X-Beam Alignment System

Sponsored by

Advisor:
Professor Eileen Rossman
erossman@calpoly.edu

Overseer:
Ian Davison
iandavison@microvu.com

Team X:

Joseph Falcao
jfalcao@calpoly.edu

Whittaker Hamill
whamill@calpoly.edu

Robert Tam
rotam@calpoly.edu

Date: Jun 08, 2017
Statement of Disclaimer

Since this project is a result of a class assignment, it has been graded and accepted as fulfillment of the course requirements. Acceptance does not imply technical accuracy or reliability. Any use of information in this report is done at the risk of the user. These risks may include catastrophic failure of the device or infringement of patent or copyright laws. California Polytechnic State University at San Luis Obispo and its staff cannot be held liable for any use or misuse of the project.
# Table of Contents

List of Figures ......................................................................................................................... xii
List of Tables ................................................................................................................................. xiii

1. Introduction ................................................................................................................................. 1

2. Background ................................................................................................................................. 1

2.1. Senior Project .......................................................................................................................... 1

2.2. Sponsor Background and Needs ............................................................................................ 2

2.3. Existing Products and Specifications ..................................................................................... 4

2.3.1. FARO Laser Tracker ......................................................................................................... 4

2.3.2. Keysight Technologies Interferometer ............................................................................... 4

2.3.3. Heidenhain Metro ............................................................................................................. 6

3. Objectives ................................................................................................................................. 7

3.1. Problem Statement ................................................................................................................ 7

3.2. Customer Requirements ....................................................................................................... 7

3.2.1. Rails Must Be Straight ..................................................................................................... 7

3.2.2. Rails Must Be parallel to Each Other ............................................................................. 7

3.2.3. Automated Rail Alignment .............................................................................................. 7

3.2.4 Automated Torqueing of the Screws ............................................................................... 8

3.2.5. Fast ................................................................................................................................. 8

3.2.6. Low-Skill Level ............................................................................................................... 8

3.2.7. Must Fit on a Table ......................................................................................................... 8

3.2.8. Must have a Length to Accommodate X-Beam ................................................................. 8

3.2.9. Scalability ....................................................................................................................... 8

3.2.10. Must Use Power Available at the Site .......................................................................... 9

3.2.11. Beam Must Be Sufficiently Constrained ....................................................................... 9

3.2.12. Basic Safety .................................................................................................................. 9

3.2.13. Must Convey Error to Operator .................................................................................. 9
3.2.14. Reusable ................................................................. 9
3.2.15. Repeatable ............................................................ 9
3.2.16. Cost ................................................................. 10

3.3: Engineering Specifications ............................................. 10
3.3.1. Straightness ....................................................... 12
3.3.2. Parallelism ....................................................... 12
3.3.3. Automated Rail Alignment .................................... 12
3.3.3. Automated Torqueing of Screws ............................. 12
3.3.4. Fast .............................................................. 12
3.3.5. Low-Skill Level .................................................. 13
3.3.6. Must Work on a Table ........................................ 13
3.3.7. Must Fit the X-Beam ........................................... 13
3.3.8. Must Use Power Available at the Site ...................... 13
3.3.9. Beam Must Be Constrained ................................. 14
3.3.10. Basic Safety .................................................... 14
3.3.11. Lifetime ......................................................... 15
3.3.12. Reliability ...................................................... 15
3.3.13. Cost ........................................................... 15
3.3.14. Loading ......................................................... 15

3.4. Quality Function Deployment ........................................ 15
3.5. Comparison of Specifications ...................................... 15

4. Design Process ............................................................... 17
4.1. Ideation and Selection of Design .................................... 17
4.1.1. Concept Generation ............................................ 17

Brain writing ............................................................. 17
Brainstorming ........................................................... 18
SCAMPER ............................................................... 18
4.1.2. Design Concepts

Design 1. Fixed Position Feedback Actuators
Design 2. Moving Gantry with Actuators and Two Sensors
Design 3. Moving Sensor Gantry with Fixed Actuators
Design 4. Moving Beam with Fixed Actuators and Sensors
Design 5. Moving Gantry with Actuators and Three Sensors
Design 6. Moving Floating Sensor Gantry
Design 7. Moving Actuator Gantry with Gauge Blocks
Design 8. Moving Actuator Gantry with Fixed Gauge Blocks
Design 9. Fixed Actuators with Probes and Gantry

4.1.3. Idea Selection

Sponsor Meeting
Pugh Matrices
Design Matrix
Selected Final Concept
Design Benefits
Design Drawbacks

4.2. Final Design Overview

4.2.1. Granite Subassembly
Granite Base Plate
Granite Parallel Gauge Blocks
Hardstop
Line Constraint
Gantry Linear Rails

4.2.2. X-Beam Leveling Actuator Subassembly

4.2.3. Gantry Actuator Subassembly

4.2.4. Housing and Electronics Subassembly
4.2.5. Gantry Subassembly .......................................................... 30
  Gantry Top ............................................................................ 30
  Gantry Legs ......................................................................... 30
  Bearings and Bearing Attachment Plate ................................ 31
  Heidenhain Probe and Housing ........................................... 31
  Screwdriver sub-subassembly ............................................. 31
  Rail Actuators ..................................................................... 31

4.3. Detailed Design and Assemblies ....................................... 31
  4.3.1. Analysis .................................................................. 31
    Gantry X-Axis Movement Actuator .................................... 32
    Linear Rail Actuator .......................................................... 32
    Gantry Structural Analysis ............................................... 32
    Tolerance Stackup ............................................................ 33
    X-Beam Leveling Actuator ............................................... 33
  4.3.2. Parts Selection and Design ........................................... 33
  4.3.3. Full Assembly Summary ............................................. 34
  4.3.4. XBAS Functions ......................................................... 34
    Assembling the X-Beam .................................................... 35
    Calibration ........................................................................ 36

5. Manufacturing, Assembly, Programming, and Testing ........ 36
  5.1. Manufacturing .............................................................. 36
    5.1.1. Manufacturing Plan ................................................ 36
    5.1.2. Part Job Planners ..................................................... 36
    5.1.3. Manufacturing ........................................................ 37
  5.2. Design Inspections ......................................................... 37
    5.2.1. Parts Inspection ....................................................... 37
    5.2.2. Safety .................................................................. 37
5.2.3. Power Source ............................................................................................................. 38
5.3. Assembly ....................................................................................................................... 38
  5.3.1. Mechanical Assembly .............................................................................................. 38
  5.3.2. Electrical Assembly ................................................................................................. 39
5.4. Programming ................................................................................................................ 39
  5.4.1. Programming Language and Board ........................................................................ 39
  5.4.2. Stepper Driver Codes ............................................................................................ 39
    __init__(): ....................................................................................................................... 40
    GoTo() ............................................................................................................................. 40
    GetStatus() ...................................................................................................................... 40
    GoUntil() ......................................................................................................................... 40
    ReleaseSW() .................................................................................................................... 40
    isStalled() ....................................................................................................................... 40
    HardHiZ() ................................................................................................................------- 41
5.4.3. Programming Architecture ..................................................................................... 41
    Class Quad_Encoder ....................................................................................................... 41
    task_share.py ............................................................................................................... 41
    Probe.Probe(Limit = False, UpperLimit = 0, LowerLimit = 0) ....................................... 41
    Probe.Home() ............................................................................................................... 42
    Import.Song() ............................................................................................................... 42
    Import.Calibration() ..................................................................................................... 42
    Import.BoltPattern() ...................................................................................................... 43
    Gantry.Move(Destination,Probe = False) ..................................................................... 43
    BeamActuator.Move(Destination,Probe = False) ......................................................... 44
5.5. Testing the Design ...................................................................................................... 47
  5.5.1. Survey Testing ......................................................................................................... 47
    Loading by User ............................................................................................................... 48
Learning Time .................................................................................................................. 48
Ease of Use ....................................................................................................................... 48
5.5.2. Iterative Testing ....................................................................................................... 48
Rail Straightness ............................................................................................................... 48
Rail Parallelism .................................................................................................................. 48
XBAS Speed ...................................................................................................................... 49
Rail Screw Torque .............................................................................................................. 49
System Automation Reliability .......................................................................................... 49
5.5.2. Other tests ............................................................................................................... 49
Error Conveyance ............................................................................................................. 49
6. Management Plan ......................................................................................................... 49
6.1. Administrative Roles ................................................................................................. 50
6.1.1. Communications Officer ....................................................................................... 50
6.1.2. Treasurer ................................................................................................................ 50
6.1.3. Secretary ................................................................................................................ 50
6.1.4. Manager .................................................................................................................. 50
6.2. Subsystem Design ..................................................................................................... 50
6.2.1. Structure Design Lead ........................................................................................... 50
6.2.2. Controller Software Lead ....................................................................................... 51
6.2.3. Controller Hardware Lead ....................................................................................... 51
6.2.4. Motor Implementation Lead .................................................................................... 51
6.2.5. Manufacturing Lead ............................................................................................... 51
6.2.6. CAD ....................................................................................................................... 51
6.4. Key Events and Deadlines ....................................................................................... 51
6.5. Gantt Chart ................................................................................................................ 52
7. Tear Down ..................................................................................................................... 52
7.1. What Worked ............................................................................................................. 52
7.2. What Did Not Work and What Should Change ........................................... 53
7.3. Proposed Solutions ....................................................................................... 54
8. What’s Next? ....................................................................................................... 55
  8.1. Necessary Adjustments for Full Functionality ............................................. 55
  8.2. Future Recommendations ......................................................................... 55
References ........................................................................................................ 57
Table of Appendices .......................................................................................... 58
Appendix A: Customer Requirements ............................................................... 62
Appendix B: QFD ................................................................................................. 63
Appendix C: Pugh Matrices ............................................................................... 64
Appendix D Table of Contents .......................................................................... 67
  Appendix D1: Gantry X-Axis Movement Actuator ....................................... 68
  Appendix D2: Linear Rail Actuator Calculations .......................................... 70
    D2.1. Case 1 .................................................................................................. 71
    D2.2. Case 2 .................................................................................................. 72
    D2.3. Case 3 .................................................................................................. 74
  Appendix D3: Gantry Structural Analysis ....................................................... 76
  Appendix D4: Tolerance Stack up ................................................................. 89
  Appendix D5: X-Beam Leveling Actuator ....................................................... 91
Appendix F: List of Subassembly and Parts Drawings ..................................... 93
  Appendix F1: Full Assembly .......................................................................... 94
  Appendix F2: Granite Subassembly ............................................................... 95
  Appendix F3: Housing Subassembly .............................................................. 96
  Appendix F4: Beam Actuator Subassembly .................................................. 97
  Appendix F5: Gantry Subassembly ............................................................... 98
  Appendix F6: Gantry Actuator Subassembly ............................................... 99
  Appendix F7: Screwdriver Actuator Subassembly ....................................... 100
Appendix K: Table of Inspection Sheets

Appendix K1: Bearing Attachment Plate Configuration 1

Appendix K2: Bearing Attachment Plate Configuration 2

Appendix K3: Gantry Top

Appendix K4: Hardstop

Appendix K5: Leadscrew Support Raiser

Appendix K6: Line Constraint

Appendix L: Electronics Basic Schematic

Appendix M: Bill of Materials

Appendix N: Electronics Diagram

Appendix O: Final Program Flow Chart List

Appendix O1: BeamActuator.Home()

Appendix O2: BeamActuator.Move()

Appendix O3: Gantry.Home()

Appendix O4: Gantry.Move()

Appendix O5: Import.BoltPattern()

Appendix O6: ImportCalibration

Appendix O7: Import.Song()

Appendix O8: main.Assembly_Mode()

Appendix O9: main.Calibration_Mode()

Appendix O10: main.ErrorHandler()

Appendix O11: main.Home()

Appendix O12: main.Leveling_Mode()

Appendix O13: main.Lights_Sound_Action()

Appendix O14: main.Sleep_Mode()

Appendix O15: main.TorqueDown()

Appendix O16: Probe.Home()
# List of Figures

Figure 1: The Design Process flowchart ........................................................................ 2
Figure 2: Micro-Vu Excel Machine ........................................................................... 3
Figure 3: The X-Beam in the Micro-Vu Excel machine .............................................. 3
Figure 4: FARO Laser Tracker for linear rail alignment .............................................. 5
Figure 5: Keysight 5530 Laser Calibration System setup ........................................... 5
Figure 6: The Heidenhain Metro Probe’s section view ................................................. 6
Figure 7: Assembled X-Beam with labels ................................................................. 14
Figure 8: Fixed position feedback actuator CAD model ............................................ 19
Figure 9: Moving gantry with actuators and two sensors CAD model ....................... 19
Figure 10: Moving sensor gantry with fixed actuators CAD model ......................... 20
Figure 11: Moving beam with fixed actuators and sensors CAD model ................... 20
Figure 12: Moving gantry with actuators and three sensors CAD model ............... 21
Figure 13: Moving actuator and sensor gantry CAD model ..................................... 21
Figure 14: Moving actuator and gauge blocks gantry CAD model ......................... 22
Figure 15: Moving actuator gantry with multiple fixed gauge blocks ..................... 22
Figure 16: Selected Concept CAD model .................................................................. 23
Figure 17. Selected Final Concept ............................................................................ 27
Figure 18. Assembly of XBAS .................................................................................. 39
Figure D2.1. Linear Rail Actuator Calculation Diagram 1 ....................................... 71
Figure D2.2. Linear Rail Actuator Calculation Diagram 2 ....................................... 72
Figure D2.3. Linear Rail Actuator Calculation Diagram 3 ....................................... 74
Figure D4.1. Schematic of tolerance stack up for beam to rail analysis showing beam .... 89
Figure D4.2. Schematic of tolerance stack up for beam to rail analysis showing rail .... 90
Figure D5.1. X-Beam Leveling Actuator Calculation Diagram ............................... 91
List of Tables

Table 1. Engineering Specifications derived from customer requirements. ......................... 11
Table 2. Comparison of proposed specifications with current products ............................. 17
Table 3. Design Matrix for the XBAS project ................................................................. 26
Table 4. Summary of Analysis. ......................................................................................... 32
Table A1. Customer Requirements .................................................................................. 62
Table C1: Pugh Matrix of systems for moving the gantry or carriage along the x-axis .... 64
Table C2: Pugh Matrix of systems for adjusting the rails along the y-axis ..................... 65
Table C3: Pugh Matrix of systems for measuring and aligning the rails ....................... 65
Table C4: Pugh Matrix of systems for torqueing the screws on the rail ....................... 66
Table C5: Pugh Matrix of gantry or carriage frame styles ............................................. 66
Table D1.1. Calculated Time required to torque down the screws .............................. 68
Table D1.2. Calculated time spent making the gantry stop or go .................................. 69
Table D1.3. Calculation of the velocity required to fulfill time constraint .................. 69
Table D1.4. Calculation of motor rpm to fulfill velocity requirement ......................... 69
Table D2.1. Linear Rail Actuator Constant Values ......................................................... 70
Table D2.2. Calculation of the force required actuator force output for Case 1 .......... 71
Table D2.3. Calculation of the force required actuator force output for Case 2 .......... 73
Table D2.4. Calculation of the force required actuator force output for Case 2 .......... 75
Table D4.1. Tolerance stack up results for beam to rail analysis for straightness ...... 89
Table D4.2. Tolerance stack up results for rail to rail analysis for parallelism .. 90
Table D5.1: Calculation for the stroke length needed .................................................... 92
Table D5.2. Calculation of force output necessary to level the X-Beam ..................... 92
Table G1: List of Manufactured Parts ........................................................................... 122
1. Introduction
The X-Beam Precision Rail Alignment project is a 2016-17 California Polytechnic State University of San Luis Obispo (Cal Poly) Mechanical Engineering (ME) senior project sponsored by Micro-Vu, producer of high precision measurement systems. Both Ian Davison, who represents Micro-Vu, and Professor Eileen Rossman of the Cal Poly Mechanical Engineering Department at Cal Poly, will oversee the project. The ME senior project advisors’ committee has selected three Cal Poly students (Team X) to complete this project which will be presented at Cal Poly’s Engineering Project Expo in Spring Quarter of 2017.

The goal of the project is to design, build, and test an automated machine for Micro-Vu that will align a pair of rails to the necessary precision as to reduce manual labor and human error. This paper will henceforth refer to this project as the X-Beam Alignment System or XBAS for short. This report will relay relevant background research, explicitly state our objectives, and explain how we plan to complete them.

2. Background
This section details information about Cal Poly’s ME Senior Project and our sponsor Micro-Vu, especially what their problem is and what they are expecting of us. Additionally, this section has information about existing products we found which could assist and expedite our sponsor’s current process, helping solve their problem.

2.1. Senior Project
Cal Poly’s ME Senior Project is a three-quarter long project in which a team of students design, source, manufacture, and test a prototype design in order to address their sponsor’s needs. In Figure 1 is a flowchart depicting the design process and how the project is divided up over three quarters for this Senior Project for Micro-Vu.
2.2. Sponsor Background and Needs

Micro-Vu is a company that specializes in high accuracy and precision machines for measuring the dimensions of parts, materials, and objects. Micro-Vu has a number of these machines for sale ranging from non-contact visual comparators to contact probes that perform high precision 2D and 3D measurements, namely the Micro-Vu Excel (Figure 2). Micro-Vu is an internationally recognized brand; however, their machines are highly sensitive and their assembly can affect their performance significantly. Micro-Vu has asked Cal Poly’s ME Senior Project Program to assist with this issue, specifically helping with the assembly of their X-Beam and rails subassembly, which Micro-Vu uses in many of their machines.

The X-Beam is an integral part of the Micro-Vu Excel series machines; an example of which is included in Figure 3. It is on this beam that the gantry, consisting of measuring equipment, rides on. The X-Beam is a steel tube that is machined to specific dimensions and then powder coated to accommodate a pair of hardened steel precision linear rails and a linear encoder for the x-axis motor. For the axis, please refer to Figure 3. For further information on the X-Beam and the machines that use the X-Beam, please look at Reference [1].
Due to the low production yield of precision linear rail machines, a high-skill assembly technician manually performs the alignment as these machines are ordered. During this process, Micro-Vu technicians first constrain the X-Beam with a point and a line on top of a granite micro-flat table that doubles as a datum for taking reliable measurements. Then the technician cleans the rail surfaces and installs linear rails on either side loosely with the screws inserted in their holes. Next, the technician checks the parallelism of the X-Beam itself to the micro-flat surface by running a dial indicator across the linear encoder surface and adjusting the X-Beam until either end of the beam reads 0 on the dial probe, disregarding sag in the center of the X-Beam. This measurement has an added error of 0.5μm due to the dial probe’s resolution error. After this, the technician torques the first screw on one end of the rail down so that the screw is about center to the center hole of the rail. After that, the technician proceeds to torque down every other screw along the rail, applying force to the top or bottom of the rail above the screw so that the dial probe reads
zero to the first screw. The reason why Micro-Vu has its technicians torque down every other screw is to prevent the rail warping and creating high-order waveforms. The technician repeats this process until the end of the rail and runs back along the beam, making sure the rail stays within the ±1μm range. If not, the technician will loosen bolts from the point where the rail deviates more than the tolerance all the way to the nearest end of the beam and repeats the process of torquing the screws down. The technician then performs the same alignment for the rail on the opposite side of the X-Beam. By decreasing the variation of height along the rails, he ensures the rails’ parallelism inherently.

When we visited Micro-Vu on October 22, 2016, we assembled an X-Beam with the help of onsite assembly technicians. While we took around 45 minutes to assemble the X-Beam, the technician completed the assembly in five minutes due to years of experience. This highlighted to us that while the speed of the machine will be important, the skill level involved in assembly of the beam creates the biggest difference in time.

2.3. Existing Products and Specifications

Beyond Micro-Vu, many companies employ linear rails in their Computer Numerically Controlled (CNC) systems. Due to the necessity for these systems to be aligned and the low production runs of these companies, the methods applied generally involve manual correction of the rails while measuring them. Team X has decided to compare various measurement systems with Micro-Vu’s current process detailed above and our upcoming design.

2.3.1. FARO Laser Tracker

FARO has developed a laser metrology system that utilizes a secondary laser tracking mechanism to measure the parallelism and positional tolerances of linear rail when it is set up on a machine. This system is incredibly portable because the laser scanner is self-contained and battery powered. This allows the user to transport the laser tracker unit between various locations with ease. A demonstration of the rail is on Figure 4 displayed on the next page. While this system is still relatively accurate for most CNC machines, the accuracy of 150μm is simply not accurate enough for our system to use. Still, it is the easiest to use, as it requires relatively no knowledge to run the tests. Refer to Reference [4] for detailed information pertaining to the FARO Laser Tracker.

2.3.2. Keysight Technologies Interferometer

In terms of precision measuring, Keysight Technologies has a laser-based metrology system currently on the market with impressive specifications, which makes it an ideal competitor for our upcoming design. The Keysight 5530 Dynamic Laser Calibrator is used to measure machine tool positioning accuracy for CNC lathes and mills with the use of a laser that measures a broad variety of geometric dimensions and tolerances. This includes
straightness and parallelism, which are the two tolerances that our design must measure. Although the process of collecting measurements for this device is automated, the process of realigning the part being inspected still must be done manually with human interaction. The general setup and usage of this device can be seen in Figure 5 on the next page.

Figure 4: FARO Laser Tracker for linear rail alignment. The blue cylinder is the actual laser system while the device the subject in the photo is holding is the other part of the system for measurement. This image is courtesy of FARO.

Figure 5: Keysight 5530 Laser Calibration System setup for measuring vertical straightness along the x-axis of a mill. This requires an operator to adjust the part and/or laser precisely measure the
The data sheets provided from Keysight state that their product has a straightness measuring resolution of 0.01μm for short ranges (0.1m - 3m) and 0.1μm for long ranges (1m - 30m). These are extremely accurate tolerances that our design may not be able to compete with, but our design will be much more automated in that the rail adjustment process and possibly the screw tightening process as well will be completed with no human interaction. Some other interesting key features about the Keysight 5530 Dynamic Laser Calibrator that are good to take note of are portability with a lightweight of 5.5 kg for the laser itself and laser reliability with a lifetime of 50,000 hours mean-time-between-failures (MTBF). If needed, Reference [3] contains more information about the Keysight 5530 Dynamic Laser Calibrator.

2.3.3. Heidenhain Metro
While not directly related to rail precision, the Heidenhain Metro is well known in the metrology industry as a reliable and accurate length gauge. In function, the MT60 Probe is similar to a dial indicator, but displays its values via an electrical signal to an LCD. Figure 6 below details the internal workings of the Metro length gauge which allow for such accurate measurements such as the ball-bush guide that allow for these measurements. While the gauge is small and accurate, it still requires an operator to correct the rail and move the probe to each point along the rail.

Figure 6: The Heidenhain Metro Probe’s section view. It should be noted that this system is just the sensor for measuring rails; the user is still required to correct the measurements. This image is courtesy of Heidenhain
The datasheets provided by Heidenhain state that the Metro has a positional tolerance of 0.2μm for all distances the plunger is actuated to. Some of the advantages of the Heidenhain Metro include its reliable measurements, its small footprint, and its ease of use for measuring. Further data can be found in the data sheet in Reference [4].

3. Objectives
This section details the problem presented by Micro-Vu, their wishes as well as the engineering requirements we derived from their requests, our research, and careful considerations.

3.1. Problem Statement
Technicians at Micro-Vu need a high precision way to automate the assembly and validation of their X-Beam because the current process is time consuming, manual, and inconsistent.

3.2. Customer Requirements
As this is a project for Micro-Vu, there are several requirements and requests that they asked us to complete. For a comprehensive list of Micro-Vu’s requests and wishes, please refer to Appendix A.

3.2.1. Rails Must Be Straight
The first requirement specified by the customer is that the rails must be straight to some tolerance. Specifically, when measured from the granite flat top surface datum that the rail is assembled upon, there cannot be a change in elevation at any point from one end of the rail to the other greater than the defined amount which is further discussed in Section 3.3.1.

3.2.2. Rails Must Be Parallel to Each Other
The next customer requirement is that the rails are parallel. This requirement is paramount to the customer as the rail parallelism affects the bearing’s lifetime, machine accuracy, and the time needed to assemble the camera and Z-axis assembly onto the X-Beam.

3.2.3. Automated Rail Alignment
The third requirement by Micro-Vu is that we design and prototype an automated system to align the rails. This is because currently the process is time consuming as detailed in
Section 2.1: Sponsor Background and Needs.

