokaw. (2008, June 05). "RE: Support Required - from University Of Manitoba" Personal e-mail.

[13] J.Glatt. (2006, Mar. 28). "*COM in plain C*" [Online]. Available: http://www.codeproject.com/KB/COM/com_in_c1.aspx [July 12, 2009].

[14] M.Dunn. (2000, June 01). "*Introduction to COM - What It Is and How to Use It*" [Online]. Available: http://www.codeproject.com/KB/COM/comintro.aspx [July 12, 2009].

[15] D.Chapman, *Sams Teach Yourself Visual C++ 6 in 21 days*. Indianapolis, Indiana: Sams Publishing, 1998.

[16] Microsoft Corporation, *Visual Basic® User's Guide Microsoft® Excel Version 5.0 Automating, Customizing, and Programming in Microsoft Excel with the Microsoft Visual Basic Programming System, Applications Edition Microsoft Corporation*. United States of America: Microsoft Press, 1993.

[17] Baldor UK Ltd. (2001, May). "*MN1278 - Mint™ version 4 PC Programming Guide*" [Online]. Available: http://www.baldor.com/downloads/manuals/_downloads/1278-501.pdf [January 21, 2010].

[18] Microsoft Corporation. (2004, June 29). "*How To Call a VB ActiveX Server from a VBA Application*" [Online]. Available: http://support.microsoft.com/kb/185731 [June 12, 2009].

[19] unknown, (2010, April 1), "pid.jpg" in *Google Image Search* [Online], Available: http://radhesh.files.wordpress.com/2008/05/pid.jpg [April 1, 2010].

[20] William Navidi, *Statistics for Engineers and Scientists*. New York, New York: McGraw Hill, 2006.

[21] Microsoft Corperation. (2003). "*CONFIDENCE*" [Online]. Available: http://office.microsoft.com/en-ca/excel/HP052090211033.aspx [September 04, 2010].

# Appendix A
# Quadrature Positioning

Rotary incremental encoders capable of quadrature positioning have two signal outputs, each outputting the same number encoder positions per revolution. These signals are offset from each other as shown in the signal outputs of Figure A-1, as labelled by channel X and Y.

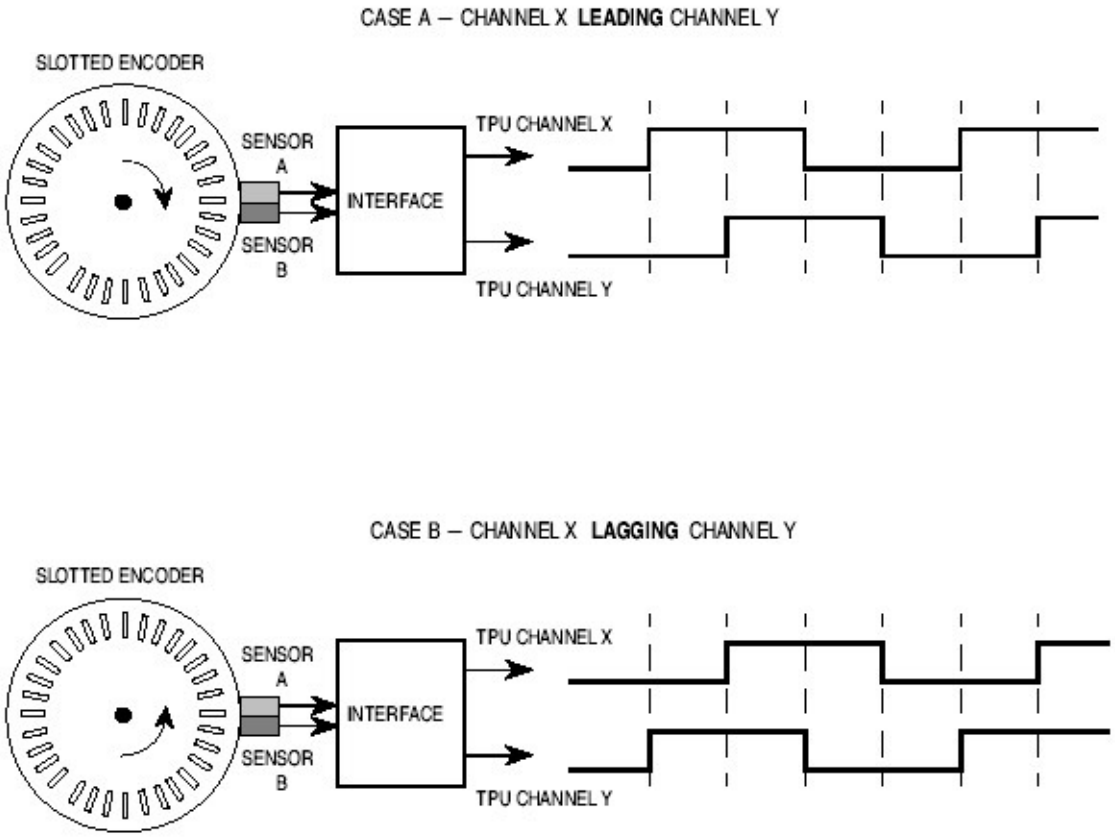


**Figure A-1 - Quadrature Positioning [19]**

Quadrature decoding utilizes the relative offset of the two signals to create four times the number of encoder positions. This is possible by the fact that Channel

X is 90 degrees out of phase with Channel Y. This phase difference allows the on-off state of each encoder pulse to be subdivided into four regions. Each sub-region represents one quarter of one full encoder cycle's pulse displacement.