3.2.4 Automated Torqueing of the Screws
This was not a requirement set forward by Micro-Vu as Micro-Vu is primarily interested in the rail alignment system. That said, Micro-Vu has stated that it would appreciate it if we could add a feature to our automation of the rail alignment to tighten and torque the screws.

3.2.5. Fast
Another of Micro-Vu’s wishes is that the machine operates within reasonable limits. While they would be very happy with the XBAS performing as fast as or faster than their technicians do (five to six minutes), Micro-Vu is happy if the machine takes “10 to 15 minutes” as that would allow an operator to walk away and complete other tasks during that time.

3.2.6. Low-Skill Level
Another requirement that Micro-Vu has is to lower the skill level required for the alignment of the rails on the X-Beam. This requirement is mentioned in the previous section about the speed of the machine in that Micro-Vu would like to remove this assembly from the list of activities their technicians must complete. Lowering the skill level of aligning the rails would allow a technician to do more important tasks, leaving the operation of the XBAS to an intern for example.

3.2.7. Must Fit on a Table
While Micro-Vu mentioned that they do have room in their assembly area for the XBAS, they did say that they prefer it not take up floor space. This is because Micro-Vu has many other machines, parts, and materials to store as well as a lot of foot traffic. Therefore, they would like the completed XBAS to sit on a table and out of the way.

3.2.8. Must have a Length to Accommodate X-Beam
This requirement states that the XBAS must be able to work only for their 500mm X-Beam. While Micro-Vu did mention that they might scale up the XBAS to fit their larger X-Beams and possibly even convert the system to work with their Z-beams, they decided that those goals were outside of this project’s purview.

3.2.9. Scalability
As previously mentioned, if the XBAS performs well Micro-Vu may use the design and
extrapolate it to fit their other X-Beams. This was hinted to us and as such we have added it here as Micro-Vu’s wish for us to design the XBAS in a way that would make future attempts to scale the XBAS up easy or downright obvious.

3.2.10. Must Use Power Available at the Site
Micro-Vu says that the site they will place the XBAS in has access to both shop air and electricity. It is an implied requirement that our machine uses those resources, as it would save Micro-Vu trouble maintaining the XBAS. This is a requirement.

3.2.11. Beam Must Be Sufficiently Constrained
This requirement states that our setup must fully restrain the X-Beam during the process of aligning the rails via kinematic restraints. This is to ensure accurate and precise measurements.

3.2.12. Basic Safety
Our sponsor has stated that the machine must maintain the same level of safety as traditional manufacturing machines so that a trained technician can use the XBAS with minimal risk. To elaborate, the technician should face no direct hazards to their health or wellbeing if the technician has been informed of the XBAS’ hazards and the proper use of the machine.

3.2.13. Must Convey Error to Operator
Another wish is some sort of error signal conveyed to the operator via a visual or auditory queue when the machine encounters a fatal error that prevents it from completing the X-Beam’s assembly.

3.2.14. Reusable
This is an implied requirement that the machine must be able to work many times. We decided to quantify the prototype’s lifetime once we started working on the engineering specifications.

3.2.15. Repeatable
As the XBAS is to be an automated process, it is implied that the XBAS must be repeatable. In other words, the machine must be able to achieve the same accuracy in alignment and parallelism repeatedly.
3.2.16. Cost
The last of the customer requirements, which we received from Micro-Vu, was an estimation of the funding. Of course, it is a true requirement that we spend within the budget set forward for us by Micro-Vu.

3.3: Engineering Specifications
From the customer requirements detailed above, we determined a few engineering specifications that characterize the most important of the customer requirements. Below, we will detail each of these engineering specifications. This entails three things.

First, each specification has a target and tolerance that the machine should achieve. Second, each specification is assigned a risk rating of high, medium, or low-risk parameter depending on how important said parameter is to Micro-Vu and the design of the XBAS. Finally, we will name how we will verify that the XBAS has reached its target within tolerance which is also called compliance. Additionally, in Table 1 on the next page, we have tabulated and summarized engineering specifications and their risk levels and compliance.
Table 1. Engineering Specifications derived from customer requirements. From left to right, there is the specification number, the parameter, the requirement, the tolerance, the risk, and the compliance. Risk is denoted with H, M, or L for High, Medium, and Low respectively and represents the importance of the parameter to the design. Compliance is denoted as an A, I, T, or S that stands for analysis, inspection, testing, and similarity respectively. Compliance is how we will validate the parameters to check whether or not they reach target within tolerance.

<table>
<thead>
<tr>
<th>Spec</th>
<th>Parameter</th>
<th>Requirement or Target</th>
<th>Tolerance</th>
<th>Risk</th>
<th>Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Straightness</td>
<td>Rails must be straight</td>
<td>±1μm</td>
<td>H</td>
<td>I</td>
</tr>
<tr>
<td>2</td>
<td>Parallelism</td>
<td>Rails must be parallel to each other</td>
<td>±1μm</td>
<td>H</td>
<td>I</td>
</tr>
<tr>
<td>3</td>
<td>Automated</td>
<td>Minimal to no human interaction.</td>
<td>n/a</td>
<td>H</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>Automated Torqueing of the Screws</td>
<td>Torque the screws with no human interaction to 2Nm</td>
<td>±0.2Nm</td>
<td>L</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>Fast</td>
<td>Rails should be installed within 15 minutes</td>
<td>maximum</td>
<td>M</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>Low-Skill Level</td>
<td>10min needed to teach new technician</td>
<td>maximum</td>
<td>M</td>
<td>T/S</td>
</tr>
<tr>
<td>7</td>
<td>Must Work on Table</td>
<td>2m x 1.2m x 0.9m table with a 1.83m gap between the table and ceiling</td>
<td>±0.25m</td>
<td>M</td>
<td>I</td>
</tr>
<tr>
<td>8</td>
<td>Must Fit X-Beam</td>
<td>150mm x 895 mm, 33.5kg beam</td>
<td>minimum</td>
<td>M</td>
<td>I/A</td>
</tr>
<tr>
<td>9</td>
<td>Must Use Power at the Given Location</td>
<td>Electricity: 110V, 220V, Air: 100-115 psi</td>
<td>n/a</td>
<td>L</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>Beam Must be Constrained</td>
<td>Constrained</td>
<td>n/a</td>
<td>H</td>
<td>I</td>
</tr>
<tr>
<td>11</td>
<td>Must be Reusable</td>
<td>five years lifetime</td>
<td>Minimum one year.</td>
<td>M</td>
<td>A</td>
</tr>
<tr>
<td>12</td>
<td>Repeatable</td>
<td>99.95% reliability</td>
<td>±0.05%</td>
<td>M</td>
<td>T</td>
</tr>
<tr>
<td>13</td>
<td>Cost</td>
<td>$6000</td>
<td>±$1000</td>
<td>M</td>
<td>I</td>
</tr>
<tr>
<td>14</td>
<td>Loading</td>
<td>OSHA and NIOSH standards</td>
<td>n/a</td>
<td>M</td>
<td>I/T</td>
</tr>
</tbody>
</table>
3.3.1. Straightness
Rail straightness is a high-risk parameter. In fact, among all our specifications, it is second only to the parallelism between the two rails for Micro-Vu. As previously stated, from one end of the beam to the other, the change in elevation cannot exceed ±1μm. This will be checked via inspection of the full assembly of the X-Beam and rails after the XBAS has finished putting them together using Micro-Vu’s current setup to run dial indicators across the rail’s length to check that the rail straightness is within tolerance.

3.3.2. Parallelism
Rail parallelism is the other high-risk parameter. This matter is of huge concern for Micro-Vu; however, they do not have a way of measuring this currently. We propose to measure both rails at the same time and make sure that the measurements are within ±1μm. Rail parallelism will be checked via inspection similarly to straightness using a pair of dial indicators at the same time and make sure that the measurements do not vary from each other by more than the tolerance.

3.3.3. Automated Rail Alignment
Our third specification is that the XBAS must receive minimal to no human assistance after the operator has cleaned and placed the X-Beam, rails, and screws in their respective initial positions. This does not include instances where a critical error has occurred at which time an error signal will notify the operator asking for assistance. The automation will be checked for compliance by testing the XBAS via multiple runs. This is a high-risk specification.

3.3.3. Automated Torqueing of Screws
The manufacturer data sheet for the rail and bearings has stated that a good torque for the screws would be about 2Nm. We initially estimate that this has a tolerance of ±0.2Nm. We will check that the XBAS can torque screws to the target via testing as it goes hand in hand with testing the automation of the rail alignment. This is a low-risk specification as it is not required by Micro-Vu. Further information and a datasheet on each rail can be found in Reference [5]

3.3.4. Fast
This is a medium-risk rated specification that describes how fast the machine should operate. An experienced Micro-Vu technician can take five minutes to align rails to the X-Beam while it took inexperienced users 45 minutes to complete a fraction of what the technician was able to do. While Micro-Vu understands that we will not be able to automate this process to finish within five minutes, they do expect that the automation
process not take “a long time”. The goal is the machine to finish assembling the X-Beam in less than 15 minutes. This will be checked by testing along with both automation processes.

3.3.5. Low-Skill Level
Currently, our goal for skill level is to make the machine an easy to use automated process, which should take no more than 15 minutes to teach to a new engineering intern at Micro-Vu. This can be checked for compliance in one of two ways. First, we can test this specification by asking several Cal Poly students to participate in an experiment in which we teach them how to run the machine in a set time and ask them to run the machine. We would see how many students can run the machine with no assistance past the first lesson. The second method is to compare the XBAS to other devices in terms of complexity and user skill. This is a medium rated risk specification due to the arbitrary nature of which the compliance is validated.

3.3.6. Must Work on a Table
This specification has been rated as a medium-risk specification as we have the option of designing the XBAS to sit on the floor. However; we have chosen to design the XBAS to fit on a table at Micro-Vu, which is about 2m in length, 1.2m in width, and .9m tall with a working space about 1.8m tall between the table surface and ceiling. Another part of this specification is that the XBAS cannot exceed the weight limit of the table, the value of which has yet to be determined. The size of the XBAS will be checked by inspection of whether or not the machine can sit on the surface of a table with the given dimensions. The weight of the XBAS will be tested by analysis of the table.

3.3.7. Must Fit the X-Beam
The 500mm X-Beam actually refers to the length of the beam that the gantry with the measuring device runs. The beam itself is 895mm long, 149.15mm wide, and 99mm tall. In addition to accommodating the dimensions of the X-Beam, we also are adding an additional specification that the XBAS must be able to support the X-Beam’s mass of 33.2±0.5kg. This specification is rated as a medium-risk spec. Compliance will be checked by inspection of if the XBAS fits around the 500mm X-Beam and by analysis to check the XBAS can hold the X-Beam.

3.3.8. Must Use Power Available at the Site
Micro-Vu has access to both 110V and 220V AC voltage outlets and 100-115 psi shop air.
Our design should be able to use one of these power sources to run itself for Micro-Vu’s convenience. Validation of this low-risk specification will be analysis of the design, checking the various parts and circuits for compatibility with the chosen energy source.

3.3.9. Beam Must Be Constrained
This is a high-risk specification as the constraining of the beam has a significant effect on the alignment of the rails. For fully constraining the X-Beam, we propose using the same set-up as Micro-Vu shown on the below in Figure 7. This consists of the line and point restraint, to restrain the X-Beam, constraining the X-Beam in three directions; linear translation in the y-axis, rotation about the x-axis, and rotation about the line created by the line restraint. Additionally, we will add two more point restraints to restrain and prevent linear translation and rotation of the X-Beam in the z-axis. This would allow us to ensure that the probes, which would replace the dial indicators in Figure 7, remain on top of the rail. The set-up of the X-Beam’s constraints will be checked via inspection of the XBAS’s setup.

![Figure 7: Assembled X-Beam with labels. The X-Beam with rails and bearings affixed in its three-dimensional constrained setting used by Micro-Vu during assembly.](image)

3.3.10. Basic Safety
In terms of safety, we aim to make the XBAS no more dangerous than a 3D printer. This is because the XBAS will definitely have pinch points and moving parts. That said, the XBAS should prove to be much safer than many shop machines that Micro-Vu owns, so we
believe it will not be required to follow as many regulations as a CNC machine for example. This is a medium-risk specification and will be checked via a thorough inspection for hazards on the XBAS.

3.3.11. Lifetime
Our goal is for the XBAS to have a lifetime of five years and one-year minimum without the need to replace parts. This is a rated as a medium-risk specification and will be checked through analysis.

3.3.12. Reliability
In the spirit of Micro-Vu, we want this machine to be repeatable with a reliability of 99.95% over the course of its lifetime. This has a medium-risk rating and will be validated through testing.

3.3.13. Cost
This is a medium-risk specification as Micro-Vu has expressed that we have some play room with the budget they gave us as they prioritize functionality over cost. Currently our budget is $6000 with maybe a $1000 margin to play with.

3.3.14. Loading
The X-Beam is a heavy and large piece of steel, and this is the smallest one they have. As such, the XBAS must be easy to load for average employees. For this specification, we will be following the standards set forward by both NIOSH and OSHA. We consider this a medium-risk specification as we do not wish our machines to cause health issues and we will check this through both inspection and testing.

3.4. Quality Function Deployment
We used Quality Function Deployment (QFD) to refine our requirements. QFD is a method used to quantify customer requirements, evaluate existing products, determine engineering specifications, and assess the relative importance of each objective. The QFD process is represented graphically by a chart known as a “house of quality” which can be seen in Appendix C. Building a QFD diagram allowed us to systematically develop engineering specifications that can be measured in order to ensure that a product meets customer needs. Our QFD generated a comparison between current rail alignment systems from competitors and Micro-Vu and our rail alignment system.

3.5. Comparison of Specifications
On the next page in Table 2 we compared our proposed engineering specifications with those of the existing products discussed earlier. This will allow us to benchmark our design against
established products while developing unique solutions to satisfy our engineering specifications. By comparing our design to others with engineering specifications, we can generate a relatively accurate rating system for the relative importance of various features.
Table 2. Comparison of proposed specifications with current products

<table>
<thead>
<tr>
<th>Competitor</th>
<th>Cost</th>
<th>Accuracy (μm)</th>
<th>Speed (cm/hr)</th>
<th>Automation</th>
<th>Ease of Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>FARO Laser Tracker</td>
<td>$80,000-$120,000</td>
<td>150</td>
<td>Instantaneous</td>
<td>Requires user to correct rail straightness.</td>
<td>User places machine down on floor and runs the test.</td>
</tr>
<tr>
<td>Keysight Technologies Interferometer</td>
<td>$7,000</td>
<td>0.1</td>
<td>Instantaneous</td>
<td>Requires user to correct rail straightness.</td>
<td>User must set up system according to specified requirements.</td>
</tr>
<tr>
<td>Heidenhain Metro</td>
<td>$500</td>
<td>0.3</td>
<td>50</td>
<td>Requires user to correct rail straightness.</td>
<td>User must understand how to read signal from sensor.</td>
</tr>
<tr>
<td>Project Proposed Specs</td>
<td>$5,000-$6,000</td>
<td>2</td>
<td>600</td>
<td>Runs without user contact.</td>
<td>User has no input to machine once it runs.</td>
</tr>
</tbody>
</table>

4. Design Process
This project will be completed by following a formal design process that consists of planning, design, and production phases. Fall quarter of 2016 focused on planning and ideation. Winter quarter of 2017 consisted of detail design, design so that we can order parts.

4.1. Ideation and Selection of Design
Ideation and selection of design is a long process which took the majority of our fall quarter to complete. The process involves defining the problem, doing background research, specifying requirements, and brainstorming potential solutions. The research and work for the first three blocks can be reviewed in Sections 4.1.1. to 4.1.3. Here, we will detail the bulk of the work done this quarter which was brainstorming and choosing a design concept as our final design which is covered in 4.1.4.

4.1.1. Concept Generation
Over the course of two weeks, we had three sessions in which we generated ideas for the XBAS. This included a number of different brainstorming methods and simple prototyping.

**Brain writing**
Brain writing is a process for generating ideas in which each member writes down their
own ideas separately. After a certain amount of time, each member passes his or her ideas onto the next member, expanding on their team member’s ideas or generating new ones. This allowed us to develop a huge quantity of designs that we could evaluate later on.

**Brainstorming**

Brainstorming involved the group discussing and developing new ideas as a group as opposed to brain writing where the ideation process was mostly done in silence on paper. Brainstorming allows team members to bounce ideas off of each other quickly, helping each other think of the problem in new, innovative ways.

**SCAMPER**

SCAMPER is an acronym which stands for Substitute, Combine, Adapt, Modify, Put to another use, Eliminate, and Reverse. This process is a tool to look at a problem and/or solutions from different angles. For example, substituting one system for another, combine system functions, changing the system orientation, and so forth.

### 4.1.2. Design Concepts

Over the course of the concept generation, we came up with a lot of ideas. We condensed these ideas down to eight design concepts detailed below. The following figures have been color-coded to assist with component visibility: green refers to the position sensors used while yellow refers to the linear actuators that move the gantry in the x-direction (along the length of the beam). Red refers to the linear actuators that move the linear rail in the y-direction.
Design 1. Fixed Position Feedback Actuators

This design involves the X-Beam placed in a fixture where position feedback actuators are used every other screw. A single touch probe runs the length of the beam measuring the displacement of the linear encoder surface, allowing the actuators to correct the rail position to be parallel to the encoder surface. A SolidWorks CAD model of this prototype has been included below (Figure 8).

![Figure 8: Fixed position feedback actuator CAD model](image)

Design 2. Moving Gantry with Actuators and Two Sensors

This design features a U-shaped gantry that moves the length of the X-Beam and corrects the rail height by measuring the position with the touch probes mounted above the linear actuators. One of these touch probes would rotate up at first to measure the linear encoder strip surface, reducing the number of sensors required. The CAD work is below in Figure 9.

![Figure 9: Moving gantry with actuators and two sensors CAD model](image)
Design 3. Moving Sensor Gantry with Fixed Actuators
This design features a U-shaped gantry that moves the length of the X-Beam and corrects the rail height by measuring the position with the touch probes mounted above the linear actuators. One of these touch probes would rotate up at first to measure the linear encoder strip surface, reducing the number of sensors required. The design can be seen in Figure 10 on the next page.

Figure 10: Moving sensor gantry with fixed actuators CAD model

Design 4. Moving Beam with Fixed Actuators and Sensors
Design 4 (Figure 11) moves the beam through a fixed structure to measure the displacement of the linear rail and correct itself using fixed actuators and touch probes.

Figure 11: Moving beam with fixed actuators and sensors CAD model

Design 5. Moving Gantry with Actuators and Three Sensors
Design 5 (Figure 12) functions identically to Design 3 by having actuators on a moving
gantry except that it uses three sensors instead of two, increasing the rigidity of the system.

![Figure 12: Moving gantry with actuators and three sensors CAD model](image)

**Design 6. Moving Floating Sensor Gantry**

We designed the moving floating gantry to traverse the beam by simply correcting the rails to the linear encoder surface, eliminating the need to align the beam or use the granite table. The design’s CAD model is below in Figure 13.

![Figure 13: Moving actuator and sensor gantry CAD model](image)
Design 7. Moving Actuator Gantry with gauge Blocks

Two of our designs (Figures 7 and 8) use gauge blocks instead of touch probes to achieve the positional tolerance required for this project. Design 7 has gauge blocks attached to the moving actuator gantry, allowing the actuators to press the linear rail to be parallel with the micro-flat table. The design is below in Figure 14.

![Figure 14: Moving actuator and gauge blocks gantry CAD model](image)

Design 8. Moving Actuator Gantry with Fixed Gauge Blocks

Design 8 uses gauge blocks positioned under various points of the linear rail to ensure the rail is straight when the moving actuator gantry pushes the linear rail down onto the gauge blocks. This forces the linear rail to be parallel with the micro-flat table. The design’s CAD model is below in Figure 15.

![Figure 15: Moving actuator gantry with multiple fixed gauge blocks](image)
Design 9. Fixed Actuators with Probes and Gantry

Design 9 features two fixed actuators at the ends of the rails. These actuators adjust the rails using a position feedback control system using two probes which move along the rails via a gantry.

Figure 16: Selected Concept CAD model. This is the fixed end, fixed Actuator, U-gantry design constructed and rendered in SolidWorks.

4.1.3. Idea Selection

This section talks about how we selected our ideas. This includes the meeting we had with our sponsor, the QFD, Pugh matrices, and design matrices.

Sponsor Meeting

Our sponsor proposed meeting during the ideation phase of our project once we had narrowed our ideas to five main designs so that we could discuss alternate ideas and comments to further our idea selection process. This was an informal video conference where we presented CAD models of our main designs and explain their basic functions and setups with our sponsor and one of his co-workers. We obtained valuable feedback from our sponsor regarding what they speculated to be the strengths and weaknesses of each design. This not only narrowed down the list of ideas that we had, it also brought up new ideas that our sponsor suggested we consider, such as using a single long gauge block to assist in precisely aligning the rails similar to design 8 in section 4.1.2. This meeting with our sponsor allowed us to continue with our ideation process as we grew closer to evaluating our designs in greater detail.
Pugh Matrices

Pugh Matrices are invaluable design tools that allowed us to evaluate the various functions of different designs with reference to a datum. We created Pugh Matrices for each function of our design and compared various ways to accomplish these functions to the current setup and method that Micro-Vu uses to align linear rails. Our Pugh Matrices can be viewed in Appendix C where we developed a Pugh Matrix for each function. The functions that we examined include x-direction movement, y-direction movement, measurement, screwdriver, and gantry style. Each function had a unique set of concept options that were analyzed for different criteria. Criteria that were evaluated for each function’s concept options include rigidity, repeatability, speed, scalability, projected cost, complexity, ease of maintenance, and a few other criteria since it varied for each function. Each function had several concept options that were given either a “+”, “−”, or “S” for each criterion. The “+” means that the concept option performed better than the datum for that criteria, the “−” means that the concept option performed worse than the datum for that criteria, and the “S” means that the concept option performed roughly the same as the datum for that criteria. To summarize our results for each function, it appears that using ball and leadscrews will be the best option for moving the gantry in both the x and y directions. For obtaining precise measurements, it seems that both gauge blocks and touch probes can be viable options. Either the screwdriver will be an automated torque feedback controlled screwdriver or a DC motor with torque screw since they both performed well in the matrix. We also noted the gantry style would have to be a U shape going over the x-beam. The results from these Pugh Matrices were then used to finalize our designs even further so that we could then generate complete system designs to analyze.

Design Matrix

After our Pugh matrices, we developed a design matrix to compare all our system designs with each design specification. This allowed us to evaluate each design in complete detail so that we could narrow it all down to one final design. The design matrix is on the next page in Table 3.

The first column on the left is the list of design specifications that we rated the designs against when scoring them. The second category to the right of the design specifications is “Weight” in which we scored each specification on a scale from 1 to 5, averaging each person’s input (left column) and normalized that value (right column).

Then we took an average of each person’s rating from 1 to 5 on each design for each specification where 1 is the worst and 5 is the best. After that, we multiplied the score by the normalized weight and summed each design’s total score displayed on the bottom row of the matrix.
There were many considerations; however, these are the highlights. Systems that include a moving gauge block or actuator introduce another degree of freedom and source of error. As such, these designs scored badly on measurement validity.

Systems using gauge blocks scored badly, especially in the areas regarding maintenance. This was because we found the idea of having to validate the gauge blocks themselves at regular intervals to be undesirable. This is especially true for designs that have multiple gauge blocks on either side of the beam as this would require a large amount of work to validate all the blocks, a task with exponentially scaling difficulty as the X-Beams increase in length.

Finally, systems that use arrays of measurement systems scored low with regards to cost. As previously stated, multiple gauge blocks under the rails on either side of the X-Beam would cost a lot in terms of time, but also money as they would have to be replaced on top of the initial cost of buying ten or more gauge blocks of the necessary accuracy. This is why the “multiple gauge blocks” design did so poorly in the cost category.

Similarly, the position feedback actuator array design would cost a lot in the beginning to buy all the actuators and, while they won’t need to be replaced or checked as often, they will take more time to program. That is why this design scored so badly in both cost and complexity of assembly.

In conclusion, the design with the highest score was the fixed end, fixed actuator, U-gantry with a score of 4.271. The fixed end, moving actuator, U-gantry was close by achieving the second highest score with 4.045. The other three system designs scored poorly with scores below 4.
Table 3. Design Matrix for the XBAS project

<table>
<thead>
<tr>
<th>Design Specifications</th>
<th>Weight</th>
<th>Fixed end, Fixed Actuator, U-gantry</th>
<th>Fixed end, moving Actuator, U-gantry</th>
<th>Multiple position feedback Actuator array</th>
<th>Singular gauge Block and Actuator moving on a gantry</th>
<th>Multiple gauge Blocks with moving Actuator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure Straightness</td>
<td>5.000</td>
<td>5.0</td>
<td>4.3</td>
<td>4.8</td>
<td>3.7</td>
<td>4.0</td>
</tr>
<tr>
<td>Measure Parallelism</td>
<td>5.000</td>
<td>5.0</td>
<td>4.3</td>
<td>4.7</td>
<td>3.7</td>
<td>3.3</td>
</tr>
<tr>
<td>Speed of Task Completion</td>
<td>2.667</td>
<td>3.3</td>
<td>2.8</td>
<td>3.8</td>
<td>3.5</td>
<td>4.0</td>
</tr>
<tr>
<td>Properly Constrains Beam for valid measurements</td>
<td>4.000</td>
<td>5.0</td>
<td>4.3</td>
<td>5.0</td>
<td>3.8</td>
<td>3.7</td>
</tr>
<tr>
<td>Scalability for beams of different lengths</td>
<td>1.667</td>
<td>3.3</td>
<td>4.5</td>
<td>1.5</td>
<td>4.0</td>
<td>2.3</td>
</tr>
<tr>
<td>Ease of Loading</td>
<td>3.333</td>
<td>4.0</td>
<td>4.3</td>
<td>3.3</td>
<td>3.3</td>
<td>3.0</td>
</tr>
<tr>
<td>Projected Cost estimated by number of actuators, probes, and motors needed</td>
<td>1.333</td>
<td>4.3</td>
<td>4.3</td>
<td>1.7</td>
<td>4.3</td>
<td>3.2</td>
</tr>
<tr>
<td>Complexity of Assembly</td>
<td>4.000</td>
<td>3.0</td>
<td>3.8</td>
<td>2.0</td>
<td>4.0</td>
<td>3.7</td>
</tr>
<tr>
<td>Ease of Maintenance</td>
<td>2.333</td>
<td>4.2</td>
<td>3.2</td>
<td>4.0</td>
<td>2.2</td>
<td>1.0</td>
</tr>
<tr>
<td>Sum</td>
<td>1.000</td>
<td>4.271</td>
<td>4.045</td>
<td>3.780</td>
<td>3.612</td>
<td>3.311</td>
</tr>
</tbody>
</table>
Selected Final Concept

After much deliberation, we decided that, despite our design matrix evaluation, the best design would be to use a single gauge block for each rail and have a moving gantry to press the rails down on the gauge block. This concept can be seen below in Figure 17. We chose this design due to the fact we believed our previous analysis did not reflect the simplicity and ease of execution for this design. Additionally, several assumptions we had made, such as the gauge block lifetime, were wrong and corrected during meetings with our sponsor and other consultations. As such, we chose to use a design that used gauge blocks under the rails for aligning the rails.

Figure 17. Selected Final Concept. This design uses fixed gauge blocks that the rails are pressed onto to achieve tight enough tolerances.
Design Benefits
The number one benefit of our chosen design is simplicity. This system is simpler than the other system in terms of assembly, programming, and it has lower in costs because it has less expensive parts. For example, our selected design has one probe whereas most others have two to three. These probes cost over $3000 each, so reducing the number used went a long way to reducing costs.

Additionally, less complex and sensitive parts such as the probes means less chance of failure.

Another benefit of this design makes it that it is easy to confirm the rails are both straight and parallel to each other due to the fact that the rails are parallel by virtue of the two gauge block top surfaces are manufactured to be parallel to each other. This means that to straighten the rails, we only need press them down onto the gauge block surfaces. This in turn leads to another benefit which is ease of programming since we don’t need to add a micro controller to check the position of the rail to ensure straightness.

Design Drawbacks
The first major drawback of the design is calibration of various parts. The gantry rails must be measured so any variation in the y direction can be calibrated out of the X-Beam’s leveling. We plan to design the machine to have two set ups, one for calibrating, and the other for assembling the X-Beam.

Another part that needs to be calibrated is the gauge blocks which must be checked and changed out as they degrade. This is not a huge issue though as the granite gauge blocks are checked monthly.

4.2. Final Design Overview
The XBAS is broken down into five subassemblies. In this section, we will talk about each subassembly, sub-subassembly, and part of interest. For the drawing depicting the full XBAS assembly drawing as well as drawing for all mentioned parts and assemblies, please refer to Appendix F: Table of Subassemblies and Parts Drawings. The XBAS loaded with the X-Beam can also be viewed in Figure 17: Selected Final Concept.

4.2.1. Granite Subassembly
The purpose of the granite subassembly is to act as the base and datum for measurements for the XBAS. This subassembly consists of the XBAS’s base which is comprised of a granite block or base plate, two granite parallel gage blocks, a hardstop, a line constraint, two linear rails, and around 60 bolts.
Granite Base Plate
The granite base plate is a large 1050x400x100 mm block of granite which serves as the base of the XBAS and the datum for which all measurements are taken; ensured to be within a micrometer by a flatness tolerance of 0.005 micrometers. The granite is GRANITE_JINAN_BLACK, GRADE_0_JB/T7975-1999.

Granite Parallel Gauge Blocks
The granite parallel gauge blocks are made to have parallel top and bottom as well as parallel to each other with a surface finish of 0.001 micrometer. This allows us to create a verifiable datum on the top of both granite parallels based on the granite base plate, allowing for accurate results when the rail is straightened against the granite parallels. The granite is GRANITE_JINAN_BLACK, GRADE_0_JB/T7975-1999.

Hardstop
The hardstop is a part we will be manufacturing ourselves from stock metal rods. Its purpose is to support the X-Beam after loading and protect the X-Beam leveling actuator from unnecessary until the XBAS begins to run or impact from the X-Beam as the user loads the XBAS. This will improve the lifetime of the X-Beam leveling actuator. The plan for manufacturing the hardstop can be seen in section 5.1. The hardstop will be made of Aluminum 6061.

Line Constraint
The line constraint is a small half cylinder piece meant to replicate a line in supporting the X-Beam. The line constraint, in conjunction with the point constraint provided by the X-Beam actuator, kinematically and fully constrains the X-Beam, ensuring accurate measurements. The line constraint is a part we plan to make; the description of our manufacturing plan can be seen in section 5.1. The line constraint will be made of Aluminum 6061.

Gantry Linear Rails
The linear rails in the granite subassembly are not to be confused with the rails on the X-Beam and as such will be referred to as the gantry linear rails. The gantry’s linear rails were chosen for their high reliability, low friction, and Micro-Vu’s familiarity with IKO rails. These rails we be made of carbon steel.
4.2.2. X-Beam Leveling Actuator Subassembly
The X-Beam leveling actuator subassembly consists of stock parts from Misumi and it has two purposes. The first is to act as a point constraint and constrain the X-Beam in conjecture with the line constraint mentioned in section 4.2.1. This subassembly’s second purpose is to lift and lower the end of the X-Beam and level the X-Beam in conjunction with the Heidenhain MT-60M Probe on the Gantry Subassembly (see section 4.2.5). This subassembly consists of a non-captive rail and NEMA14 motor supported by an L-Bracket, double bushing bearing, and precision linear shaft.

4.2.3. Gantry Actuator Subassembly
The gantry actuator subassembly was designed to move the gantry back and forth along the X-Beam. It consists of all stock parts from Misumi, consisting of a NEMA23 Motor, flexible coupling, double stepped linear screw, linear rail end support, anti-backlash nut, and a part we machine called the Leadscrew raiser. Calculations and analysis were discussed in section 4.3.2 while the manufacturing plan for the lead screw raiser can be found in section 5.1.

4.2.4. Housing and Electronics Subassembly
The housing’s purpose is to provide an area for the gantry to be stored for protection as well as provide area to mount electronics such as the processor, power supply, buttons, and indicators.

4.2.5. Gantry Subassembly
The gantry subassembly carries a number of parts in order to fulfill its purpose which is measuring and straightening the X-Beam and rails. This subassembly consists of a number of parts, starting with a three-part frame, bearings and their attachments, MT-60 Heidenhain probe and housing, screwdriver subassembly, and rail actuators.

**Gantry Top**
The top of the gantry is where the actuators and probe will be fixed to. Calculations for the deflections, stresses, and fatigue lifetimes are all in the relevant section of section 4.3.1.

**Gantry Legs**
The gantry legs connect the gantry top part to the bearings, bearing attachment plates, and gantry rails mentioned in the next paragraph. One of the gantry legs is also connected to the gantry actuator’s anti-backlash nut.
Bearings and Bearing Attachment Plate
The bearings are matched to the gantry’s linear rails, allowing the gantry subassembly low friction guided movement. The bearing attachment plates will be manufactured by us and are used to connect the bearings to the bottom of the gantry legs.

Heidenhain Probe and Housing
The Heidenhain probe is a MT-60M Metro probe used in conjunction with the X-Beam leveling actuator to level the X-Beam to the granite base plate. It is protected by bent sheet metal due to its importance.

Screwdriver sub-subassembly
The screwdriver sub-subassembly consists of a number of stock parts including gears, a DC motor, solenoids, and a friction slip plate torque limiter. This assembly only has two parts we manufacture, the solenoid pins and the gearbox housing. Details for the manufacturing can be found in section 5.1.

Rail Actuators
These are the actuators used to press the linear rails of the X-Beam down into the granite parallels, straightening them. Calculations used to spec them can be found in section 4.3.1.

4.3. Detailed Design and Assemblies
After the selection of our design, we went onto doing calculations, analysis, and sourcing parts from various suppliers. This section summarizes the mentioned subjects.

4.3.1. Analysis
Below are the results of the different major analysis we did. These calculations were for checking the design’s feasibility as well as for sizing parts which is discussed further in the section 4.3.2. Additionally, below in Table 4 is a summary of our analysis.
Table 4. Summary of Analysis. This table summarizes the results of the analysis listed on the left column and lists the Appendix where the calculations can be found.

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Result</th>
<th>Appendix #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gantry Actuator Motor Speed</td>
<td>430 rpm</td>
<td>D1</td>
</tr>
<tr>
<td>Linear Rail Straightening Actuator Force Output</td>
<td>417 N</td>
<td>D2</td>
</tr>
<tr>
<td>Gantry Deflection Analysis</td>
<td>0.88 μm</td>
<td>D3</td>
</tr>
<tr>
<td>Gantry Plastic Deformation</td>
<td>None</td>
<td>D3</td>
</tr>
<tr>
<td>Gantry Fatigue Analysis</td>
<td>Infinite Life</td>
<td>D3</td>
</tr>
<tr>
<td>Tolerance Stack up</td>
<td>0.1mm</td>
<td>D4</td>
</tr>
<tr>
<td>X-Beam Leveling Actuator Force Output</td>
<td>167 N</td>
<td>D5</td>
</tr>
<tr>
<td>X-Beam Leveling Actuator Stroke Length</td>
<td>5 mm</td>
<td>D5</td>
</tr>
<tr>
<td>X-Beam Leveling Actuator Resolution</td>
<td>0.1 μm</td>
<td>D5</td>
</tr>
</tbody>
</table>

**Gantry X-Axis Movement Actuator**

The actuator for driving the motor as well as the lead screw used was calculated using the time design specification of completing the X-Beam assembly in 10 minutes as the defining factor. A sample calculation is shown in Appendix D1. In summary, for a lead screw with a 2mm step and a NEMA23 triple stack motor with a 1.8 degree step, we need the motor to run at 430 rpm to achieve the speed necessary for the XBAS to complete the X-Beam assembly in 10 min.

**Linear Rail Actuator**

In order to size the actuators for pressing the rails down onto the granite parallel gauge blocks, we needed to know what the force output was necessary for the worst-case scenario to ensure we could pick a motor that would always straighten the rail against the gauge blocks. We modeled the rail as a cantilever beam in three different configurations. The conclusion was that we needed a motor that could output 417 N of force. The calculations are displayed in Appendix D2.

**Gantry Structural Analysis**

The gantry, which consists of three bolted parts, was modeled as a single uniform piece of 1018 cold drawn steel in order to analyze the structural integrity under various load cases. The applied load we considered was the weight of all the components attached to the gantry, the weight of the steel gantry frame, and the maximum forces exerted from both of the linear rail actuators. A quick deflection calculation was performed which showed that the maximum deflection or the gantry is 0.88 μm.
Further calculations show that the maximum stress the gantry will experience is 0.54 MPa which is significantly lower than the yield strength of the steel being 370 MPa. From these results, we conclude that the gantry should not undergo any plastic deformation.

Fatigue calculations were also performed according the ASME-elliptic formula with infinite life. We calculated the endurance limit of the steel to be 340 MPa with an alternating strength of 251 MPa. The maximum stress that the gantry will experience does not exceed the endurance limit of the steel which supports our earlier conclusions that the gantry will not fail due to fatigue. This concludes the structural analysis for the gantry. The calculations are displayed in Appendix D3.

**Tolerance Stackup**
In order to satisfy the 1 μm straightness and parallelism requirements for the rails, a tolerance stackup analysis was performed to determine the allowable tolerances for the granite parallel and line constraint. The analysis was performed by identifying a loop from beam to rail to check the straightness requirement and rail to rail to check the parallelism requirement. The results show that the granite parallel can have a maximum height linear tolerance of 0.1 mm and the line constraint can have a height linear tolerance of 0.1 mm as well. The calculations are displayed in Appendix D4.

**X-Beam Leveling Actuator**
The X-Beam actuator has to level the X-Beam to within a micrometer with the help of a probe which would measure the X-Beam’s height and check for straightness by comparing rail height at two different points. The actuator had a couple constraints that needed to be calculated before being chosen. These were calculated in Appendix D5. To summarize, the actuator for leveling the X-Beam to within a tenth of a micrometer must have a force output of 167N, a stroke length of 5mm, and a minimum resolution of 0.1 micrometer.

**4.3.2. Parts Selection and Design**
The XBAS is designed with both stock parts and parts we manufacture. We tried to minimize the number of parts we had to manufacture due to the precision needed for our project; however, there were some we had to make ourselves. The parts we must manufacture ourselves are:

1. The three parts of the gantry’s frame
2. The hardstop
3. The line constraint
4. The bearing attachment plates
5. Lead screw support raiser
6. Housing and Electronics housing
7. Heidenhain probe housing
These are further detailed in section 4.2 and 5.1 where we discuss these parts purposes and manufacturing plan respectively.

When it came to selecting parts for the design, we wanted to keep a few points in mind.

1. Use wholesalers that Micro-Vu was familiar with. There are two reasons for this.
   a. Some parts such as the IKO rails we are using are kept en masse by Micro-Vu, so if it needed to be replaced, Micro-Vu already has a warehouse of related parts.
   b. These wholesalers already have a working relationship with Micro-Vu which makes it convenient for Micro-Vu in a number of ways.
2. Sourcing parts from the same company for easier part ordering.
3. Wholesalers had to have abundant supplies of the stock we were ordering as to limit lead times for ordering if parts needed to be replaced by Micro-Vu in the future.
4. Parts had to fulfill design specifications.
5. Wholesalers had to be reputable and have good documentation on where the parts came from.

4.3.3. Full Assembly Summary
In summary, our current design of the XBAS costs $9,648.11 and consists of 297 parts (66 unique parts) sourced from 11 sources with a vast majority of parts coming from Misumi USA. The XBAS and how it functions is described in the next section. It should be noted that this is the ideal cost estimate based on assembling the entire XBAS with no need to reorder damaged or wrong parts.

4.3.4. XBAS Functions
Below are the functions that the XBAS’ current design can currently perform and how the XBAS performs those functions. Remember, all parts and assemblies can be viewed in Appendix F.
Assembling the X-Beam

Once the user has set up the XBAS as shown in Figure 17, the user can turn on and activate the assembly mode of the XBAS. This will begin the first “mode” of the Assembly function of the XBAS.

The first mode of the Assembly function is the leveling of the X-Beam. The gantry first moves out to measure the height of the X-Beam at the linear encoder strip area nearest to the XBAS gantry housing, further referred to as the near side. Then it moves to the opposite end (the far side) of the X-Beam and measures the height of the X-Beam at the linear encoder strip over the line constraint. From this, it calculates the X-Beam’s level to the granite base plate and actuates the X-Beam leveling actuator to correct the X-Beam’s level. After that, it measures at the far side of the X-Beam, runs back to the near side and measures the X-Beam’s height again, confirming whether or not the X-Beam is leveled to the micrometer. If it is not, it will repeat the entire process until the X-Beam has been leveled to tolerance. If the beam has been leveled, the X-Beam will switch to the next mode.

The second mode of the XBAS Assembly functions is straightening the rails on the X-Beam and torqueing them down. This is achieved by the gantry moving to the first screw on the X-Beam’s long rail, pressing the rail down with the actuators on the gantry, and torqueing the bolt down with the screwdriver subassembly. It then moves two bolts over and repeats the process to torque down the rail. This is done until the XBAS gantry reaches the X-Beam’s short rail’s first bolt. It straightens and torques down that side before moving over one bolt and torqueing the longer rail’s bolt. The gantry then proceeds to alternate between the rails, torqueing one on each side until it reaches the third from last bolt on the shorter rail. The gantry at this point straightens and torques both rails at this bolt, moves two bolts over, and again aligns and torques both rails. The gantry then moves over one bolt on the longer rail and torques that bolt down before moving to the last bolt on the longer rail and bolting that down as well. After that, it runs back along the X-Beam, torqueing all remaining bolts down.

Once it finishes, it returns to the housing where the machine will signal the operator that the XBAS has finished assembling the X-Beam.
Calibration
The XBAS' calibration mode is designed to calibrate variations of the rail's change in height in the vertical direction out of the measurements taken by the gantry’s probe at the two ends of the X-Beam in the initial part of the Assembly function. To do this, the user mounts the probe in one of two calibration spots and runs the calibration mode. The XBAS will move the gantry to the near and far side of the X-Beam to take measurements of height based on the granite parallel surfaces. The gantry then runs back to the other side at which time the operator mounts the probe on the second position and runs the calibration again. These two values are stored and used for the probe calibration until someone reruns the calibration function, overwriting the data stored.

5. Manufacturing, Assembly, Programming, and Testing
Spring quarter of 2017 was comprised of everything after the brainstorming block on Figure 1. During spring quarter, we assembled the machine, assembled the electronics, and designed a program to run the machine. After all of that was completed, we debugged the program, tested the XBAS, and finally presented the XBAS during the spring 2017 Senior Project expo. We will summarize our manufacturing, programming, and testing process.

5.1. Manufacturing
In this section, we talk about the manufacturing that was completed for the XBAS.

5.1.1. Manufacturing Plan
The table displayed in Appendix G was developed to list all the parts that were manufactured along with all other additional manufacturing processes that were needed. The table shows all eighteen parts that were made along with each part’s corresponding stock material, material cost, part size, and source of who made the part. The total cost for all stock material is $1,736.67. We verified with one of the Supervising Directors of the ME machine shops that the shops do have all the appropriate tooling and equipment needed to make our parts in metric units. Most of the parts were completed by us in the ME machine shops on the Cal Poly campus while other parts such as the gantry top and legs were made by Micro-Vu and the granite parts were made by a granite supplier. The Gantt Chart in Appendix H shows the time estimates for each phase along with all major manufacturing tasks that were accomplished for this project.

5.1.2. Part Job Planners
Part job planners were developed for each part that was manufactured using the manual
mill to clearly specify what operations must be completed along with all necessary tooling selections, setup, equipment, and operation selections. Each planner shows all the necessary operations in chronological order from setup to polishing along with estimated times and tools needed for each operation so that any user can easily follow them. The user making these parts would have a copy of both the part drawing and part job planner to facilitate the manufacturing process of all parts. All part job planners can be seen in Appendix H.

5.1.3. Manufacturing
All manufacturing operations for the parts that we were responsible for were completed in the ME machine shops on the Cal Poly campus with the use of a manual mill, drill press, and manual sheet metal bender. The bearing attachment plates, line constraint, hard stop, and lead screw support raiser were all made on a manual mill from the stock material acquired using the part job planners developed to manufacture each part accordingly. All sheet metal parts were bent from stock sheet metal using the manual sheet metal bender to achieve all the necessary dimensions and features as seen in the corresponding part drawings. The additional manufacturing processes that were needed were drilling holes into the gear box covers and applying taps where needed.

5.2. Design Inspections
In this section, we talk about the inspections we performed on the XBAS over the course of the machine’s assembly and testing.

5.2.1. Parts Inspection
The inspection of all the manufactured parts we made consisted of checking the hole positions and critical dimensions using the optical comparator, a caliper, drop height indicator, and test type indicator. If any hole or dimension was not to tolerance, then that part would have failed inspection and would need to be redone. Parts inspections have been detailed in Appendix K. Please refer to Appendix K: Table of Inspection Sheets to find individual part’s inspection plans. All manufactured parts passed inspection and were ready for assembly in early Winter quarter.

5.2.2. Safety
This inspection was completed on 05/04/17 by Professor Rossman. After all safety considerations were considered and discussed, it was determined that our design was safe and could operate at the senior expo if a few sheet metal edges were sanded down. Failure would be if the ME department declared the XBAS as unsafe and hazardous in which case we would need to modify our design to remove the safety issue.
5.2.3. Power Source
This inspection is very simple and consisted of checking whether or not there was a viable wall socket for the XBAS to plug into. If there was no suitable wall socket, then the XBAS would have failed the inspection.

5.3. Assembly

5.3.1. Mechanical Assembly
While various subassemblies such as the gantry and actuator subassemblies were started to be put together as soon as parts arrived in early Winter quarter, the complete assembly of the XBAS was officially started once the granite parts were received. The granite plate was carefully loaded onto the table with the use of a forklift so that all the components of the XBAS could then be assembled on top of it. The granite plate and granite parallel constraints were properly cleaned following basic maintenance procedures for granite to ensure that the assembly of the XBAS would be done correctly. The linear rails were then bolted onto the granite plate along with the various constraints, subassemblies, and housing components of the XBAS. Pictures of the XBAS during assembly can be seen in Figure 18. With most of the mechanical components assembled onto the granite plate, the electrical components could then be assembled.
Figure 18. Assembly of XBAS. The various subassemblies and constraints can be seen being assembled onto the granite plate.

5.3.2. Electrical Assembly
Electronics assembly began late Winter quarter with the assembly of the microcontroller stack and the wiring of the power supply. Further assembly was completed as parts came in, however nothing could be fully assembled until the sheet metal housing was completed. As such, electronics were put on a hold until late April at which point the layout for components in the housing were finalized and assembly began. We ran into further issues in late May where a stepper driver board was fried due to a grounding issue. This was easily rectified but set us back a week because it took a while to debug the issue. Once electrical assembly was completed, we were ready to test our programming.

5.4. Programming
In this section, this report will address the major design decisions we made concerning the programming for the XBAS. This includes the our choice in programming language, what we did for connecting to the hardware, and the overall architecture of the code.

5.4.1. Programming Language and Board
For this senior project, we decided to use the micropython computer language for four major reasons.

1. Robert already had python programming experience.
2. Whittaker and Robert were both in a Mechatronics class which was taught in both python and micropython. So we were both very familiar with these languages applied to robotics.
3. The Mechatronics class had a good set up of a pyboard (microprocessor designed for micropython), motor shield, and a custom foot attachment for the board which we wanted to use for its convenience and it would save lots of time we would otherwise have to spend picking a board and customizing it to our project.
4. Python is an object orientated programming language which made it much easier to program the driver codes for the four motors we are using.

5.4.2. Stepper Driver Codes
The stepper driver code was written as a joint effort from Whittaker and Sam Artho-Bentz, based on code provided to them by John Ridgely, John Barry, and Anthony Lombardi. Dr. Ridgely provided the base driver code which formed the beginnings of our SPI communication protocol. Because we communicated to the stepper drivers over SPI, we needed code that would take care of processing commands into bytes and transmitting them between the board and the drivers. After the SPI communication protocol was correctly configured, we implemented John Barry and Anthony Lombardi’s Command dictionaries to read and write to the drivers. These dictionaries processed incoming data
and contained the address of registers to write to when reading and writing from the stepper boards, respectively. Because the stepper driver boards were daisy chained together, communication was completed through a single SPI port with different chip select pins.