**TABLE A-1 - QUADRATURE POSITIONING**

| Forward Travel (Case A) | | Reverse Travel (Case B) | |
|---|---|---|---|
| Channel X | Channel Y | Channel X | Channel Y |
| 0 to 1 transition | 0 | 0 to 1 transition | 1 |
| 1 to 0 transition | 1 | 1 to 0 transition | 0 |
| 1 | 0 to 1 transition | 0 | 0 to 1 transition |
| 0 | 1 to 0 transition | 1 | 1 to 0 transition |

When these quarter-pulse regions are interpreted in a Boolean context, they generate a table similar to that of TABLE A-1. As shown in TABLE A-1, by collecting a quarter-cycle of data, not only is that quarter cycle's displacement given, but also direction of rotation because each of the eight possible permutation pairs' are unique.

# Appendix B
# PID Control System

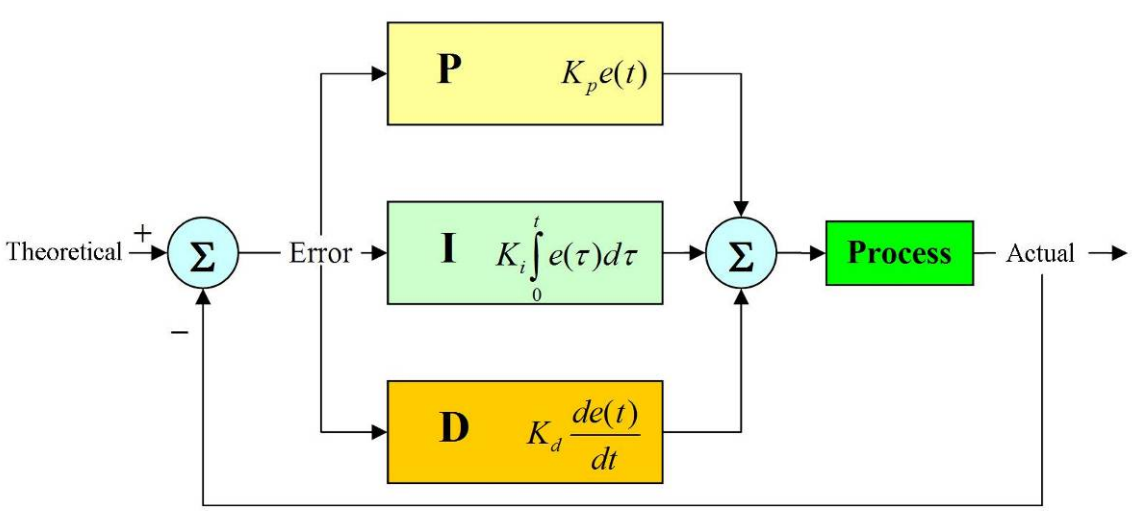


**Figure B-1 - PID block diagram [19]**

Figure B-1 illustrates, via a block diagram, the functionality of a first order PID system. It is considered a first order PID system because it only considers the first derivative and first integral of the error.

The difference of the actual position to the theoretical position is used to adjust the process output accordingly, to minimize the error of future iterations.

Minimizing the error of the next iteration entails, the Actual value being subtracted from the Theoretical value, and the resulting error and its sample-time respective derivative and integral, being multiplied by system constants. The sum, of these three products, are used to determine the input to the process for the following iteration.

# Appendix C
# Ziegler-Nichols Tuning Method

The Ziegler-Nichols closed-loop tuning method was developed in the 1940's, and to date is still taught as a starting point for developing a first order PID control system [10,11]. Based on the system's response to a step input, the heuristic states that if a suitable $K_U$ term can be found to set the system into steady osculation, and that osculation has a period of $P_U$, then $K_P$, $K_I$, and $K_D$ can be set, based on TABLE C-1.

**TABLE C-1 - ZIEGLER-NICHOLS CLOSED-LOOP TUNING HEURISTIC**

| | $K_P$ | $K_I$ | $K_D$ |
|---|---|---|---|
| P | $K_U/2$ | | |
| PI | $K_U/2.2$ | $P_U/1.2$ | |
| PID | $K_U/1.7$ | $P_U/2$ | $P_U/8$ |

Due to the requirement of a step input for establishing steady osculation, the Ziegler-Nichols closed-loop tuning method is inapplicable for establishing a set of initial PID constants for calibrating the control system for the axes of the research mill.

# Appendix D
# Confidence Interval

**Equation D-1 - Confidence Interval Calculation [20]**

$$\mu = z_{a/2} \times \left( \frac{s}{\sqrt{n}} \right)$$

The calculation of a confidence interval, as shown in Equation D-1, is based on normally distributed data sample, where s is the sample's standard deviation, n is the number of samples and z is a function of the desired percentage that the confidence interval is accurate for [20]. The confidence interval allows data samples to be used to predict the upper and lower extremes of future data, in relation to the sample mean, as shown in Equation D-2, where $\overline{U}$ denotes the sample mean.

**Equation D-2 - Predicted Upper and Lower Limits**

$$\overline{U} \pm \mu$$

**Equation D-3 - Calculation of α**

$$\alpha = 1 - (\text{Desired level of confidence \%})$$

During the analysis phase, the data was analysed using Microsoft Excel's built in confidence interval equation [21], which utilizes a pre-calculated sample standard deviation, n as above, and $\alpha$, as shown in Equation D-3.