Dual6470 was written as a class and is detailed briefly below. The main commands used were GoTo(), GetStatus(), GoUntil(), ReleaseSW(), isStalled(), and HardHiZ().

__init__():
__init__() is called every time a new object is created using the Dual6470 class. This sets up the SPI lines for whichever pins are selected as the SCK and nCS pins as well as the SPI channel selected. It then resets the board to check for communication and writes stepper motor parametrization values to the drivers.

GoTo():
The GoTo() command initiates a move by a number of steps specified by the user for a specified motor (1 or 2). This command is executed at full speed and is performed using absolute positioning. Motor microsteps at specified microstepping until full speed is reached, at which point driver switches to full stepping.

GetStatus():
This command retrieves values from the status register of the microcontroller and parses them to determine the status of the stepper driver. It then prints the status register in an easily readable format. This command also resets the status register for both drivers on a single board, a useful command to clear any error flags during startup.

GoUntil():
This command executes a move at a specified speed in steps/sec for the motor given. When an external switch is closed for this board, the absolute position register is zeroed and a hard stop is initiated, effectively homing the actuator.

ReleaseSW():
ReleaseSW() moves at minimum speed until the external switch is no longer pressed, ensuring the stepper actuator is at the same home position every time.

isStalled():
This command checks if the stall flag has been raised and returns a true/false depending on if the flag has been raised. It performs this check by reading the status register and checking a specific bit through bit shifting.
HardHiZ()

HardHiZ() de-energizes the coils in the stepper motor specified, effectively turning that stepper motor off.

5.4.3. Programming Architecture

Immediately after we bought parts, we proceeded to programming. This was done through a series of state diagrams which transitioned into a very extensive series of flow charts. Then we would write the code, compare them, and make revisions. This process was repeated several times in addition to numerous checks for logic and syntax errors during the time we were assembling the XBAS. For the sake of efficiency, we have only included the final version of the flowcharts and script file in Appendix O and P respectively.

In this section, we will talk about significant functions and classes; their purpose, inputs, outputs, and the important design decisions of that function that one should be aware of. The stepper driver codes are addressed separately as they were both written separately and by different people as the following program.

**Class Quad_Encoder**

This class is used by the probe to operate the quadrature encoder. The class requires two pins for channel A and B of the encoder as well as a Timer channel for those two pins. This class has two class members, read and zero.

Class member read updates and returns the current encoder value; however, it must be run often so that the encoder does not roll over more than twice.

Class member zero zeroes out the encoder’s position value as well as other values used in calculating position.

**task_share.py**

This file and all of its class objects were written by Doctor John Ridgley, a professor at Cal Poly. This function creates buffers which can be used like global variables; however, these buffers also come with multithreading in case of interrupts which is why we chose to use this if we needed to handle interrupts. While this turned out to be unnecessary, we left the buffers as it would take time to remove and replace the buffers as well as check that the other functions are not affected by this change.

**Probe.Probe(Limit = False, UpperLimit = 0, LowerLimit = 0)**

Probe() is a function of the Probe.py python module. It has three inputs and two outputs.
The inputs are Limit, UpperLimit, and LowerLimit. If Limit = True, then the function Probe will compare its readings to the given UpperLimit and LowerLimit. If Limit = False, then it ignores this check.

The outputs are either a probe reading or “Error Occurred” if the probe read a value outside of the limits given during the check.

Probe() has one design consideration of note, the ability to check the readings and throw an error if the reading is outside of the limits. This was implemented to prevent incorrect readings from situations such as the user stopping the probe prematurely (with their hand or another object) or if the X-Beam the probe is measuring is just not there.

Probe.Home()
This Home() is a function of the Probe.py python module. It has no inputs or outputs. This function homes the Probe.

Notable design choice, due to lack of time at the end of senior project, we implemented a time out for the probe’s homing. This is because the Heidenhain probe’s reference tick was not being read, so we created this temporary solution.

Import.Song()
Song() is a function of the Import.py python module. It takes no inputs and outputs either “No Error” or “Error Occurred”.

It reads a piano switchboard and places an ID code (an integer) corresponding to the combination of on and off switches on the board into a buffer object called XBeam for use by other Import functions.

Design consideration of note, this function was created as is to accommodate for more X-Beam lengths in case Micro-Vu does size up this design.

Import.Calibration()
Calibration() is a function of the Import.py python module. It takes no inputs and outputs an “Error Occurred” in the case of a list of values which was imported from the Calibration file corresponding to the ID in the XBeam buffer provided by Import.Song().

This function uses the ID in XBeam to figure out the name of the Calibration csv file it needs to access and checks if that file is present. If the file is present, it reads the values, processes them into a list, and checks the values to make sure they are valid. If they are invalid, the function reports which value is wrong and where it is. This function also writes
to an “Error Report.txt” file the information about which data points are invalid.

Design consideration of note, as with Import.Song(), this function is designed to work with Import.Song() to be able to accommodate many different X-Beams in case Micro-Vu scales this project up.

**Import.BoltPattern()**

BoltPattern() is a function of the Import.py python module. It takes no inputs and outputs an “Error Occurred” in the case of a list of values which was imported from the BoltPattern file corresponding to the ID in the XBeam buffer provided by Import.Song().

This function uses the ID in XBeam to figure out the name of the BoltPattern csv file it needs to access and checks if that file is present. If the file is present, it reads the values, processes them into two lists (one for each of in the BoltPattern file), and checks the values to make sure they are valid. If they are invalid, the function reports which value is wrong and where it is and writes this information to the “Error Report.txt” file.

Design consideration of note, as with Import.Song(), this function is designed to work with Import.Song() to be able to accommodate many different X-Beams in case Micro-Vu scales this project up.

**Gantry.Move(Destination, Probe=False)**

This Move() function is a function in the Gantry.py python module file. This function controls the gantry, moving it along the length of the X-Beam.

Move() takes two inputs. First, Destination is the position in mm along the X-Beam the gantry is to move to. Second, Probe indicates if the function will call the Probe.Probe() function after if finishes moving the gantry. Probe defaults to False as we decided that the user should consciously declare using the probe in the function call.

Move() has one of three outputs. If there was an error, it returns “Error Occurred” in case an error occurred. If it took a valid probe reading, it returns the value of the probe reading. Otherwise, it returns a “Done”.

This function has no significant design considerations.

**Gantry.Home()**

This Home() function is a function in the Gantry.py python module file. It has no inputs and outputs either a “Done” or “Error Occurred” if there were or were no errors during the function respectively. The function homes the gantry to a position just off its limit switch.
This function has no significant design considerations.

BeamActuator.Move(Destination, Probe = False)
This Move() function is a function in the BeamActuator.py python module file. The
function controls the beam actuator to move the end of the X-Beam up and down in order
to level the X-Beam.

Move() takes two inputs. First, Destination is the position in mm the beam actuator is to
lift the X-Beam above the hardstop. Second, Probe indicates if the function will call the
Probe.Probe() function after if finishes moving the gantry. Probe defaults to False as we
decided that the user should consciously declare using the probe in the function call.

Move() has one of three outputs. If there was an error, it returns “Error Occurred” in case
an error occurred. If it took a valid probe reading, it returns the value of the probe
reading. Otherwise, it returns a “Done”.

This function has no significant design considerations.

BeamActuator.Home()
This Home() function is a function in the BeamActuator.py python module file. It has no
inputs and outputs either a “Done” or “Error Occurred” if there were or were no errors
during the function respectively.

The function homes the beam actuator to a position just off its limit switch.

This function has no significant design considerations.

RailAct.Home(Side)
This Home() function is a function in the RailAct.py python module file. It has one input
indicating which rail actuator is to be homed and outputs a “Done” when finished. As with
the other Home functions, this function homes the selected rail actuator.

Design consideration of note, stall is not checked in this Home function because the rail
actuators are retracting and pose no chance or crushing an object, thus the stall check was
not programmed into this function.

RailAct.Move(Side, Destination, stall = 90)
This Move() function is a function of the RailAct.py python module file. This function
moves the selected actuator to the destination at a given stall threshold.

It has three inputs. Side indicates which motor is being moved while Destination indicates
position the actuator will move to in steps. Stall is a variable that defaults to 90 and
determines the stall threshold of the rail actuator for this move. The default of 90 is strong
enough to pinch bodily parts, but not crush anything.

Design consideration of note, stall is set at 90 so that the user must specify a higher stall
threshold if they want higher torque. This was done to prevent the chance of the system
being damaged due to the rail actuator crashing on an object.

setup.py
This python module file that when imported does the following:
1. Define FileCheck and zero_flags
2. Check files using FileCheck
3. Create buffer objects from task_share.py
4. Zeroes buffers using zero_flags
5. Define pins
6. Create boards for stepper drivers
7. Parameterize stepper driver boards
8. Define functions to run the DC motors and solenoids
9. Define a Mode() function to read the 3 position switch

The functions in setup are simple and only FileCheck will be covered here. For further
details, consult the script file of setup.py in Appendix P9.

setup.FileCheck()
FileCheck is a function of the setup.py python module file. It checks for system critical files
and has its set of code to handle an error. The function has no inputs and outputs a string
if an error occurred.

Design consideration of note, because this function can throw an error before any error
flag buffers are or can be created (ie if task_share.py is missing), this function has its own
error handling section which is identical to the one in main.Lights_Sound_Action().

main.py
This is the main program which is run upon the board’s initialization. This file contains a
number of functions which will be discussed in the following sections. The main program
itself basically initializes the machine and then waits in a while loop checking the three
position switch using Mode() from setup.py and executing the selected function. Due to
time constraints, this part of the program was replaced with a test version which operated
in the same way except instead of running the function Calibration_Mode(),
Sleep_Mode(), Leveling_Mode(), and Assembly_Mode(), the test version ran test1(),
Sleep_Mode(), and test2().
**main.Lights_Sound_Action()**

Lights_Sound_Action is a function in main.py which toggles the LEDs and buzzer according to the buffer objects used as error flags. It determines which error flag is raised, sets some internal variables, and then alternates the LEDs for that error and the buzzer between on and off at 1 second intervals. If the user hits the green go button once, the buzzer turns off. If the user hits the Go button again, the function finishes. If there is no error, the function will turn on the green LED and turn the buzzer on for one second.

This function has a sister function Lights_Sound_Off() which turns off all the LEDs and buzzer. Both functions have no inputs or outputs, however both utilize the error buffer flags.

Design consideration of note, this function is pretty complex in order to make it easier to input new combinations of LED patterns for different errors. It only takes 5 lines of code at most where as how I originally had it could take 30+ lines of codes.

Additionally, this function does not cover the error from setup.FileCheck() since that function has to be handled separately.

**main.ErrorHandler()**

ErrorHandler() is a function in the main.py file which writes error messages to the “Error Report”.txt file depending on the buffer error flags raised before calling the Lights_Sound_Action() function. It has no inputs or outputs otherwise. There are four error states which are not covered by ErrorHandler as those errors handle themselves. Those exceptions are missing files, incorrect piano switchboard combination, and errors importing the BoltPattern or Calibration csv files.

**main.Home(*arg)**

This Home function in the main.py file utilizes the other Home() files from different modules to home the system. It take as many inputs as you want, checking the inputs for strings indicating which object you want homed. For example, if “All” were one of the inputs, the function would home the entire system. This design choice was chosen to make it easier to home various motors, solenoids, and the probe.

**main.Calibration_Mode()**

This function in the main.py utilizes the previous functions to to calibrate the X-Beam by running the gantry to one of two spots, taking a measurement, and calculating the difference. This is supposed to calibrate difference in heights along the rails the gantry runs along out of the system.

This function has no inputs or outputs. It utilize error flag buffers and the csv files for
information and writes to the Calibration csv file for the X-beam being worked on.

main.Sleep_Mode(Input)
This function of the main.py homes the system and does nothing until the user has selected a different mode on the three position switch. It is effectively the machine’s off mode. It has an input which is the time the machine spent checking the three position switch; however, it was never really used in the program as intended and might be obsolete. The same can be said for it only output which is the time spent in sleep mode.

main.Leveling_Mode()
This function of the main.py levels the X-Beam. It has no inputs or outputs.

main.Assembly_Mode()
This function moves the gantry to a position along the X-Beam as per the information supplied in the BoltPattern csv file for the X-Beam being worked on and calls the TorqueDown function. It repeats itself until it’s moved to every position indicated in the csv file.

main.TorqueDown(Input)
This function in the main.py is used by the Assembly mode to bolt down the rails. It uses the rail actuators to press down on the rails, activates the screwdriver, torques the bolt, and homes the actuator and screwdriver. The function has one input which indicates the side that is being torqued down. This determines which rail actuator and screwdriver is used.

5.5. Testing the Design
After the XBAS was assembled and inspected, various aspects were tested to validate whether or not the design fulfilled the engineering specifications we set forth. It should be noted that our screwdriver actuators failed to function because the solenoids were far too underpowered. Also, the SPI communication protocol kept messing up the stepper boards so that only one board could be used per controller run. The list of all the testing is listed in Appendix J as the DVPR.

5.5.1. Survey Testing
In this test, we planned to ask a number of engineering students to help test the machine by using it, the results of which would be used to check multiple specifications such as how easy the XBAS is to load, how easy it is to teach a new user to use the XBAS, and how easy the XBAS is to operate. We were unable to reach this stage of development and recommend that this be done in the future if possible.
Loading by User
One test that we had planned was to ask volunteers to do is load the X-Beam into the machine. This would allow us to see how easy it is for a user to load the X-Beam onto the constraints and hardstop as well as check for awkward loading positions which could lead to injury. A failure would be considered to be awkward loading positions, injury during loading, or inability to load the X-Beam into position without assistance. While we did not find any students to help us test the machine, loading the machine ourselves proved difficult as the X-Beam could not always be pushed up the hardstop.

Learning Time
Another test that we wanted to do during the survey was to see how long it takes to teach someone to use the X-Beam. A failure would have been if it takes more than 10 minutes to teach someone to operate the XBAS. We were unable to complete this test because we ran out of time trying to make our machine functional.

Ease of Use
Another survey that we wanted to ask volunteers to do is rate how easy they found the machine to use on a one to five scale with five being easy and one being hard. A failure would be an average below three. We were unable to complete this test because we ran out of time trying to make our machine functional.

5.5.2. Iterative Testing
In this series of tests, we would run the XBAS multiple times to check specifications not checked during the survey such as reliability and how long the XBAS takes to complete assembly of one X-Beam. We did not reach this stage of testing due to time constraints.

Rail Straightness
After every run, we would have checked the straightness of each rail individually by hand using a dial probe in the same setup as how Micro-Vu checks their rails as shown in Figure 7. As stated in the design specifications, a failure was if the rails vary in straightness by more than two micrometers in total. We were unable to complete this test because we were unable to run the rail actuators and finish the Assembly section of the program.

Rail Parallelism
After every run, we would have checked the parallelism of the rails by measuring two spots on both rails. If the change in straightness for the two had differed more than 1 micrometer, then the machine failed. We were unable to complete this test for the same reason as we could not check the Rail straightness.
XBAS Speed
During every run, we planned to time the XBAS to see if it completed its task within 10 minutes of starting the machine. Failure was if the XBAS took longer than 10 minutes to assemble the X-Beam. While we were unable to perform the test, we know that the XBAS would have failed to pass this test because our gantry’s actuator was unable to move the gantry quickly and took a good 2 minutes just to travel the length of the X-Beam.

Rail Screw Torque
After every run, we would have used a torque wrench to check that the torques of each bolt is within specification. Failure would have been if any one of the bolts is not within the specified torque limits. While we were able to use the torque limiters and verify they worked correctly, because the solenoids did not function we were unable to fully test the torques of the bolts.

System Automation Reliability
After every run, we would have recorded the results. After we finished all the runs, we would have checked how many times the XBAS failed. If it had failed more than once in 20 tries, than we will consider it as the XBAS has failed to achieve the goal set forth. Due to a number of hardware and software issues, we were unable to finish the prototype and achieve the full system automation necessary to test this specification.

5.5.2. Other tests
The following test is one we can complete outside of doing a survey or iterative testing.

Error Conveyance
For this test, we purposefully caused an error to occur and check to see if the XBAS does signal the operator or personnel in the vicinity that an issue has occurred. This signal was conveyed through both a visual and auditory alarm. If the XBAS did not successfully convey error at the appropriate time, then it would have failed this test. All errors when tested successfully conveyed their errors through the flashing of lights, sound, and error messages written by the controller. Therefore, our prototype passed this test.

6. Management Plan
This section details how the team functions and what steps are required to complete the project. Critical administrative roles are listed along with which team member is responsible for accomplishing it and what duties that team member must fulfill. Each member will be able to delegate his responsibilities to other members at his discretion if necessary. We also intend to switch roles throughout the course of this project for the sake of allowing each team member the opportunity to experience the responsibilities of each position. Additionally, we have included a
general timeline of key events and deadlines pertaining to the project.

6.1. Administrative Roles
These are roles that we have assigned for the fall quarter and are focused mostly on team dynamics, resources, communication, and documentation.

6.1.1. Communications Officer
Robert Tam is the main point of contact between the project team and the project sponsor, Ian Davison at Micro-Vu. Robert facilitates meetings with the project sponsor and with lab advisor, Eileen Rossman, Professor in the Cal Poly ME Department. In case Robert cannot complete his duty, Joseph has volunteered to take his place.

6.1.2. Treasurer
Whittaker Hamill manages the team’s funds. He allocates the team’s funds for build materials and travel as necessary and reviews part orders before they are made. If Whittaker cannot complete his duty, Joseph has volunteered to take his place.

6.1.3. Secretary
Joseph Falcao maintains an information repository for the team. Most of this information is saved on Google Drive, in a folder shared with the rest of the team. Joseph also takes detailed notes during interactions with advisors and sponsors. Joseph will be the last person to review and edit the team’s documents and outgoing emails. Joseph will be assisted and replaced if necessary by Whittaker.

6.1.4. Manager
Whittaker Hamill is in charge of managing the team progress which includes making the weekly progress reports for our lab advisor. Robert has volunteered to take his place if Whittaker cannot complete his duties.

6.2. Subsystem Design
These roles are more focused on the actual responsibilities for development of the X-Beam. However, because development is still far away, these are more tentative roles that we will re-evaluate when we begin the design process of the XBAS.

6.2.1. Structure Design Lead
Joseph ensures the structural integrity of the machine. He has performed material
selection and sized mechanical parts for sufficient strength and stiffness. He is assisted by Whittaker.

6.2.2. Controller Software Lead
Robert will supervise the code on which the XBAS runs. He will be the first member proficient with each of the software packages required to program the robot. When necessary, he will instruct the other team members on how to use these packages and assign them software-related tasks. Whittaker will assist Robert in this.

6.2.3. Controller Hardware Lead
Whittaker is in charge of designing and specifying parts of the electrical hardware for the XBAS. This includes circuit design, selection of parts used in those circuits, selection of controllers and motor shields, and wire management. He is assisted in this by Robert.

6.2.4. Motor Implementation Lead
Robert selects appropriate motors and actuators using data from calculations, motor specifications, and recommendations from engineers. Joseph will assist Robert with the implementation of the motors.

6.2.5. Manufacturing Lead
Joseph decides which manufacturing processes will be used to make the product. He will arrange dates for purchasing materials and working in the machine shops. Robert will assist Joseph in deciding the method of manufacturing.

6.2.6. CAD
Whittaker heads the CAD for both the mechanical and electrical systems, delegating work to both Joseph and Robert who will focus on the mechanical and electrical CAD work respectively, leaving Whittaker to oversee the integration of the two systems.

6.4. Key Events and Deadlines
Below in Table 4, we summarized the key events and deadlines for the X-Beam Precision Rail Alignment senior project, as specified in the course syllabus. A more detailed timeline is provided in the form of a Gantt chart discussed below in section 6.5. As of this report, we’ve completed the Critical Design Report and will be moving onto ordering parts and finalizing our manufacturing plan.
Table 5. Key Events and Deadlines for Senior Project

<table>
<thead>
<tr>
<th>Item</th>
<th>Quarter</th>
<th>Week</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Proposal</td>
<td>Fall</td>
<td>5</td>
<td>10/25/16</td>
</tr>
<tr>
<td>Preliminary Design Report</td>
<td>Fall</td>
<td>8</td>
<td>11/17/16</td>
</tr>
<tr>
<td>Critical Design Report</td>
<td>Winter</td>
<td>5</td>
<td>02/07/17</td>
</tr>
<tr>
<td>Manufacturing and Test Review</td>
<td>Winter</td>
<td>10</td>
<td>03/16/17</td>
</tr>
<tr>
<td>Project Update Report</td>
<td>Winter</td>
<td>10</td>
<td>03/16/17</td>
</tr>
<tr>
<td>Project Hardware/Safety Demo</td>
<td>Spring</td>
<td>5</td>
<td>05/02/17</td>
</tr>
<tr>
<td>Final Design Project Expo</td>
<td>Spring</td>
<td>9</td>
<td>06/02/17</td>
</tr>
<tr>
<td>Final Design Hardware Handoff</td>
<td>Spring</td>
<td>9</td>
<td>06/02/17</td>
</tr>
<tr>
<td>Final Design Report</td>
<td>Spring</td>
<td>9</td>
<td>06/02/17</td>
</tr>
</tbody>
</table>

6.5. Gantt Chart
The Gantt Chart is a useful organization tool to determine how long the steps in a given process will take for a project. Our Gantt chart can be located in Appendix H. Judging by the complexity of our project, we gave ourselves extensive time to manufacture and test our system by pushing several tasks up.

7. Tear Down
In this section, we will summarize what worked and what didn’t work with the XBAS.

7.1. What Worked
1. All the stepper motors worked separately
2. The relays for the DC motors and solenoids work
3. DC motors worked
4. The Heidenhain probe could take readings
5. Granite surfaces and dimensions were within tolerance
6. Leadscrew worked
7. Gantry could be moved to a position with a variance of 0.5 mm every run
8. Gantry ran without binding
9. Both Gantry Housing and Electronics Housing were structurally sound
10. Beam actuator could be moved to a position with a variance of 0.25 mm
11. Rail actuators could be moved to a position consistent with a variance 0.1 mm
12. Electronics worked
13. The buttons, LEDs, and buzzer worked
14. Reading the three position switch worked
15. Emergency Stop and the interrupt worked
16. The error handling and writing to text file worked
17. Error conveyance worked
18. Storing, writing, and importing data from csv files (Calibration and BoltPattern) worked
19. Checking the piano switch board for the switch combination and converting that to a number to check which file to use worked
20. Hardstop did not break and protected the Beam Actuator
21. X-Beam was fully constrained by the line constraint and Beam Actuator when leveled
22. Gantry could traverse the length of the X-Beam to all necessary positions
23. The program for the XBAS Sleep Mode worked

7.2. What Did Not Work and What Should Change
1. Of the four stepper motor actuators, the Beam and Gantry Actuators could not work in conjunction with the rail actuators.
2. Program for the Calibration Mode was never tested.
3. Program for the Leveling and Assembly Mode were never tested.
   a. Assembly could not be tested as the rail actuators could not be used after the gantry had been moved, see item 1.
4. The gantry’s limit switch was not aligned with the bolt on the back of the gantry.
5. The housing was not properly constrained. It was tilted backwards, placing strain on the shaft coupling between the NEMA23 motor and leadscrew.
6. The solenoids were undersized.
7. Shaft coupler for the leadscrew was undersized.
8. Bearing at the end of the leadscrew was not fixed inside the leadscrew support.
9. Housing was slightly too small and electronics were barely able to fit inside.
10. Hole placement on the housing for the LEDs were wrong.
11. Re-organize the button and LED positions for better placement
12. Probe reference tick pin did not read values
13. Loading the X-Beam was difficult as the Gantry in the home position was in the way and the X-Beam would often get stuck on the hardstop
14. Gantry moved at a much slower speed than required.
15. XBAS is not easy to move
16. Resizing screwdriver components so that the screwdriver bit does not collide with the rails on the X-Beam
17. A better connection on the SG6DM’s three pin port
18. Replace the four pin connecting wire to the Heidenhain probe with a longer wire
19. Beam Actuator runs at a very high temperature
20. Lead screw was slightly too short

7.3. Proposed Solutions
In this section, we will address possible solutions for the issues in section 7.2.

1. For item 1, if Micro-Vu continues to change solder bridges on one of the stepper drivers to use a whole new SPI port or try using commands to clean out RAM and see if that allows the controller to use all the stepper motors at the same time.
2. For items 4, 5, 9, 10, and 11, re-making the housing with better hole tolerancing and flanges on the bottom to bolt to the granite would fix the issues we saw in our prototype.
3. For items 6 and 7, sizing up the coupler and solenoids should solve the issues we saw with the prototype.
4. For item 8, the lead screw raiser was supposed to have a groove on the inside where a snap ring could be placed to constrain the bearing. This was not present, so we suggest checking with Misumi about this part.
5. For item 12 and 17, getting a female 3 pin for the SG6DM would solve both the issue of the reference pin not being read and the reliability issues we faced with those three pins.
6. For item 13, we had a few solutions.
   a. The first was to extend the length of the hardstop so that the angle at which the user pushes the X-Beam is smaller.
   b. Second, add a second hardstop identical to the other on the other side of the beam actuator.
   c. Third, add rollers or some device to prevent the X-Beam’s edge being pushed along the granite which produces a lot of friction.
7. For item 14 and 20, we feel that replacing the leadscrew with a ball screw would significantly decrease friction in the system and increase the gantry’s overall speed, not to mention Micro-Vu can pick a longer ball screw. If this is still not enough to meet the
minute time limit, then changing the gantry actuator to a different system entirely such as a pneumatic system should be considered.

8. For item 15, assuming the XBAS needs to be moved, placing the XBAS on a cart would work. Another solution would be to add a frame with table legs to the XBAS so that there is enough room to fit a fork lift under.

9. For item 16, resizing the screwdriver parts or possibly even re-manufacturing the gantry to give the screwdrivers more space to clear the rails on the X-Beam are both valid options to solve the issue.

10. For item 18, Micro-Vu can contact Heidenhain for a longer wire.

11. For item 19, adding heat sink fins or active cooling would help the issue; however, this may not be a big issue as turning off the beam actuator in the software when the actuator is not being used greatly reduces the chances of the actuator overheating.

8. What’s Next?
This section discusses additional tasks and modifications for the future advancement of the XBAS. Since the XBAS prototype that was developed for this senior project had a limited time frame and limited scope, future work will still need to be implemented by Micro-Vu in order to further develop this prototype into a productive working unit in Micro-Vu’s production line.

8.1. Necessary Adjustments for Full Functionality
Before discussing future improvements, there are several issues that need to be addressed to make the XBAS fully functional. First, the controller and stepper driver should be replaced with an appropriate PLC system, such as the Siemens S7. New stepper drivers will also be needed, but the choice of these are up to Micro-Vu. Beyond the controller, there are several hardware changes that would help the machine achieve its specifications such as using a ball screw instead of the leadscrew and anti-backlash nut and sizing up the solenoids in the screwdriver actuators so that they function.

Another critical aspect of the XBAS that needs to be renovated is a more thorough calibration process. This fully-developed calibration process should require the need to calibrate the touch probe three times using Abby’s Law every time there is a major change to the machine or machine’s environment such as transporting or moving the machine or having it in a new atmospheric environment. This calibration should also be required over a pre-defined duration of time such as six months or however much Micro-Vu deems necessary for the precise performance of the XBAS.

8.2. Future Recommendations
Beyond the scope of this project, there are a few things Micro-Vu is considering to do with this
project. Most notable is the scaling up of this design for larger X-Beam assemblies. In order to do this, we will list out what needs to be done to our current design in order to scale it up.

First, the following parts must be resized in order to make the gantry have a far enough range to cover the length of the desired X-Beam.

1. Gantry Actuator Lead Screw
2. Granite Base Plate
3. Granite Parallel Gauge Blocks
4. Gantry Rails

Additionally, in order to accommodate the new leadscrew, the gantry legs might have to be made thicker to accommodate the longer and possibly thicker lead screw. This also means that the leadscrew support, leadscrew support raiser, flexible shaft coupling, and stepper motor may all need to be scaled up as well. It should be noted that any changes made to the parts in the future should also be reflected in the part and assembly drawings to ensure that they are up to date.

Another recommendation that Micro-Vu can look into in the future is to develop a more effective way to load the X-beam into the XBAS. The current prototype’s process of manually loading the X-beam onto the granite parallels and pushing it up the hard stop ramp into position does work, but it is still fairly difficult to handle the heavy weight of the X-beam along with moving it through all the tight constraints when positioning the X-beam. Depending on how sophisticated Micro-Vu wants this process to be, potential solutions can range from simply adding in rollers on top of the granite plate to come in contact with the bottom of the X-beam when it is being pushed in to assist the movement of the X-beam or even more complex ideas such as developing a separate automated process for loading the X-beam into place. If the XBAS is to be used for assembling large scale quantities of X-beams then investing the time and effort into developing an automated process for loading the X-beam can be a viable option, though for now, simply adding in rollers or a similar mechanism to assist the movement of the X-beam should be adequate.

It should also be noted that the electronics housing that was manufactured for this XBAS prototype has slight imperfections that make it difficult to properly constrain the lead screw, thus affecting the lead screw’s alignment with the rails underneath it. Although the design of the electronics housing is fine, it is the manufacturing imperfections from bending the sheet metal that make it difficult to properly align the lead screw. For now, the imperfections are not that severe to where it will significantly affect the functionality of this working prototype, though if future models of this XBAS design are to be built it is recommended that the electronics housing be manufactured with tighter tolerances or more precise manufacturing methods to ensure that the lead screw is more precisely aligned with the rails to achieve better results.
References


## Table of Appendices

Appendix A: Customer Requirements ................................................................. 62
Appendix B: QFD ................................................................................................. 63
Appendix C: Pugh Matrices ............................................................................... 64
Appendix D Table of Contents ............................................................................. 67
  Appendix D1: Gantry X-Axis Movement Actuator............................................. 68
Appendix D2: Linear Rail Actuator Calculations .................................................. 70
    D2.1. Case 1 .................................................................................................. 71
    D2.2. Case 2 .................................................................................................. 72
    D2.3. Case 3 .................................................................................................. 74
Appendix D3: Gantry Structural Analysis .............................................................. 76
Appendix D4: Tolerance Stack up ....................................................................... 89
Appendix D5: X-Beam Leveling Actuator ............................................................. 91
Appendix F: List of Subassembly and Parts Drawings ........................................... 93
  Appendix F1: Full Assembly ............................................................................. 94
  Appendix F2: Granite Subassembly ................................................................. 95
  Appendix F3: Housing Subassembly ................................................................. 96
  Appendix F4: Beam Actuator Subassembly ....................................................... 97
  Appendix F5: Gantry Subassembly .................................................................. 98
  Appendix F6: Gantry Actuator Subassembly .................................................... 99
  Appendix F7: Screwdriver Actuator Subassembly ........................................... 100
  Appendix F8: Bearing Attachment Plate Configuration 1 ............................. 101
  Appendix F9: Bearing Attachment Plate Configuration 2 ............................. 102
  Appendix F10: Solenoid Pullout Pin ................................................................. 103
  Appendix F11: Electronics Housing ................................................................. 104
  Appendix F12: Gantry Leg Configuration 1 ...................................................... 105
  Appendix F13: Gantry Leg Configuration 2 ...................................................... 107
<table>
<thead>
<tr>
<th>Appendix F14: Gantry Probe Covering</th>
<th>109</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix F15: Gantry Top</td>
<td>110</td>
</tr>
<tr>
<td>Appendix F16: Gearbox Housing</td>
<td>113</td>
</tr>
<tr>
<td>Appendix F17: Granite Parallel Gauge Block Constraints</td>
<td>114</td>
</tr>
<tr>
<td>Appendix F18: Granite Plate</td>
<td>115</td>
</tr>
<tr>
<td>Appendix F19: Hardstop</td>
<td>118</td>
</tr>
<tr>
<td>Appendix F20: Leadscrew Raiser</td>
<td>119</td>
</tr>
<tr>
<td>Appendix F21: Line Constraint</td>
<td>120</td>
</tr>
<tr>
<td>Appendix F22: Leadscrew Raiser Drawing</td>
<td>121</td>
</tr>
<tr>
<td>Appendix G: Manufacturing Plan</td>
<td>122</td>
</tr>
<tr>
<td>Appendix H: List of Part Job Planners</td>
<td>123</td>
</tr>
<tr>
<td>Appendix H1: Bearing Attachment Plate Configuration 1</td>
<td>124</td>
</tr>
<tr>
<td>Appendix H2: Bearing Attachment Plate Configuration 2</td>
<td>125</td>
</tr>
<tr>
<td>Appendix H3: Gantry Leg Configuration 1</td>
<td>126</td>
</tr>
<tr>
<td>Appendix H4: Gantry Leg Configuration 2</td>
<td>127</td>
</tr>
<tr>
<td>Appendix H5: Gantry Top</td>
<td>128</td>
</tr>
<tr>
<td>Appendix H6: Hardstop</td>
<td>129</td>
</tr>
<tr>
<td>Appendix H7: Lead Screw Support Raiser</td>
<td>130</td>
</tr>
<tr>
<td>Appendix H8: Line Constraint</td>
<td>131</td>
</tr>
<tr>
<td>Appendix I: Gantt Chart</td>
<td>132</td>
</tr>
<tr>
<td>Appendix J: DVPR</td>
<td>133</td>
</tr>
<tr>
<td>Appendix K: Table of Inspection Sheets</td>
<td>134</td>
</tr>
<tr>
<td>Appendix K1: Bearing Attachment Plate Configuration 1</td>
<td>135</td>
</tr>
<tr>
<td>Appendix K2: Bearing Attachment Plate Configuration 2</td>
<td>136</td>
</tr>
<tr>
<td>Appendix K3: Gantry Top</td>
<td>137</td>
</tr>
<tr>
<td>Appendix K4: Hardstop</td>
<td>146</td>
</tr>
<tr>
<td>Appendix K5: Leadscrew Support Raiser</td>
<td>147</td>
</tr>
</tbody>
</table>
Appendix A: Customer Requirements

Table A1. Customer Requirements. A list of both the requirements and wishes or goals put forward by Micro-Vu as items they wish to get out of this project. From left to right is the requirement number, requirement description, and the letter R or W denoting whether this is really a requirement or wish.

<table>
<thead>
<tr>
<th>Requirement #</th>
<th>Description</th>
<th>Requirement or Wish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rails Must be Straight</td>
<td>R</td>
</tr>
<tr>
<td>2</td>
<td>Rails Must be Parallel to Each Other</td>
<td>R</td>
</tr>
<tr>
<td>3</td>
<td>Automated Rail Alignment</td>
<td>R</td>
</tr>
<tr>
<td>4</td>
<td>Automated Torqueing of the Screws</td>
<td>W</td>
</tr>
<tr>
<td>5</td>
<td>Fast</td>
<td>W</td>
</tr>
<tr>
<td>6</td>
<td>Low-skill Level</td>
<td>W</td>
</tr>
<tr>
<td>7</td>
<td>Must Fit on a Table</td>
<td>W</td>
</tr>
<tr>
<td>8</td>
<td>Must have a Length to Accommodate X-Beam</td>
<td>R</td>
</tr>
<tr>
<td>9</td>
<td>Scalability</td>
<td>W</td>
</tr>
<tr>
<td>10</td>
<td>Must Use Power Available at the Site</td>
<td>R</td>
</tr>
<tr>
<td>11</td>
<td>Beam Must be Fully Constrained</td>
<td>R</td>
</tr>
<tr>
<td>12</td>
<td>Basic Safety</td>
<td>R</td>
</tr>
<tr>
<td>13</td>
<td>Must Convey Error to Operator</td>
<td>W</td>
</tr>
<tr>
<td>14</td>
<td>Reusable</td>
<td>R</td>
</tr>
<tr>
<td>15</td>
<td>Repeatable</td>
<td>R</td>
</tr>
<tr>
<td>16</td>
<td>Cost</td>
<td>R</td>
</tr>
</tbody>
</table>
Appendix B: QFD House of Quality

Senior Project: X-Beam Rail Alignment System a.k.a XBAS
Date: 10/23/16
Revision: Appendix 10

Direction of Improvement:

Weight Chart

<table>
<thead>
<tr>
<th>Column #</th>
<th>Engineering Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Must use power available at location</td>
</tr>
<tr>
<td>2</td>
<td>Competitor #2: FARO Laser Tracker</td>
</tr>
<tr>
<td>3</td>
<td>Competitor #4: Patent US5768137A</td>
</tr>
<tr>
<td>4</td>
<td>Competitor #1: Micro-VU's Current</td>
</tr>
<tr>
<td>5</td>
<td>Long Range Straightness Optics</td>
</tr>
<tr>
<td>6</td>
<td>Must fit a given X-beam</td>
</tr>
<tr>
<td>7</td>
<td>Must maintain a parallelism to each other.</td>
</tr>
<tr>
<td>8</td>
<td>Basic Safety</td>
</tr>
<tr>
<td>9</td>
<td>How Much: Target</td>
</tr>
<tr>
<td>10</td>
<td>How: Column #</td>
</tr>
<tr>
<td>11</td>
<td>Maximum Relationship</td>
</tr>
<tr>
<td>12</td>
<td>Relative Weight</td>
</tr>
</tbody>
</table>

Current Product Assessment - Engineering Specifications

<table>
<thead>
<tr>
<th>Column #</th>
<th>Engineering Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Micro-Vu Shop Technician</td>
</tr>
<tr>
<td>1</td>
<td>Micro-Vu (Company)</td>
</tr>
<tr>
<td>2</td>
<td>Micro-Vu Employee</td>
</tr>
</tbody>
</table>

Relative Weight

<table>
<thead>
<tr>
<th>Column #</th>
<th>Engineering Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Micro-Vu Shop Technician</td>
</tr>
<tr>
<td>1</td>
<td>Micro-Vu (Company)</td>
</tr>
<tr>
<td>2</td>
<td>Micro-Vu Employee</td>
</tr>
</tbody>
</table>

Relative Weight Chart

<table>
<thead>
<tr>
<th>Column #</th>
<th>Engineering Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Micro-Vu Shop Technician</td>
</tr>
<tr>
<td>1</td>
<td>Micro-Vu (Company)</td>
</tr>
<tr>
<td>2</td>
<td>Micro-Vu Employee</td>
</tr>
</tbody>
</table>

Micro-Vu Shop Technicians

- Maximum Relationship
- Moderate Relationship
- Weak Relationship

Micro-Vu (Company)

- Maximum Relationship
- Moderate Relationship
- Weak Relationship

Micro-Vu Employees

- Maximum Relationship
- Moderate Relationship
- Weak Relationship

Christopher Battles
Template Revision: 0.9  Date: 4/23/2010
## Appendix C: Pugh Matrices

Table C1: Pugh Matrix of systems for moving the gantry or carriage along the x-axis

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Concept</th>
<th>Datum: Hand pushed Granite Block on Granite Surface</th>
<th>Lead/Ball Screw</th>
<th>Belt</th>
<th>Pulley</th>
<th>Rope and Motor</th>
<th>Rack and Pinion</th>
<th>Wheeled Granite Block/Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigidity</td>
<td>S</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>S</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>Repeatability</td>
<td>S</td>
<td>+</td>
<td>-</td>
<td>S</td>
<td>S</td>
<td>+</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>Speed of Task Completion</td>
<td>S</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
</tr>
<tr>
<td>Projected Cost</td>
<td>S</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Complexity of Assembly</td>
<td>S</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Ease of Maintenance</td>
<td>S</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td># of S</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td># of +</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td># of -</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
Table C2: Pugh Matrix of systems for adjusting the rails along the γ-axis

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Concept</th>
<th>Datum: Hand actuated</th>
<th>Pneumatic</th>
<th>Hydraulic</th>
<th>Lead/ball screw</th>
<th>Rack and Pinion</th>
<th>Pulley</th>
<th>Rope and Motor</th>
<th>Belt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Precision</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Speed of Task Completion</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Size</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rigidity</td>
<td>S</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Projected Cost</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
</tr>
<tr>
<td>Complexity of Assembly</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ease of Maintenance</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Power</td>
<td>S</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td># of S</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td># of +</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td># of -</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Table C3: Pugh Matrix of systems for measuring and aligning the rails.

<table>
<thead>
<tr>
<th>Datum: Visually read dial indicators</th>
<th>gauge Block</th>
<th>Position Feed Back Actuator</th>
<th>Force Feed Back Actuator</th>
<th>Probes</th>
<th>Interferometer</th>
<th>Laser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Repeatability</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Cost</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Complexity of Assembly</td>
<td>S</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Programming</td>
<td>S</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td># of S</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># of +</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td># of -</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Table C4: Pugh Matrix of systems for torqueing the screws on the rail

<table>
<thead>
<tr>
<th></th>
<th>Datum: Hand screw</th>
<th>Hand screw with torque limiter</th>
<th>Automated Torque feedback controlled screwdriver</th>
<th>DC motor with torque limiter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Cost</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Programmability</td>
<td>S</td>
<td>S</td>
<td>-</td>
<td>S</td>
</tr>
<tr>
<td>Vibration</td>
<td>S</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Repeatability</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Precision</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td># of S</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td># of +</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td># of -</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table C5: Pugh Matrix of gantry or carriage frame styles

<table>
<thead>
<tr>
<th></th>
<th>Datum: Granite Block with L overhang</th>
<th>U traversing X-beam length</th>
<th>U gantry over x beam in z-direction</th>
<th>Square Gantry that wraps around X-Beam</th>
<th>L shape overhang</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigidity</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>S</td>
</tr>
<tr>
<td>Weight</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>S</td>
</tr>
<tr>
<td>Constrained</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>S</td>
</tr>
<tr>
<td>Loading Beam</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Durability</td>
<td>S</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Scalability</td>
<td>S</td>
<td>-</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td># of S</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td># of +</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td># of -</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
# Appendix D Table of Contents

Appendix D1: Gantry X-Axis Movement Actuator .......................................................... 68

Appendix D2: Linear Rail Actuator Calculations ............................................................ 70
  D2.1. Case 1 .................................................................................................................. 71
  D2.2. Case 2 .................................................................................................................. 72
  D2.3. Case 3 .................................................................................................................. 74

Appendix D3: Gantry Structural Analysis ................................................................. 76

Appendix D4: Tolerance Stack up .............................................................................. 89

Appendix D5: X-Beam Leveling Actuator ................................................................. 91
Appendix D1: Gantry X-Axis Movement Actuator

The linear screw and motor used for the gantry’s actuation were sized according to the time constraint listed in the design specifications. In order to calculate the time of the 10 minutes specified that the motor had to move the gantry, we had to calculate the amount of time used for the other functions of the XBAS first.

First we calculated the time required to torque down all the screws. We assume that it takes 10 revolutions per screw to max out the torque limiter. The calculation of time is shown below in equation eqD1.1 and a sample calculation is shown below.

\[
\text{Time} = \frac{\# \text{screws} \times \text{revolutions/screw}}{\text{rpm}} \quad \text{eqD1.1}
\]

Table D1.1. Calculated Time required to torque down the screws

<table>
<thead>
<tr>
<th>Screwdriver RPM</th>
<th>45.9 rpm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revolutions to torque screw</td>
<td>10 revolutions/screw</td>
</tr>
<tr>
<td># of screws to torque</td>
<td>18 screws</td>
</tr>
<tr>
<td>Time</td>
<td>3.9 minutes</td>
</tr>
<tr>
<td>This is an assumption</td>
<td></td>
</tr>
</tbody>
</table>

Then we calculate the time taken for stopping and starting the gantry. We assumed the gantry would take 1 second to stop and accelerate. This is multiplied by the sum two things:

First is the number of screws since the gantry must stop at and start from each screw position once.

Second is the number of runs the gantry makes along the X-Beam’s length to both level the X-beam and torque all the screws. This is because the gantry has to stop at the end of each run of the length of the X-Beam. We assumed for the second number a value of six, four of which are for leveling the X-Beam. A single run is 840mm. The equation used to calculate time is shown in eqD2.2. A sample calculation can be seen on the next page in Table D1.2.

\[
\frac{\text{Time to start moving} + \text{Time to stop moving}}{\text{(60s/min)}} \times (\text{screws} + \text{runs}) \quad \text{eqD1.2}
\]
Table D1.2. Calculated time spent making the gantry stop or go.

<table>
<thead>
<tr>
<th>Time to start moving gantry</th>
<th>0.5 s</th>
<th>This is an assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time for gantry to stop</td>
<td>0.5 s</td>
<td>This is an assumption</td>
</tr>
<tr>
<td>Total number of Gantry runs along X-Beam</td>
<td>6 runs</td>
<td></td>
</tr>
<tr>
<td>Total time spent starting and stopping</td>
<td>0.4 min</td>
<td></td>
</tr>
</tbody>
</table>

After calculating the two times above, we subtracted them from the target time. We then calculated the total distance of travel by multiplying the number of runs by the distance of each run. After that, we divide the distance to traverse by the time we have left for an estimate of the velocity we need the gantry to move at. A sample calculation of what was just mentioned can be seen below in table D1.3.

From there, we selected a leadscrew and motor. For the below sample calculation in table D1.4, we demonstrate the calculation of the necessary motor rpm to achieve the required velocity given a lead screw with a pitch of 2mm.

Table D1.3. Calculation of the velocity required to fulfill time constraint

<table>
<thead>
<tr>
<th>Time Target</th>
<th>10 min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time used so far</td>
<td>4 min</td>
</tr>
<tr>
<td>Time Remaining</td>
<td>6 min</td>
</tr>
<tr>
<td>Time Remaining</td>
<td>340 s</td>
</tr>
<tr>
<td>Distance to traverse</td>
<td>840 mm</td>
</tr>
<tr>
<td>Number of runs</td>
<td>6</td>
</tr>
<tr>
<td>Total distance to traverse</td>
<td>5040</td>
</tr>
<tr>
<td>Velocity needed</td>
<td>14.82 mm/s</td>
</tr>
</tbody>
</table>

Table D1.4. Calculation of motor rpm to fulfill velocity requirement

<table>
<thead>
<tr>
<th>Lead Screw Pitch</th>
<th>2 mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nema23 Step</td>
<td>1.8 deg</td>
</tr>
<tr>
<td></td>
<td>200 steps per rev</td>
</tr>
<tr>
<td>distance per step</td>
<td>0.01 mm/step</td>
</tr>
<tr>
<td>Step/second needed</td>
<td>1482 step/sec</td>
</tr>
<tr>
<td>rpm needed</td>
<td>440 rpm</td>
</tr>
</tbody>
</table>
Appendix D2: Linear Rail Actuator Calculations

For calculating the max output force of the actuators responsible for pressing the rails down against the granite parallel gauge blocks, we looked at three cases. In all three cases, we modeled the rail as a cantilever beam with an area moment inertia equal to a rectangular cross section and set constants as listed below in table D2.1. Additionally, we stated that the actuator must be able to force the rail down by at most 20 micrometers which is what we determined to be the worst case scenario from assembling the X-Beam by hand.

The results are that in the worst case scenario as presented in Case 3, the actuator would need to output 417 N of force.

Table D2.1. Linear Rail Actuator Constant Values.

<table>
<thead>
<tr>
<th>Constants</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rail Width</td>
<td>Published Value</td>
</tr>
<tr>
<td>8 mm</td>
<td>0.008 m</td>
</tr>
<tr>
<td>Rail Height</td>
<td>Published Value</td>
</tr>
<tr>
<td>24 mm</td>
<td>0.024 m</td>
</tr>
<tr>
<td>Moment of Inertia</td>
<td>(9.22 \times 10^{-9} ) m^4</td>
</tr>
<tr>
<td>Elastic Modulus</td>
<td>for AISI 303 stainless steel</td>
</tr>
<tr>
<td>193 GPa</td>
<td>1.93E+11 Pa</td>
</tr>
<tr>
<td>Deflection Desired</td>
<td>Arbitrary Value</td>
</tr>
<tr>
<td>20 micrometer</td>
<td>0.00002 m</td>
</tr>
</tbody>
</table>
D2.1. Case 1

First case, we modeled the rail as a simple cantilever beam with one end fixed and the other end free. We calculated a necessary torque requirement of 1.3N in this case. A sample excel calculation can be seen in table D2.2 and an image of the rail modeled as a beam can be seen in figure D2.1 on the next page.

![Figure D2.1. Linear Rail Actuator Calculation Diagram 1. Rail modeled as a cantilever beam with a fixed end at the screw on the far left with a free end where force is applied above the screw on the far right.](image)

<table>
<thead>
<tr>
<th>Deflection unsupported end (x) defined in the constants as the desired deflection</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = \frac{WL}{3EI} )</td>
</tr>
</tbody>
</table>

Solved for force

| \( W = (3EI/L)x \) |

<table>
<thead>
<tr>
<th>distance from previous screw to where the actuator is adjusting the rail</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 +/- 0.1 mm</td>
</tr>
<tr>
<td>0.08 +/- 0.0001 m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Calculated Force</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3 +/- N</td>
</tr>
</tbody>
</table>
D2.2. Case 2

Second case, we modeled the full length of the rail as a simple cantilever beam with one end fixed with a pin and the other as a roller. We calculated a necessary torque requirement to be 55 N in this case. A sample excel calculation can be seen in table D2.3 on the next page and an image of the rail modeled as a beam can be seen in figure D2.2 below.

![Figure D2.2. Linear Rail Actuator Calculation Diagram 2. Entire rail modeled as a cantilever beam with one end pinned and one end free to slide.](image)
Table D2.3. Calculation of the force required actuator force output for Case 2

<table>
<thead>
<tr>
<th>Maximum deflection at load</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = W(a^2)(b^2)/3EIL</td>
</tr>
<tr>
<td>Solved for force</td>
</tr>
<tr>
<td>W = 3xEIL/(a^2)(b^2)</td>
</tr>
<tr>
<td>distance (a) from fixed screw to applied force where the second screw is to be adjusted than</td>
</tr>
<tr>
<td>80  +/-     mm</td>
</tr>
<tr>
<td>0.08 +/-     m</td>
</tr>
<tr>
<td>distance b</td>
</tr>
<tr>
<td>0.37 +/-     m</td>
</tr>
<tr>
<td>Beam Length (l) from first fixed screw to the end of the rail</td>
</tr>
<tr>
<td>0.45 +/-     m</td>
</tr>
<tr>
<td>Calculated Force</td>
</tr>
<tr>
<td>55  +/-     N</td>
</tr>
</tbody>
</table>
D2.3. Case 3

Third case, we modeled a section of the rail as a simple cantilever beam with one end fixed with a pin and the other as a roller. We calculated a necessary torque requirement to be 417N in this case. A sample excel calculation can be seen in table D2.4 and an image of the rail modeled as a beam can be seen in figure D2.3 on the following two pages.

Figure D2.3. Linear Rail Actuator Calculation Diagram 3. Table of a sample calculation from excel for the calculation of the force required by the actuator to press down the point indicated in figure D2.2 down by the desired amount for a section of the rail modeled as a cantilever beam with one end pinned and the other pinned and free to slide.
Table D2.4. Calculation of the force required actuator force output for Case 2

<table>
<thead>
<tr>
<th>Maximum deflection at load</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = W\frac{a^2b^2}{3EIL} )</td>
</tr>
<tr>
<td>Solved for force</td>
</tr>
<tr>
<td>( W = \frac{3xEL}{a^2b^2} )</td>
</tr>
</tbody>
</table>

Distance \((a)\) from fixed screw to applied force

| 80 +/- mm | 0.08 +/- m | 0.08 +/- m |
| Beam Length \((l)\) from first fixed screw to the next screw hole |
| 0.16 +/- m |

Calculated Force

| 417 +/- N |
Appendix D3: Gantry Structural Analysis

Structural Analysis of Gantry

**GIVEN:**

- $b := 0.08 \text{ m}$  \hspace{0.5cm} \text{Gantry Width}
- $t := 0.04 \cdot \text{m}$  \hspace{0.5cm} \text{Gantry Thickness}
- $h := 0.12 \cdot \text{m}$  \hspace{0.5cm} \text{Gantry Leg Height}
- $L := 0.29 \cdot \text{m}$  \hspace{0.5cm} \text{Distance across Beam}
- $a1 := 0.064 \text{ m}$  \hspace{0.5cm} \text{Distance to Actuator (1)}
- $a2 := 0.102 \cdot \text{m}$  \hspace{0.5cm} \text{Distance to Touch Probe}
- $a3 := 0.225 \cdot \text{m}$  \hspace{0.5cm} \text{Distance to Actuator (2)}
- $a4 := 0.145 \text{ m}$  \hspace{0.5cm} \text{Distance to Center}
- $a5 := 0.05 \text{ m}$  \hspace{0.5cm} \text{Distance to Screwdriver Subassembly (1)}
- $a6 := a5$  \hspace{0.5cm} \text{Distance to Screwdriver Subassembly (2)}
- $b1 := 0.226 \cdot \text{m}$  \hspace{0.5cm} \text{Distance from Actuator (1)}
- $b2 := 0.189 \cdot \text{m}$  \hspace{0.5cm} \text{Distance from Touch Probe}
- $b3 := 0.0649 \text{ m}$  \hspace{0.5cm} \text{Distance from Actuator (2)}
- $b4 := a4$  \hspace{0.5cm} \text{Distance from Center}
- $F1 := 11.77 \text{ N}$  \hspace{0.5cm} \text{Weight of Actuator (1)}
- $F2 := 13.73 \cdot \text{N}$  \hspace{0.5cm} \text{Weight of Touch Probe}
- $F3 := F1$  \hspace{0.5cm} \text{Weight of Actuator (2)}
- $F4 := 56.33 \text{ N}$  \hspace{0.5cm} \text{Weight of Gantry Top Section}
- $F5 := 9.37 \text{ N}$  \hspace{0.5cm} \text{Weight of Screwdriver Subassembly (1)}
- $F6 := F5$  \hspace{0.5cm} \text{Weight of Screwdriver Subassembly (2)}
- $Fa := 800 \text{ N}$  \hspace{0.5cm} \text{Force of Actuator}
\[ F_g := 149.54 \text{ N} \quad \text{Weight of Gantry Top Total} \]

\[ E := 2 \cdot 10^{11} \text{ Pa} \quad \text{Elastic Modulus for AISI 1018 Steel, Cold Drawn} \]

\[ n := 2 \quad \text{Factor of Safety} \]

**ASSUMPTIONS:**
1) Gantry treated as one whole unit
2) Uniform material properties throughout gantry (such as rigidity, stiffness, etc.)
3) Standard machine shop atmospheric conditions
4) Holes are negligible
5) Factor of Safety of \( n = 2 \) was selected

**FIND:** Deflections at Points 1, 2, and 3 for Gantry Top

**SCHEMATIC (FBD):**
The load cases of the gantry will be modeled as a simple supports setup as seen in the figure below where each individual loading will be represented by the corresponding equations for force, $F$, which are shown in the figure below as well. Superposition Method will be used to calculate the deflection at each point by first calculating the individual deflections caused by each individual force at that given point and then adding them up. The sum of these individual deflections from each force will give the total deflection of that given point on the gantry.

\[ R_1 = \frac{Fb}{l}, \quad R_2 = \frac{Fa}{l} \]
\[ V_{AB} = R_1, \quad V_{BC} = -R_2 \]
\[ M_{AB} = \frac{Fbx}{l}, \quad M_{BC} = \frac{Fa}{l}(l-x) \]
\[ y_{AB} = \frac{Fbx}{6EI} \left( x^2 + b^2 - l^2 \right) \]
\[ y_{BC} = \frac{Fa(l-x)}{6EI} \left( x^2 + a^2 - 2lx \right) \]

Source: Shigley’s Mechanical Engineering Design 10th ed. Table A-9  p. 1023
SOLUTION:

Calculate Moment of Inertia

\[ I := \frac{b \cdot t^3}{12} \]

\[ I = (4.267 \cdot 10^{-7}) \text{ m}^4 \quad \text{Moment of Inertia} \]

Calculate Deflection at 1 using Superposition

\[ x_1 := a_1 = 0.064 \text{ m} \]

\[ y_{F1} = \left( \frac{(F1 \cdot b1 \cdot x1) \cdot (L^2 - x1^2 - b1^2)}{6 \cdot E \cdot I \cdot L} \right) \]

\[ y_{F1} = \left[ 3.32 \cdot 10^{-8} \right] \text{ m} \quad \text{Deflection from F1} \]

\[ y_{F2} = \left( \frac{(F2 \cdot b2 \cdot x1) \cdot (L^2 - x1^2 - b2^2)}{6 \cdot E \cdot I \cdot L} \right) \]

\[ y_{F2} = \left[ 4.95 \cdot 10^{-8} \right] \text{ m} \quad \text{Deflection from F2} \]

\[ y_{F3} = \left( \frac{(F3 \cdot b3 \cdot x1) \cdot (L^2 - x1^2 - b3^2)}{6 \cdot E \cdot I \cdot L} \right) \]

\[ y_{F3} = \left[ 2.5 \cdot 10^{-8} \right] \text{ m} \quad \text{Deflection from F3} \]

\[ y_{F4} = \left( \frac{(F4 \cdot b4 \cdot x1) \cdot (L^2 - x1^2 - b4^2)}{6 \cdot E \cdot I \cdot L} \right) \]

\[ y_{F4} = \left[ 2.08 \cdot 10^{-7} \right] \text{ m} \quad \text{Deflection from F4} \]

Adding all Deflections:

\[ y_1 := (y_{F1} + y_{F2} + y_{F3} + y_{F4}) \cdot n \]

\[ y_1 = [0.63] \mu \text{m} \quad \text{Deflection at 1} \]
Calculate Deflection at 2 using Superposition

\[ x_2 := a_2 = 0.102 \, \text{m} \]

\[ y_{2F1} := \frac{(F1 \cdot b1) \cdot \left( \frac{L}{b1} \right) \cdot (x_2 - a_1)^3 + (L^2 - b1^2) \cdot x_2 - (x_2)^3}{6 \cdot E \cdot I \cdot L} \]

\[ y_{2F} = (4.26 \cdot 10^{-8}) \, \text{m} \]  
Deflection from F1

\[ y_{2F2} := \frac{(F2 \cdot b2 \cdot x_2) \cdot (L^2 - x_2^2 - b2^2)}{6 \cdot E \cdot I \cdot L} \]

\[ y_{2F2} = [6.77 \cdot 10^{-8}] \, \text{m} \]  
Deflection from F2

\[ y_{2F3} := \frac{(F3 \cdot b3 \cdot x_2) \cdot (L^2 - x_2^2 - b3^2)}{6 \cdot E \cdot I \cdot L} \]

\[ y_{2F3} = [3.65 \cdot 10^{-8}] \, \text{m} \]  
Deflection from F3

\[ y_{2F4} := \frac{(F4 \cdot b4 \cdot x_2) \cdot (L^2 - x_2^2 - b4^2)}{6 \cdot E \cdot I \cdot L} \]

\[ y_{2F4} = [2.96 \cdot 10^{-7}] \, \text{m} \]  
Deflection from F4

Adding all Deflections:

\[ y_2 := (y_{2F1} + y_{2F2} + y_{2F3} + y_{2F4}) \cdot n \]

\[ y_2 = [0.88] \, \mu\text{m} \]  
Deflection at 2
Calculate Deflection at 3 using Superposition

\[ x_3 := a_3 = 0.225 \text{ m} \]

\[ y_{3F1} := \frac{(F1 \cdot b1) \left( \frac{L}{b1} \right) \left( x_3 - a1 \right)^3 + \left( L^2 - b1^2 \right) \cdot x_3 - (x_3)^3}{6 \cdot E \cdot I \cdot L} \]

\[ y_{3F1} = (2.5 \cdot 10^{-8}) \text{ m} \quad \text{Deflection from F1} \]

\[ y_{3F2} := \frac{(F2 \cdot b2) \left( \frac{L}{b2} \right) \left( x_3 - a2 \right)^3 + \left( L^2 - b2^2 \right) \cdot x_3 - (x_3)^3}{6 \cdot E \cdot I \cdot L} \]

\[ y_{3F2} = (4.11 \cdot 10^{-8}) \text{ m} \quad \text{Deflection from F2} \]

\[ y_{3F3} := \frac{\left( F3 \cdot b3 \cdot x3 \right) \left( L^2 - x3^2 - b3^2 \right)}{6 \cdot E \cdot I \cdot L} \]

\[ y_{3F3} = [3.39 \cdot 10^{-8}] \text{ m} \quad \text{Deflection from F3} \]

\[ y_{3F4} := \frac{(F4 \cdot b4) \left( \frac{L}{b4} \right) \left( x_3 - a4 \right)^3 + \left( L^2 - b4^2 \right) \cdot x_3 - (x_3)^3}{6 \cdot E \cdot I \cdot L} \]

\[ y_{3F4} = (2.10 \cdot 10^{-7}) \text{ m} \quad \text{Deflection from F4} \]

Adding all Deflections:

\[ y_3 := (y_{3F1} + y_{3F2} + y_{3F3} + y_{3F4}) \cdot n \]

\[ y_3 = [0.62] \text{ \mu m} \quad \text{Deflection at 3} \]

**RESULTS:** The maximum deflections for each point are \( y_1 = [0.63] \text{ \mu m}, \ y_2 = [0.88] \text{ \mu m}, \) and \( y_3 = [0.62] \text{ \mu m} \) with 0.88 \text{ \mu m} being the maximum deflection.
**FIND:** Deflections at Points 4 and 5 for Gantry Legs

**SCHEMATIC (FBD):**

Load Cases Examined:
- **Case 1:** At Rest
- **Case 2:** Actuator 1 Active (F1 becomes Fa)
- **Case 3:** Actuator 2 Active (F3 becomes Fa)
- **Case 4:** Both Actuators Active (F1 and F3 become Fa)
The load cases of the gantry will be modeled as a simple supports setup as seen in the figure below where each individual loading will be represented by the corresponding equations for force, F, which are shown in the figure below as well. Superposition Method will be used to calculate the deflection at each point by first calculating the individual deflections caused by each individual force at that given point and then adding them up. The sum of these individual deflections from each force will give the total deflection of that given point on the gantry.

6 Simple supports—intermediate load

\[
\begin{align*}
R_1 &= \frac{Fb}{l} \\
R_2 &= \frac{Fa}{l} \\
V_{AB} &= R_1 \\
V_{BC} &= -R_2 \\
M_{AB} &= \frac{Fbx}{l} \\
M_{BC} &= \frac{Fa}{l}(l - x) \\
y_{AB} &= \frac{Fbx}{6EI} \left( x^2 + b^2 - l^2 \right) \\
y_{BC} &= \frac{Fa(l - x)}{6EI} \left( x^2 + a^2 - 2lx \right)
\end{align*}
\]

Source: Shigley's Mechanical Engineering Design 10th ed. Table A-9 p. 1023
SOLUTION:

Calculate Cross-Sectional Area of Gantry Leg

\[ A = 0.04 \, m \cdot 0.08 \, m \]

\[ A = 0.0032 \, m^2 \quad \text{Cross-Sectional Area} \]

Calculate Reactions at 1 and 2 for Case 1

\[ \sum M_1 = 0: \quad F_5(a_5) - F_1(a_1) - F_2(a_2) - F_3(a_3) - F_g(a_4) + R_2(L) - F_6(a_6) = 0 \]

\[ R_2 = \frac{-(F_5 \cdot a_5 - F_1 \cdot a_1 - F_2 \cdot a_2 - F_3 \cdot a_3 - F_g \cdot a_4 - F_6 \cdot a_6)}{L} \]

\[ R_2 = 91.3 \, N \]

\[ \sum F_y = 0: \quad R_1 + R_2 - F_5 - F_1 - F_2 - F_3 - F_g - F_6 = 0 \]

\[ R_1 = -(R_2 - F_5 - F_1 - F_2 - F_3 - F_g - F_6) \]

\[ R_1 = 114.2 \, N \]

Calculate Deflections at 4 and 5 for Case 1

\[ \delta_4 := \frac{R_1 \cdot h}{A \cdot E} \cdot n \]

\[ \delta_4 = 0.043 \, \mu m \quad \text{Deflection at 4 for Case 1} \]

\[ \delta_5 := \frac{R_2 \cdot h}{A \cdot E} \cdot n \]

\[ \delta_5 = 0.034 \, \mu m \quad \text{Deflection at 5 for Case 1} \]

Calculate Reactions at 1 and 2 for Case 2

\[ \sum M_1 = 0: \quad F_5(a_5) + F_a(a_1) - F_2(a_2) - F_3(a_3) - F_g(a_4) + R_2(L) - F_6(a_6) = 0 \]

\[ R_2 := \frac{-(F_5 \cdot a_5 + F_a \cdot a_1 - F_2 \cdot a_2 - F_3 \cdot a_3 - F_g \cdot a_4 - F_6 \cdot a_6)}{L} \]

\[ R_2 = -87.8 \, N \]
\[ \Sigma F_y = 0 : \quad R1 + R2 - F5 + Fa - F2 - F3 - Fg - F6 = 0 \]
\[ R1 := -(R2 - F5 + Fa - F2 - F3 - Fg - F6) \]
\[ R1 = -518.4 \text{ N} \]

Calculate Deflections at 4 and 5 for Case 2

\[ \delta_4 := \frac{R1 \cdot h}{A \cdot E} \cdot n \]
\[ \delta_4 = -0.19 \mu m \] \quad \text{Deflection at 4 for Case 2} \\
\[ \delta_5 := \frac{R2 \cdot h}{A \cdot E} \cdot n \]
\[ \delta_5 = -0.03 \mu m \] \quad \text{Deflection at 5 for Case 2} \\

Calculate Reactions at 1 and 2 for Case 3

\[ \Sigma M_1 = 0 : \quad F5 (a5) - F1 (a1) - F2 (a2) + Fa (a3) - Fg (a4) + R2 (L) - F6 (a6) = 0 \]
\[ R2 := \frac{-(F5 \cdot a5 - F1 \cdot a1 - F2 \cdot a2 + Fa \cdot a3 - Fg \cdot a4 - F6 \cdot a6)}{L} \]
\[ R2 = -538.5 \text{ N} \]
\[ \Sigma F_y = 0 : \quad R1 + R2 - F5 - F1 - F2 + Fa - Fg - F6 = 0 \]
\[ R1 := -(R2 - F5 - F1 - F2 + Fa - Fg - F6) \]
\[ R1 = -67.7 \text{ N} \]

Calculate Deflections at 4 and 5 for Case 3

\[ \delta_4 := \frac{R1 \cdot h}{A \cdot E} \cdot n \]
\[ \delta_4 = -0.03 \mu m \] \quad \text{Deflection at 4 for Case 3} \\
\[ \delta_5 := \frac{R2 \cdot h}{A \cdot E} \cdot n \]
\[ \delta_5 = -0.20 \mu m \] \quad \text{Deflection at 5 for Case 3}
Calculate Reactions at 1 and 2 for Case 4

\[ \Sigma M_1 = 0 : \quad F_5 (a_5) + Fa (a_1) - F_2 (a_2) + Fa (a_3) - F_g (a_4) + R_2 (L) - F_6 (a_6) = 0 \]
\[ R_2 := \frac{- (F_5 \cdot a_5 + Fa \cdot a_1 - F_2 \cdot a_2 + Fa \cdot a_3 - F_g \cdot a_4 - F_6 \cdot a_6)}{L} \]
\[ R_2 = -717.6 \, N \]

\[ \Sigma F_y = 0 : \quad R_1 + R_2 - F_5 + Fa - F_2 + Fa - F_g - F_6 = 0 \]
\[ R_1 := - (R_2 - F_5 + Fa - F_2 + Fa - F_g - F_6) \]
\[ R_1 = -700.3 \, N \]

Calculate Deflections at 4 and 5 for Case 4

\[ \delta_4 := \frac{R_1 \cdot h}{A \cdot E} \cdot n \]
\[ \delta_4 = -0.26 \, \mu m \quad \text{Deflection at 4 for Case 4} \]

\[ \delta_5 := \frac{R_2 \cdot h}{A \cdot E} \cdot n \]
\[ \delta_5 = -0.27 \, \mu m \quad \text{Deflection at 5 for Case 4} \]

**RESULTS:** The maximum deflections for each point are \( \delta_4 = 0.26 \, \mu m \) and \( \delta_5 = 0.27 \, \mu m \).
**FIND:** Endurance Limit of Gantry for infinite life & whether it will experience plastic deformation

**SCHEMATIC (FBD):**

![FBD Diagram]

**SOLUTION:**

From Table A-20 for AISI 1018 CD Steel p.1048

\[ S_{ut} := 440 \text{ MPa} \quad \text{Tensile Strength} \]

\[ S_y := 370 \text{ MPa} \quad \text{Yield Strength} \]

Calculate Cross-Sectional Area of Gantry Top

\[ A := b \cdot t \]

\[ A = 3200 \text{ mm}^2 \]

Check for Yielding

\[ \sigma_{max} := \frac{F_a+F_a+F_1+F_2+F_3+F_4+F_5+F_6}{A} \]

\[ \sigma_{max} = 0.54 \text{ MPa} \]

**RESULT:** Since the max stress, \( \sigma_{max} = 0.54 \text{ MPa} \), is significantly lower than the Yield Strength, \( S_y = 370 \text{ MPa} \), the Gantry should not experience any plastic deformation during operation.
Calculating Endurance Limit for AISI 1018 CD Steel for infinite life

\[ S'_e := 0.5 \cdot S_{ut} \]

\[ S'_e = 220 \text{ MPa} \]

Determining Modifying Factors, K's

\[ k_a := 4.51 \cdot (S_{ut})^{-0.265} \]

For Cold Drawn Steel where \( a = 4.51 \) \& \( b = -0.265 \) from Table 6-2 p.296

\[ k_a := 0.897 \]

\[ d_e := 0.808 \cdot \sqrt{b \cdot t} \]

Equivalent Diameter for Rectangle for calculating \( k_b \)

\[ d_e = 0.046 \text{ m} \]

\[ k_b := 1.24 \cdot (d_e)^{-0.107} \]

For \( 2.79 \text{ mm} < d_e < 51 \text{ mm} \) where \( d_e = 46 \text{ mm} \)

\[ k_b := 1.725 \]

\[ k_c := 1 \]

For Bending

\[ k_d := 1 \]

For Room Temperature

\[ k_e := 1 \]

For Reliability

\[ k_f := 1 \]

For Miscellaneous Effects

Modified Endurance Limit

\[ S_e := k_a \cdot k_b \cdot k_c \cdot k_d \cdot k_e \cdot k_f \cdot S'_e \]

\[ S_e = 340 \text{ MPa} \]

Calculating Alternating Strength using ASME-elliptic Formula

\[ S_a := \sqrt{\frac{S_e^2 \cdot S_y^2}{S_e^2 + S_y^2}} \]

\[ S_a = 251 \text{ MPa} \]

**RESULT:** The max stress, \( \sigma_{max} = 0.54 \text{ MPa} \), of the Gantry does not exceed the Endurance Limit, \( S_e = 340 \text{ MPa} \), nor the Alternating Strength, \( S_a = 251 \text{ MPa} \), of the 1080 CD Steel.
Appendix D4: Tolerance Stack up

Table D4.1. Tolerance stack up results for beam to rail analysis for straightness tolerance requirement as shown in figure D4.1.

<table>
<thead>
<tr>
<th>Part</th>
<th>Dimension</th>
<th>Nominal</th>
<th>±</th>
<th>Tolerance</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam</td>
<td>A</td>
<td>100</td>
<td>±</td>
<td>0.1</td>
<td>mm</td>
</tr>
<tr>
<td>Line Constraint</td>
<td>B</td>
<td>12</td>
<td>±</td>
<td>0.1</td>
<td>mm</td>
</tr>
<tr>
<td>Granite Parallel</td>
<td>C</td>
<td>50</td>
<td>±</td>
<td>0.1</td>
<td>mm</td>
</tr>
<tr>
<td>Rail</td>
<td>D</td>
<td>24</td>
<td>±</td>
<td>0.1</td>
<td>mm</td>
</tr>
</tbody>
</table>

|                  |           |         |    |           |       |
| Upper Limit E:   | 38.4      |         |    |           |       |
| Lower Limit E:   | 37.6      |         |    |           |       |
| E                | 38        | ± 0.4   | mm |           |

Figure D4.1. Schematic of tolerance stack up for beam to rail analysis showing beam (A ± a), line constraint (B ± b), granite parallel (C ± c), and rail (D ± d).
Table D4.2. Tolerance stack up results for rail to rail analysis for parallelism tolerance requirement as shown in figure D4.2.

<table>
<thead>
<tr>
<th>Part</th>
<th>Dimension</th>
<th>Nominal</th>
<th>±</th>
<th>Tolerance</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rail</td>
<td>A</td>
<td>24</td>
<td>±</td>
<td>0.1</td>
<td>mm</td>
</tr>
<tr>
<td>Granite Parallel</td>
<td>B</td>
<td>50</td>
<td>±</td>
<td>0.1</td>
<td>mm</td>
</tr>
<tr>
<td>Granite Parallel</td>
<td>C</td>
<td>50</td>
<td>±</td>
<td>0.1</td>
<td>mm</td>
</tr>
<tr>
<td>Rail</td>
<td>D</td>
<td>24</td>
<td>±</td>
<td>0.1</td>
<td>mm</td>
</tr>
</tbody>
</table>

|            | Upper Limit E: | 0.4 |
|            | Lower Limit E: | -0.4 |
| E           | ± 0.4         | mm  |

Figure D4.2. Schematic of tolerance stack up for beam to rail analysis showing rail (A ± a), granite parallel (B ± b), granite parallel (C ± c), and rail (D ± d).
Appendix D5: X-Beam Leveling Actuator
This section entails the calculations for the force requirement of the actuator needed to level the X-Beam. It is assumed the actuator force acts and the line constraint are at extreme opposites of the X-Beam which signifies the worst case scenario. The analysis will be based off of Figure D5.1.

To summarize, the actuator for leveling the X-Beam to within a tenth of a micrometer must have a force output of 167 N, a stroke length of 5mm, and a minimum resolution of 0.1 micrometer.

![Figure D5.1. X-Beam Leveling Actuator Calculation Diagram. X-Beam side view with marked positions at the ends of the linear encoder strips at point A and B.](image)

With the dimensions indicated above in figure D5.1, we can calculate the stroke length needed by checking the stroke length required to adjust point B by some arbitrary amount. This is done by comparing the distance moved by point B multiplied by the fraction of length $b/l$ as shown by equation D5.1 where $x$ is the vertical distance you want to adjust point B by. On the next page in Table D5.1 is an example calculation by excel.

$$\frac{b}{l} * x$$  

**eq D5.1**
Additionally, we calculated the resolution achieved at point A and point B in the same way as we calculated the stroke length necessary using equation D5.1 where \( x \) represents the resolution of the error and dimension \( b \) is substituted with dimension \( a \) when calculating the resolution at point A.

**Table D5.1: Calculation for the stroke length needed.**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>If point B needs to be adjusted by</td>
<td>5 mm</td>
</tr>
<tr>
<td>and beam length is</td>
<td>895 mm</td>
</tr>
<tr>
<td>length ( a ) is</td>
<td>35 mm</td>
</tr>
<tr>
<td>length ( b ) is</td>
<td>860 mm</td>
</tr>
<tr>
<td>Actuator needs stroke Length</td>
<td>5 mm</td>
</tr>
<tr>
<td>Resolution of Actuator</td>
<td>0.1 micrometer</td>
</tr>
<tr>
<td>Projected resolution at point A</td>
<td>0.00391 micrometer</td>
</tr>
<tr>
<td>Projected resolution at point B</td>
<td>0.096 micrometer</td>
</tr>
</tbody>
</table>

After calculating the stroke length and resolution needed for the actuator, we calculated the necessary force output of the actuator which was done by equating the moment generated by the weight of the X-Beam to the moment generated by the actuator summed around the fixed pin at the bottom left of the diagram in Figure D5.1. A sample excel calculation is shown below in Table D5.2.

**Table D5.2. Calculation of force output necessary to level the X-Beam**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam Mass</td>
<td>75 lbs</td>
</tr>
<tr>
<td>Conversion</td>
<td>0.45 kg/lbs</td>
</tr>
<tr>
<td>Beam Mass</td>
<td>34 kg</td>
</tr>
<tr>
<td>Beam Weight</td>
<td>334 N</td>
</tr>
<tr>
<td>Moment Arm</td>
<td>149 Nm</td>
</tr>
<tr>
<td>Actuator Moment</td>
<td>149 Nm</td>
</tr>
<tr>
<td>Actuator Force Output</td>
<td>167 N</td>
</tr>
</tbody>
</table>
Appendix F: List of Subassembly and Parts Drawings

Appendix F1: Full Assembly ................................................................. 94
Appendix F2: Granite Subassembly ....................................................... 95
Appendix F3: Housing Subassembly ....................................................... 96
Appendix F4: Beam Actuator Subassembly ......................................... 97
Appendix F5: Gantry Subassembly ......................................................... 98
Appendix F6: Gantry Actuator Subassembly ......................................... 99
Appendix F7: Screwdriver Actuator Subassembly ................................. 100
Appendix F8: Bearing Attachment Plate Configuration 1 ....................... 101
Appendix F9: Bearing Attachment Plate Configuration 2 ....................... 102
Appendix F10: Solenoid Pullout Pin ...................................................... 103
Appendix F11: Electronics Housing ...................................................... 104
Appendix F12: Gantry Leg Configuration 1 ............................................ 105
Appendix F13: Gantry Leg Configuration 2 ............................................ 107
Appendix F14: Gantry Probe Covering .................................................. 109
Appendix F15: Gantry Top ................................................................. 110
Appendix F16: Gearbox Housing ......................................................... 113
Appendix F17: Granite Parallel Gauge Block Constraints ....................... 114
Appendix F18: Granite Plate .............................................................. 115
Appendix F19: Hardstop ................................................................. 118
Appendix F20: Leadscrew Raiser ......................................................... 119
Appendix F21: Line Constraint .......................................................... 120
Appendix F22: Leadscrew Raiser Drawing ............................................ 121
<table>
<thead>
<tr>
<th>ITEM NO.</th>
<th>PART NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GRANITE SUBASSEMBLY</td>
</tr>
<tr>
<td>2</td>
<td>HOUSING SUBASSEMBLY</td>
</tr>
<tr>
<td>3</td>
<td>BEAM ACTUATOR ASSEMBLY</td>
</tr>
<tr>
<td>4</td>
<td>GANTRY ASSEMBLY</td>
</tr>
<tr>
<td>5</td>
<td>GANTRY ACTUATOR ASSEMBLY</td>
</tr>
<tr>
<td>6</td>
<td>SCREWDRIVER ACTUATOR ASSEMBLY</td>
</tr>
</tbody>
</table>

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM

X±0,5
X±1°
XXX±0,1 X±0,5" INTERNAL -6H

ID NUMBER: 934FA005

TITLE
FULL ASSEMBLY

SOLIDWORKS Educational Product. For Instructional Use Only
# Appendix F2: Granite Subassembly

## Table of Parts

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Part Number</th>
<th>Description</th>
<th>QTY.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>934FA014</td>
<td>Granite Plate</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>934FA013</td>
<td>Granite Parallel Constraints</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>SCB12-50_2_03</td>
<td>M12 Bolts</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>SCB4-10_2_03</td>
<td>M4 Bolts</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>934FA016</td>
<td>Hard Stop</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>LWLF24C1R10008CST1P</td>
<td>Linear Rails</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>934FA021</td>
<td>Line Constraint</td>
<td>1</td>
</tr>
</tbody>
</table>

**Dimensions:**
- Unless otherwise specified, dimensions are all in mm.
- \(X \pm 0.5\) mm
- \(X \pm 0.3\) mm
- \(X \pm 0.5^\circ\) mm
- \(X \pm 1^\circ\) mm
- External -5G, 6G
- Internal -6H

**Materials:**
- Refer to BOM unless otherwise specified.

**ID Number:** C934FA015

**Title:** Granite Subassembly

**Drawn by:** Joseph Falcao

**Scale:** 1:5

**Cal Poly Mechanical Engineering**
Appendix F3: Housing Subassembly

<table>
<thead>
<tr>
<th>ITEM NO.</th>
<th>PART NUMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HOUSING</td>
<td>ELECTRONICS HOUSING AND PLATE</td>
</tr>
<tr>
<td>2</td>
<td>ESP10-600-24</td>
<td>POWER SUPPLY</td>
</tr>
<tr>
<td>3</td>
<td>GH7028-ND</td>
<td>RELAY BOARD AND RELAYS</td>
</tr>
<tr>
<td>4</td>
<td>317436-02</td>
<td>SG60M HEIDENHAIN SWITCHBOX</td>
</tr>
<tr>
<td>5</td>
<td>SC84-10_2_03</td>
<td>M4 SCREWS</td>
</tr>
<tr>
<td>6</td>
<td>S36397-01</td>
<td>HEIDENHAIN INTERPOLATION ELECTRONICS</td>
</tr>
<tr>
<td>7</td>
<td>2AP3</td>
<td>EMERGENCY STOP AND RUN BUTTONS</td>
</tr>
<tr>
<td>8</td>
<td>2AS2-3</td>
<td>3 POSITION SWITCH</td>
</tr>
<tr>
<td>9</td>
<td>350-3981-ND</td>
<td>INDICATOR LEDS</td>
</tr>
<tr>
<td>10</td>
<td>STM32 NUCLEO 476RG</td>
<td>MICROCONTROLLER</td>
</tr>
<tr>
<td>11</td>
<td>MSPSS6_2_03</td>
<td>LIMIT SWITCH</td>
</tr>
<tr>
<td>12</td>
<td>934FA018</td>
<td>EXTERNAL HOUSING</td>
</tr>
<tr>
<td>13</td>
<td>HousingPlate</td>
<td>HOUSING ACCESS PANEL</td>
</tr>
<tr>
<td>14</td>
<td>HousingCableConnection</td>
<td>HOUSING CABLE CONNECTION</td>
</tr>
</tbody>
</table>

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0,5        ANGLE: THREADS:
X±0,3 X±1°          EXTERNAL -5G 6G
X±0,1 X±0,5° INTERNAL -6H

MATERIAL: STEEL AND ALUMINUM
FINISH: NONE
SCALE: 1:4

DATE: 01-14-2017
DRAWN BY: TEAM X
TITLE: HOUSING AND ELECTRONICS

SOLIDWORKS Educational Product. For Instructional Use Only
## Appendix F5: Gantry Subassembly

<table>
<thead>
<tr>
<th>ITEM NO.</th>
<th>PART NUMBER</th>
<th>DESCRIPTION</th>
<th>QTY.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>934FA001</td>
<td>BEARING ATTACHMENT PLATE LEADScrew -STEEL</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>934FA002</td>
<td>BEARING ATTACHMENT PLATE -STEEL</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>934FA004</td>
<td>GANTRY LEG CALIBRATION - STEEL</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>934FA007</td>
<td>GANTRY LEG NUT - STEEL</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>934FA010</td>
<td>GANTRY TOP - STEEL</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>linear_actuator_57H4 A-3.25-B15</td>
<td>LINEAR ACTUATOR</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>_MLF24Slide_4</td>
<td>LINEAR RAILS AND BEARINGS MLF24</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>MT60</td>
<td>HEIDENHAIN PROBE MT-60M</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>SCB3-10_2_03</td>
<td>M3 BOLTS FOR METAL STOCK</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>SCB4-10_2_03</td>
<td>M4 BOLTS FOR METAL STOCK</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>SCB4-40_2_03</td>
<td>M4 BOLTS FOR PROBE</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>SCB5-20_2_03</td>
<td>M5 BOLTS FOR ACTUATORS</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>SCB12-50_2_03</td>
<td>M4 BOLTS TO ATTACH METAL STOCK TO GANTRY FRAME</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>RGOCG14-50-MC6_2_03</td>
<td>PRECISION RODS - 1045 CARBON STEEL</td>
<td>2</td>
</tr>
</tbody>
</table>

**ID NUMBER:** C934FA009

**MATERIAL:** REFER TO BOM

**FINISH:** NONE

**DRAWN BY:** JOSEPH FALCAO

**SCALE:** 1:4

**DIMENSIONS ARE ALL IN MM**

**X±0.5, X±1°, EXTERNAL -5G,6G**

**XX±0.1, X±0.5°, INTERNAL -6H**

**UNLESS OTHERWISE SPECIFIED**

**FINISH:** NONE

**MATERIAL:** REFER TO BOM

**SCALE:** 1:4

**DRAWN BY:** JOSEPH FALCAO

**DATE:** 01-14-2017

**TITLE:** GANTRY SUBASSEMBLY

**Cal Poly Mechanical Engineering**

**FINISH:** NONE

**SCALE:** 1:4

**DRAWN BY:** JOSEPH FALCAO

**DATE:** 01-14-2017

**TITLE:** GANTRY SUBASSEMBLY
### Appendix F6: Gantry Actuator Subassembly

<table>
<thead>
<tr>
<th>ITEM NO.</th>
<th>PART NUMBER</th>
<th>DESCRIPTION</th>
<th>QTY.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SCB5-20_2_03</td>
<td>M5-20 Stainless Steel Socket Head Bolt</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>NEMA23</td>
<td>NEMA23 Stepper Motor</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>MTSRW12-970-F20-V8-S13-Q6_2_03</td>
<td>Double Stepped Leadscrew</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>MTSNR12_2_03</td>
<td>Anti-Backlash Leadscrew Nut</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Leadscrew Raiser</td>
<td>Leadscrew Support Raiser</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>MTUZ8_2_03</td>
<td>Leadscrew End Support</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>SCB4-10_2_03</td>
<td>M4-10 Stainless Steel Socket Head Bolt</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>CPLSC20-6-6_35_2_03</td>
<td>Flexible Slit Shaft Coupling</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>SCB6-45_2_03</td>
<td>M6-45 Stainless Steel Socket Head Bolt</td>
<td>2</td>
</tr>
</tbody>
</table>

**DIMENSIONS ARE ALL IN MM**
- X±0.5
- ANGLE: X±1°
- THREADS: X,XX±0.3, X±0.3°, X±0.5°, EXTERNAL -5G, 6G, INTERNAL -6H

**ID NUMBER:** C934FA011

**TITLE:** GANTRY ACTUATOR

**DRAWN BY:** Robert Tam

**SCALE:** 1:4
<table>
<thead>
<tr>
<th>ITEM NO.</th>
<th>PART NUMBER</th>
<th>DESCRIPTION</th>
<th>QTY.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BSJM10-100-F5-E15-P5-Q8_b</td>
<td>BALL SPLINE</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>BSSS10_nr</td>
<td>BALL SPLINE NUT</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>GEABB1.0-48-6-B-21_b</td>
<td>SPUR GEAR</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>HBR8x</td>
<td>BEARING BLOCK</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>GM92345023-R1</td>
<td>DC TORQUING MOTOR</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>GEABB1.0-20-6-B-6.35_b</td>
<td>SPUR GEAR</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>BLUZS-SUD-A110-B80-T3-H20-V90</td>
<td>GEAR HOUSING</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>MCOG17-5-8</td>
<td>FLEXIBLE SHAFT COUPLING</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>20 INCH LBS MINIATURE TORQUE LIMITER</td>
<td>TORQUE LIMITER</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>30TF09</td>
<td>BALL END HEX BIT</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>FNCLCH-V21_0-D25-L18_0_2_03</td>
<td>SHAFT COLLAR</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>74097841</td>
<td>SOLENOID</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>FL6805ZZ</td>
<td>FLANGED BALL BEARINGS</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>Steel Shaft</td>
<td>SOLENOID ACTUATION POINT</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>UF8-15</td>
<td>COMPRESSION SPRINGS</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>SCB4-10</td>
<td>M4 SOCKET CAP SCREW</td>
<td>2</td>
</tr>
</tbody>
</table>

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5 ANGLE: THREADS:
X±0.3 X±1° EXTERNAL -5G 6G
XX±0.1 XX±0.5° INTERNAL -6H

ID NUMBER: 934FA020
TITLE: SCREWDRIVER ACTUATOR SUBASSEMBLY

MATERIAL: REFER TO BOM
FINISH: NONE
SCALE: 1:2
DRAWN BY: TEAM X

Cal Poly
Mechanical Engineering
Appendix F8: Bearing Attachment Plate Configuration 1

NOTES: (UNLESS OTHERWISE SPECIFIED)
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 1.0 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5
XX±0.1
X±1°
EXTERNAL -5G 6G
XX±0.5°
INTERNAL -6H

ANGLE: THREADS:

FINISH: POWDER COAT BLACK SEMI-GLOSS NO TEXTURE CARDINAL #BK05 EX MASKING SEE DRAWING

MATERIAL: AISI 1018 STEEL COLD DRAWN

SOLIDWORKS Educational Product. For Instructional Use Only
NOTES: (UNLESS OTHERWISE SPECIFIED)
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 1.0 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5  X±0.3 ±0.1°  ±0.5°
X.X±0.5  X.X±0.5°  INTERNAL -6H

INSTALL:  POWDER COAT BLACK SEMI-GLOSS NO TEXTURE CARDINAL #BK05 EX MASKING SEE DRAWING

MATERIAL:  AISI 1018 STEEL COLD DRAWN
FINISH:  POWDER COAT BLACK SEMI-GLOSS NO TEXTURE CARDINAL #BK05 EX MASKING SEE DRAWING

ID NUMBER:  934FA002

TITLE:  BEARING ATTACHMENT PLATE

REV.  DESCRIPTION
A  PROTOTYPE RELEASE

DATE  01-14-2017

DRAWN BY:  JOSEPH FALCAO
ID NUMBER:  934FA002

SCALE:  2:1

DATE:  01-14-2017
DRAWN BY:  JOSEPH FALCAO
ID NUMBER:  934FA002

SOLIDWORKS Educational Product. For Instructional Use Only
SOLNOID PULL PIN

MATERIAL: CARBON STEEL

FINISH: BLACK OXIDE

SCALE: 8:1

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5 \hspace{1cm} X±0.3 \hspace{1cm} X±1° \hspace{1cm} EXTERNAL -5G 6G
X.X±0.1 \hspace{1cm} X.X±0.5° \hspace{1cm} INTERNAL -6H

DATE: 01-14-2017

DRAWN BY: TEAM X

TITLE: SOLENOID PULL PIN

REV: A

DESCRIPTION: PROTOTYPE RELEASE

PREPARED FOR: SOLIDWORKS Educational Product. For Instructional Use Only
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE ALL IN MM

±0.5  ANGLE: THREADS:
X,X±0.3  X±1°  EXTERNAL -5G,6G
X.X±0.1  X,X±0.5°  INTERNAL -6H

ID NUMBER: 934FA018

TITLE: MAIN HOUSING

MATERIAL: ALUMINUM 5052
FINISH: NONE
SCALE: 1:3

Cal Poly
Mechanical Engineering

DRAWN BY: TEAM X

PROTOTYPE RELEASE
01-14-2017

DATE
DRAWN BY
ID NUMBER
TITLE
MATERIAL
FINISH
SCALE
104
Appendix F12: Gantry Leg Configuration 1

NOTES: (UNLESS OTHERWISE SPECIFIED)
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 1.0 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX
4. THREAD PER ISO 261

SECTION A-A

| ID NUMBER: | 934FA006 |
| UNLESS OTHERWISE SPECIFIED |
| DIMENSIONS ARE ALL IN MM |
| X±0.5 |
| X±1° |
| X±0.3 |
| X±0.5° |
| external -5G 6G |
| internal -6H |
| MATERIAL: AISI 1018 STEEL COLD DRAWN |
| FINISH: POWDER COAT BLACK SEMI-GLOSS NO TEXTURE CARDINAL #BK05 EX MASKING SEE DRAWING |
| SCALE: 1:2 |
| DRAWN BY: JOSEPH FALCAO |
| TITLE: GANTRY LEG CALIBRATION |

SOLIDWORKS Educational Product. For Instructional Use Only
UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5     ANGLE:    THREADS:
X.X±0.3     X±1°     EXTERNAL -5G 6G
X.X±0.1     X.X±0.5°   INTERNAL -6H

MASK ALL SHADED AREAS

ID NUMBER:
934FA006

TITLE
GANTRY LEG CALIBRATION

MATERIAL:
AISI 1018 STEEL COLD DRAWN

FINISH:
POWDER COAT BLACK SEMI-GLOSS NO TEXTURE CARDINAL #BK05 EX MASKING SEE DRAWING

SCALE: 1:2

SOLIDWORKS Educational Product. For Instructional Use Only
NOTES: (UNLESS OTHERWISE SPECIFIED)
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 1.0 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX
4. THREAD PER ISO 261

SECTION A-A
**BACKSIDE**

**MASK ALL SHADED AREAS**

**DIMENSIONS ARE ALL IN MM**

- $x \pm 0.5$
- $x, x \pm 0.3$
- $x, x \pm 0.5^\circ$
- $x, x \pm 0.1$
- $x, x \pm 1^\circ$
- $x, x \pm 0.5^\circ$
- $xx \pm 0.1$
- $xx \pm 0.5^\circ$

**THREADS:**
- EXTERNAL -5G 6G
- INTERNAL -6H

**MATERIAL:**
- AISI 1018 STEEL COLD DRAWN

**FINISH:**
- POWDER COAT BLACK SEMI-GLOSS NO TEXTURE CARDINAL #BK05 EX MASKING SEE DRAWING

**SOLIDWORKS Educational Product. For Instructional Use Only**
Appendix F14: Gantry Probe Covering

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5  ANGLE:  THREADS:  EXTERNAL -5G 6G
X±1°  X±0.1  INTERNAL -6H
X±0.5°

ID NUMBER:  394FA008
TITLE  PROBE HOUSING
MATERIAL:  
FINISH:  NONE
SCALE:  1:1
NOTES: [UNLESS OTHERWISE SPECIFIED]
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 0.5 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX
4. THREAD PER ISO 261

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5 ANGLE: THREADS: X±0.3 X±1° EXTERNAL -5G 6G
X±0.3 X±0.5° INTERNAL -6H

MATERIAL: AISI 1018 STEEL COLD DRAWN
FINISH: POWDER COAT BLACK SEMI-GLOSS NO TEXTURE CARDINAL #BK05 EX MASKING SEE DRAWING

SOLIDWORKS Educational Product. For Instructional Use Only

ID NUMBER: 934FA010
TITLE: GANTRY TOP

DRAWN BY: JOSEPH FALCAO
DATE: 01-14-2017
NOTES: (UNLESS OTHERWISE SPECIFIED)
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 0.5 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX
4. THREAD PER ISO 261

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM

FINISH: POWDER COAT BLACK SEMI-GLOSS NO TEXTURE CARDINAL #BK05 EX MASKING SEE DRAWING

SOLIDWORKS Educational Product. For Instructional Use Only
UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5
X.X±0.3
X.X±0.1
ANGLE:
EXTERNAL -5G 6G
INTERNAL -6H

MATERIAL:
AISI 1018 STEEL COLD DRAWN

ID NUMBER:
934FA010

FINISH:
POWDER COAT BLACK SEMI-GLOSS NO TEXTURE CARDINAL #BK05 EX MASKING SEE DRAWING

SCALE: 1:2

TITLE:
GANTRY TOP

SOLIDWORKS Educational Product. For Instructional Use Only
NOTES: (UNLESS OTHERWISE SPECIFIED)
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 1.0 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX
4. THREAD PER ISO 261

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5 X±0.3 X±1° X±0.05°  EXTERNAL: -5G,6G  INTERNAL:  -6H

ID NUMBER: 934FA046
MATERIAL: GRANITE_JINAN_BLACK, GRADE_0_JB/T7975-1999
FINISH: NONE

DRAWN BY: JOSEPH FALCAO
CALE Poly
Mechanical Engineering
SCALE: 1:3
Appendix F18: Granite Plate

NOTES: (UNLESS OTHERWISE SPECIFIED)
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 1.0 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX
4. THREAD PER ISO 261
5. POLISH DATUMS B AND C

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5
XX±0.3
X±1°
EXTERNAL -5G 6G
XXX±0.1
XX±0.5°
INTERNAL -6H

ID NUMBER: 934FA045
TITLE: GRANITE PLATE
MATERIAL: GRANITE_JINAN_BLACK, GRADE_0_JB/T7975-1999
FINISH: NONE
DRAWN BY: JOSEPH FALCAO
 SCALE: 1:5
DATE: 03-06-2017

GRANITE PLATE
SOLIDWORKS Educational Product. For Instructional Use Only
NOTES: (UNLESS OTHERWISE SPECIFIED)
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 1.0 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX
4. THREAD PER ISO 261
5. POLISH DATUMS B AND C

PLACEMENT OF RUBBER PADS TO REST THE GRANITE

4X INSERT FOR M8X1.25 \( \pm \) 16.0

\( \odot 0.3 \ ) A B C

FINISH: NONE
MATERIAL: GRANITE_JINAN_BLACK, GRADE_0_JB/T7975-1999
DRAWN BY: JOSEPH FALCAO
SCALE: 1:5

ID NUMBER: 934FA045

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X\( \pm 0.5 \)
X.X\( \pm 0.3 \)
X.XX\( \pm 0.1 \)
ANGLE: THREADS:
X.X\( \pm 1^{\circ} \)
EXTERNAL -5G 6G
X.XXX\( \pm 0.5^{\circ} \)
INTERNAL -6H

DATE: 03-06-2017
DRAWN BY: JOSEPH FALCAO
SCALE: 1:5

DESCRIPTION
PROTOTYPE RELEASE
REV.
B
REMARKS
SOLIDWORKS Educational Product. For Instructional Use Only
NOTES: (UNLESS OTHERWISE SPECIFIED)
1. GENERAL SURFACE FINISH 1.6 MICRONS RMS
2. BREAK ALL EDGES 1.0 MAX
3. RADIUS ALL INTERNAL CORNERS 0.5 MAX
4. THREAD PER ISO 261
5. POLISH DATUMS B AND C

MINIMUM AREA SHOWN

FINISH:  NONE
MATERIAL: GRANITE_JINAN_BLACK, GRADE_0_JB/T7975-1999
DIMENSIONS ARE ALL IN MM

X±0.5
X,X±0.3
X,X±0.1
X±1°
X,X±0.5°
XXX±0.1
XXX±0.5°
INTERNAL -5G 6G
INTERNAL -6H

UNLESS OTHERWISE SPECIFIED

DATE
03-06-2017

REV. DESCRIPTION
B PROTOTYPE RELEASE

DRAWN BY: JOSEPH FALCAO
SCALE: 1:5
SHEET: 3/3

CAL POLY
MECHANICAL ENGINEERING

SOLIDWORKS Educational Product. For Instructional Use Only
2X Ø 4.50 THRU ALL
Ø 8 \(\pm 0.3\)
Ø 8.05 X 90°, NEAR SIDE

M4x0.7 Tapped Hole

NOTES (UNLESS OTHERWISE SPECIFIED)
ALL HOLES Ø 0.3 A B C

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE ALL IN MM
X±0.5 ANGLE: THREADS:
X±0.3 X±1° EXTERNAL -5G,6G
XX±0.1 X±0.5° INTERNAL -6H

MATERIAL: ALUMINIUM 6061
FINISH: NONE
SCALE: 2:1

934FA016
Hardstop

SOLIDWORKS Educational Product. For Instructional Use Only
NOTES (UNLESS OTHERWISE SPECIFIED)

ALL HOLES $\varnothing 0.3$ A B C

DIMENSIONS ARE ALL IN MM

UNLESS OTHERWISE SPECIFIED

MATERIAL: ALUMINIUM 6061
FINISH: NONE
SCALE: 1:1
SHEET: 1/3
REV.: A
DESCRIPTION: PROTOTYPE RELEASE
DATE: 01-14-2017

TITLE: LEADScrew RAISER

ID NUMBER: 934FA019

DATE
DRAWN BY: ROBERT TAM
ID NUMBER:
SCALE: 1:1
SHEET:

FINISH: NONE

MATERIAL: ALUMINIUM 6061

X±0.5 ANGLE: THREADS:
X±0.3 X±1° EXTERNAL -5G,6G
X±0.1 X±0.5° INTERNAL -6H
NOTES (UNLESS OTHERWISE SPECIFIED)

ALL HOLES  \( \Phi 0.3 \) A B C

2 x \( \Phi 4.50 \) THRU ALL

\( \Phi 8 \) \( \Phi 4 \)

\( \Phi 8.05 \) X 90°, Near Side
### Appendix G: Manufacturing Plan

**Table G1: List of Manufactured Parts**

<table>
<thead>
<tr>
<th>Manufacturing Parts</th>
<th>Stock Material</th>
<th>Material Cost</th>
<th>Part Size Description</th>
<th>Made by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gantry Top</td>
<td>CD Mild Steel Rectangle 1018</td>
<td>$264.26</td>
<td>40mm x 80mm x 340mm</td>
<td>Micro-Vu</td>
</tr>
<tr>
<td>Gantry Leg Calibration</td>
<td>-Part of Steel Order</td>
<td>-</td>
<td>40mm x 80mm x 110mm</td>
<td>Micro-Vu</td>
</tr>
<tr>
<td>Gantry Leg Nut</td>
<td>-Part of Steel Order</td>
<td>-</td>
<td>40mm x 80mm x 110mm</td>
<td>Micro-Vu</td>
</tr>
<tr>
<td>Bearing Attachment Plate</td>
<td>-Part of Steel Order</td>
<td>-</td>
<td>10mm x 40mm x 80mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Bearing Attachment Plate</td>
<td>-Part of Steel Order</td>
<td>-</td>
<td>10mm x 40mm x 150mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Leadscrew</td>
<td>Aluminum Rod 6061</td>
<td>$22.55</td>
<td>Ø 25mm x 130mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Hard Stop</td>
<td>Aluminum Rectangle 6061</td>
<td>$21.43</td>
<td>11mm x 50mm x 50mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Lead Screw Support Raiser</td>
<td>Aluminum Rectangle 6061</td>
<td>$21.43</td>
<td>30mm x 68.7mm x 80mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Granite Plate</td>
<td>Granite</td>
<td>$550</td>
<td>100mm x 400mm x 1050mm</td>
<td>Granite Supplier</td>
</tr>
<tr>
<td>Granite Parallel (1)</td>
<td>Granite</td>
<td>$370</td>
<td>50mm x 50mm x 800mm</td>
<td>Granite Supplier</td>
</tr>
<tr>
<td>Granite Parallel (2)</td>
<td>Granite</td>
<td>$370</td>
<td>50mm x 50mm x 800mm</td>
<td>Granite Supplier</td>
</tr>
<tr>
<td>Housing Main</td>
<td>0.09” Aluminum Sheet 5052</td>
<td>$117</td>
<td>549.34mm x 797.63mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Housing Middle Plate</td>
<td>- Part of Aluminum Sheet Order</td>
<td>-</td>
<td>240mm x 370mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Housing Back Plate</td>
<td>- Part of Aluminum Sheet Order</td>
<td>-</td>
<td>266.76mm x 393.05mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Cable Bracket</td>
<td>- Part of Aluminum Sheet Order</td>
<td>-</td>
<td>143.05mm x 812.8mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Screwdriver Housing (1)</td>
<td>- Part of Aluminum Sheet Order</td>
<td>-</td>
<td>306.14mm x 335.07mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Screwdriver Housing (2)</td>
<td>- Part of Aluminum Sheet Order</td>
<td>-</td>
<td>306.14mm x 335.07mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Touch Probe Housing</td>
<td>- Part of Aluminum Sheet Order</td>
<td>-</td>
<td>162.84mm x 235mm</td>
<td>Team X</td>
</tr>
<tr>
<td>Drill Gear Box Cover Holes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Team X</td>
</tr>
<tr>
<td>Apply taps where needed</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Team X</td>
</tr>
<tr>
<td><strong>Total Manufacturing Costs</strong></td>
<td></td>
<td></td>
<td><strong>$1736.67</strong></td>
<td></td>
</tr>
</tbody>
</table>
Appendix H: List of Part Job Planners

Appendix H1: Bearing Attachment Plate Configuration 1 ............................................. 124
Appendix H2: Bearing Attachment Plate Configuration 2 ............................................. 125
Appendix H3: Gantry Leg Configuration 1 ................................................................. 126
Appendix H4: Gantry Leg Configuration 2 ................................................................. 127
Appendix H5: Gantry Top ...................................................................................... 128
Appendix H6: Hardstop ......................................................................................... 129
Appendix H7: Lead Screw Support Raiser ............................................................... 130
Appendix H8: Line Constraint .............................................................................. 131
<table>
<thead>
<tr>
<th>OP #</th>
<th>Operation Description</th>
<th>Machine Tool Cell</th>
<th>Tooling &amp; Fixtures Required</th>
<th>OP Time (min)</th>
<th>Approval</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Cut Stock to Length &amp; Deburr</td>
<td>Horizontal Band Saw</td>
<td>Dial Caliper File</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Milling Operation #1</td>
<td>Haas Mill</td>
<td>Milling Vise, Parallels, 75 mm Face Mill, 13 mm Flat End Mill, 3.4 mm Drill Bit for Ø 3.4 mm Holes, 6.5 mm Drill Bit for Ø 6.5 mm Counterbores, 13 mm 45° Chamfer, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Operation #1 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td>Optical Comparater</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Milling Operation #2</td>
<td>Haas Mill</td>
<td>Milling Vise, Parallels, 75 mm Face Mill, 13 mm Flat End Mill, 4.5 mm Drill Bit for Ø 4.5 mm Holes, 8 mm Drill Bit for Ø 8 mm Counterbores, 13 mm 45° Chamfer, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Operation #2 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td>Drop Height Indicator, Optical Comparater</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>Clean &amp; Polish</td>
<td>Finishing Table</td>
<td>Mothers Polish, Rag</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>
### PART ROUTING / JOB PLANNER

**NAME:** Joseph Falcao  
**PART:** Bearing Attachment Plate  
**DRAWING:** OP#1 FAIR Print  
**MATERIAL:** 10mm x 40mm x 80mm Cold Finish Mild Steel Square 1018  

**NOTES:** Deburr all edges after every operation.

<table>
<thead>
<tr>
<th>OP #</th>
<th>Operation Description</th>
<th>Machine Tool</th>
<th>Cell</th>
<th>Tooling &amp; Fixtures Required</th>
<th>OP Time (min)</th>
<th>Approval</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Cut Stock to Length &amp; Deburr</td>
<td>Horizontal Band Saw</td>
<td></td>
<td>Dial Caliper File</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>
| 20   | Milling Operation #1         | Haas Mill    |            | Milling Vise  
Parallels  
75 mm Face Mill  
13 mm Flat End Mill  
3.4 mm Drill Bit for the Ø 3.4 mm Holes  
6.5 mm Drill Bit for the Ø 6.5 mm Counterbores  
13 mm 45° Chamfer  
Mechanical Edge Finder | 20.0          |          |
| 30   | Operation #1 F.A.I.R Inspection | Metrology Lab |        | Optical Comparator                                                                    | 10.0          |          |
| 40   | Milling Operation #2         | Haas Mill    |            | Milling Vise  
Parallels  
75 mm Face Mill  
13 mm Flat End Mill  
4.5 mm Drill Bit for the Ø 4.5 mm Holes  
8 mm Drill Bit for the Ø 8 mm Counterbores  
13 mm 45° Chamfer  
Mechanical Edge Finder | 20.0          |          |
| 60   | Operation #2 F.A.I.R Inspection | Metrology Lab |        | Drop Height Indicator  
Optical Comparator                                      | 10.0          |          |
| 70   | Clean & Polish              | Finishing Table |          | Mothers Polish  
Rag                                      | 5.0           |          |
## PART ROUTING / JOB PLANNER

| NAME              | Joseph Falcao |
| PART              | Gantry Leg Calibration |
| DRAWING           | OP1 FAIR Print |
| MATERIAL          | 40mm x 80mm x 120mm Cold Finish Mild Steel Square 1018 |

NOTES: Deburr all edges after every operation.

### OP # | Operation Description | Machine Tool Cell | Tooling & Fixtures Required | OP Time (min) | Approval |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Cut Stock to Length &amp; Deburr</td>
<td>Horizontal Band Saw</td>
<td>Dial Caliper, File</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Milling Operation #1</td>
<td>Haas Mill</td>
<td>Milling Vise, Parallels, 75 mm Face Mill, 13 mm Flat End Mill, 3.3 mm Tap Drill Bit for the M4X0.7 Holes, 13 mm 45° Chamfer, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Operation #1 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td>Optical Comparator</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Milling Operation #2</td>
<td>Haas Mill</td>
<td>Milling Vise, Parallels, Soft Jaw Set, 75 mm Face Mill, 15 mm End Mill, 13 mm 45° Chamfer, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Drill Press Operation #1</td>
<td>Drill Press</td>
<td>Drill Press Vise, 10.8 mm Tap Drill Bit for the M12X1.25 Holes</td>
<td>15.0</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Drill Press Operation #2</td>
<td>Drill Press</td>
<td>Drill Press Vise, 3.3 mm Tap Drill Bit for the M4X0.7 Holes</td>
<td>15.0</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>Operation #2 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td>Drop Height Indicator, Optical Comparator</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Clean &amp; Polish</td>
<td>Finishing Table</td>
<td>Mothers Polish, Rag</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>
###PART ROUTING / JOB PLANNER

| NAME: | Joseph Falcao |
| PART: | Gantry Leg Nut |
| DRAWING: | OP#1 FAIR Print |
| MATERIAL: | 40mm x 80mm x 120mm Cold Finish Mild Steel Square 1018 |

**NOTES:** Deburr all edges after every operation.

<table>
<thead>
<tr>
<th>OP #</th>
<th>Operation Description</th>
<th>Machine Tool</th>
<th>Cell</th>
<th>Tooling &amp; Fixtures Required</th>
<th>OP Time (min)</th>
<th>Approval</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Cut Stock to Length &amp; Deburr</td>
<td>Horizontal Band Saw</td>
<td></td>
<td>Dial Caliper, File</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Milling Operation #1</td>
<td>Haas Mill</td>
<td></td>
<td>Milling Vise, Parallels, 75 mm Face Mill, 13 mm Flat End Mill, 3.3 mm Tap Drill Bit for the M4X0.7 Holes, 13 mm 45° Chamfer, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Operation #1 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td></td>
<td>Optical Comparater</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Milling Operation #2</td>
<td>Haas Mill</td>
<td></td>
<td>Milling Vise, Parallels, Soft Jaw Set, 75 mm Face Mill, 15 mm End Mill, 13 mm 45° Chamfer, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Milling Operation #3</td>
<td>Haas Mill</td>
<td></td>
<td>Milling Vise, Parallels, Soft Jaw Set, 15 mm End Mill, 3.3 mm Tap Drill Bit for the M4X0.7 Holes, 13 mm 45° Chamfer, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Drill Press Operation #1</td>
<td>Drill Press</td>
<td></td>
<td>Drill Press Vise, 10.8 mm Tap Drill Bit for the M12X1.25 Holes</td>
<td>15.0</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>Drill Press Operation #2</td>
<td>Drill Press</td>
<td></td>
<td>Drill Press Vise, 3.3 mm Tap Drill Bit for the M4X0.7 Holes</td>
<td>15.0</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Operation #2 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td></td>
<td>Drop Height Indicator, Optical Comparater</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>Clean &amp; Polish</td>
<td>Finishing Table</td>
<td></td>
<td>Mothers Polish, Rag</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix H5: Gantry Top

<table>
<thead>
<tr>
<th>OP #</th>
<th>Operation Description</th>
<th>Machine Tool</th>
<th>Tooling &amp; Fixtures Required</th>
<th>OP Time (min)</th>
<th>Approval</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Cut Stock to Length &amp; Deburr</td>
<td>Horizontal Band Saw</td>
<td>Dial Caliper, File</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Milling Operation #1</td>
<td>Haas Mill</td>
<td>Milling Vise, 75 mm Face Mill, 13 mm Flat End Mill, 12 mm Drill Bit for the Ø 12 mm Holes, 4.4 mm Tap Drill Bit for the M5X0.8 Holes, 13 mm 45° Chamfer, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Operation #1 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td>Optical Comparator</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Milling Operation #2</td>
<td>Haas Mill</td>
<td>Milling Vise, Parallels, Soft Jaw Set, 75 mm Face Mill, 15 mm End MILL, 13 mm 45° Chamfer, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Drill Press Operation #1</td>
<td>Drill Press</td>
<td>Drill Press Vise, 3.3 mm Tap Drill Bit for the M4X0.7 Holes</td>
<td>15.0</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Operation #2 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td>Drop Height Indicator, Optical Comparator</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>Clean &amp; Polish</td>
<td>Finishing Table</td>
<td>Mothers Polish, Rag</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>
NOTES: Deburr all edges after every operation.

<table>
<thead>
<tr>
<th>OP #</th>
<th>Operation Description</th>
<th>Machine Tool</th>
<th>Tooling &amp; Fixtures Required</th>
<th>OP Time (min)</th>
<th>Approval</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Cut Stock to Length &amp; Deburr</td>
<td>Horizontal Band Saw</td>
<td>Dial Caliper File</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Milling Operation #1</td>
<td>Haas Mill</td>
<td>Milling Vise, Parallels for Round Stock, 75 mm Face Mill, Mechanical Edge Finder</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Drill Press Operation #1</td>
<td>Drill Press</td>
<td>Drill Press Vise, 4.5 mm Drill Bit for the Ø 4.5 mm Holes, 6.0 mm Drill Bit for the Ø 8.0 mm Counterbores, 3.3 mm Tap Drill Bit for the M4x0.7 Holes</td>
<td>15.0</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Operation #1 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td>Optical Comparater</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Clean &amp; Polish</td>
<td>Finishing Table</td>
<td>Mothers Polish, Rag</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix H7: Lead Screw Support Raiser

**PART ROUTING / JOB PLANNER**

<table>
<thead>
<tr>
<th>NAME:</th>
<th>Joseph Falcao</th>
</tr>
</thead>
<tbody>
<tr>
<td>PART:</td>
<td>Lead Screw Support Raiser</td>
</tr>
<tr>
<td>DRAWING:</td>
<td>OP#1 FAIR Print</td>
</tr>
<tr>
<td>MATERIAL:</td>
<td>30mm x 80mm x 68.7mm Aluminum Bare Rectangle 6061 T6511</td>
</tr>
</tbody>
</table>

**NOTES:** Deburr all edges after every operation.

<table>
<thead>
<tr>
<th>OP #</th>
<th>Operation Description</th>
<th>Machine Tool</th>
<th>Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Cut Stock to Length &amp; Deburr</td>
<td>Horizontal Band Saw</td>
<td>5.0</td>
</tr>
<tr>
<td>20</td>
<td>Milling Operation #1</td>
<td>Haas Mill</td>
<td>20.0</td>
</tr>
<tr>
<td>30</td>
<td>Operation #1 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td>10.0</td>
</tr>
<tr>
<td>40</td>
<td>Milling Operation #2</td>
<td>Haas Mill</td>
<td>20.0</td>
</tr>
<tr>
<td>50</td>
<td>Drill Press Operation #1</td>
<td>Drill Press</td>
<td>15.0</td>
</tr>
<tr>
<td>60</td>
<td>Operation #2 F.A.I.R Inspection</td>
<td>Metrology Lab</td>
<td>10.0</td>
</tr>
<tr>
<td>70</td>
<td>Clean &amp; Polish</td>
<td>Finishing Table</td>
<td>5.0</td>
</tr>
</tbody>
</table>
### PART ROUTING / JOB PLANNER

**NAME:** Joseph Falcao  
**PART:** Line Constraint  
**DRAWING:** OP#1 FAIR Print  
**MATERIAL:** Ø 25 mm x 130mm 5056 Aluminum Rod

**NOTES:** Deburr all edges after every operation.

<table>
<thead>
<tr>
<th>OP #</th>
<th>Operation Description</th>
<th>Machine Tool</th>
<th>Cell</th>
<th>Tooling &amp; Fixtures Required</th>
<th>OP Time (min)</th>
<th>Approval</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Cut Stock to Length &amp; Deburr</td>
<td>Horizontal Band Saw</td>
<td></td>
<td>Dial Caliper File</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>
| 20   | Milling Operation #1                   | Haas Mill    |               | Milling Vise Parallels for Round Stock  
75 mm Face Mill  
Mechanical Edge Finder | 20.0          |          |
| 30   | Drill Press Operation #1               | Drill Press  |               | Drill Press Vise  
4.5 mm Drill Bit for the Ø 4.5 mm Holes  
8.0 mm Drill Bit for the Ø 8.0 mm Counterbores  
3.3 mm Tap Drill Bit for the M4X0.7 Holes | 15.0          |          |
| 40   | Operation #1 F.A.I.R Inspection        | Metrology Lab|               | Optical Comparator  
Drop Height Indicator  
Test Type Indicator | 10.0          |          |
| 50   | Clean & Polish                         | Finishing Table|               | Mothers Polish Rag | 5.0           |          |

Appendix H8: Line Constraint
## Test Plan

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>Rail Straightness</td>
<td>Run system and check with individual measurements</td>
<td>2 micron R PV 0 C 5/6 5/20 Unconfirmed 0 0</td>
<td>Theoretically achieved, not checked as we could not get both motor pairs running at the same time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration 2</td>
<td>Rail Parallelism</td>
<td>Run system and check with individual measurements</td>
<td>2 micron W PV 0 C 5/6 5/20 Unconfirmed 0 0</td>
<td>Theoretically achieved, not checked as we could not get both motor pairs running at the same time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration 3</td>
<td>XBAS Speed</td>
<td>XBAS must complete its task under the specified time</td>
<td>&lt;10 minutes J PV 10 C 5/20 5/27 Failed 0 10</td>
<td>Gantry Motor stalls out at higher speeds</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration 4</td>
<td>Rail Screw Torque</td>
<td>XBAS must torque rail screws to specified torque</td>
<td>2 Nm W PV 1 C 5/20 5/27 Failed 0 1</td>
<td>Solenoids too underpowered to move screwdriver in</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iteration 5</td>
<td>System Automation</td>
<td>Repeated Testing to determine if the machine’s automation of the XBAS assembly is reliable</td>
<td>1 failure out of 200 or more runs R PV 3 C 5/20 5/27 Failed 0 3</td>
<td>Unable to use screwdriver, couldn’t complete test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Survey 6</td>
<td>X-Beam Loading</td>
<td>Load X-Beam onto basic constraints without assistance</td>
<td>No Fail J DV 3 B 4/20 4/29 Passed 3 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Survey 7</td>
<td>Loading by User</td>
<td>Inspection for awkward loading position during testing of Ease of Use and Loading</td>
<td>No bending over, twisting of body, and a loading point from hip to mid-chest level J DV 3 B 5/11 5/23 Passed 3 0</td>
<td>Direct Loading following OSHA standards</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Survey 8</td>
<td>Learning Time</td>
<td>Get Engineering Peers and teach then how to operate the machine. Time how long that took, how they performed, and feedback on if the tutorial was enough</td>
<td>User can operate the machine with 15 min or less of teaching time W PV 0 C 5/9 3/9 Unconfirmed 0 0</td>
<td>Ran out of time as we were trying to get machine as functional as possible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Survey 9</td>
<td>Ease of Use</td>
<td>Get Engineering Peers to test the X-Bas and record the number of people that mess up and input on how easy the machine works</td>
<td>Greater than 60% positive feedback W PV 0 C 5/20 5/27 Unconfirmed 0 0</td>
<td>Ran out of time as we were trying to get machine as functional as possible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UI 10</td>
<td>Controller User Interface</td>
<td>Force system to fail to see if error is conveyed</td>
<td>No Fail R DV 5 B 5/9 5/2 Passed 5 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- **R:** Rail
- **PV:** Parallelism
- **W:** Wrench
- **J:** Jig
- **DV:** Directly
- **B:** Both
- **C:** Check
- **PV:** Parallelism
- **J:** Jig
- **DV:** Directly
- **B:** Both
- **C:** Check
- **PV:** Parallelism
- **J:** Jig
- **DV:** Directly
- **B:** Both
- **C:** Check
Appendix K: Table of Inspection Sheets

Appendix K1: Bearing Attachment Plate Configuration 1 ........................................... 135
Appendix K2: Bearing Attachment Plate Configuration 2 ........................................... 136
Appendix K3: Gantry Top .............................................................................................. 137
Appendix K4: Hardstop ............................................................................................... 146
Appendix K5: Leadscrew Support Raiser ...................................................................... 147
Appendix K6: Line Constraint ...................................................................................... 148
<table>
<thead>
<tr>
<th>Dimension ID</th>
<th>Description</th>
<th>Nominal Size</th>
<th>Limits</th>
<th>Actual</th>
<th>Device</th>
<th>Comments</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linear Height</td>
<td>10 mm</td>
<td>±0.5 mm</td>
<td>10.215 mm</td>
<td>Drop Height Indicator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>B</td>
<td>Position of Ø 3.4 mm Hole (1)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.182 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>C</td>
<td>Position of Ø 3.4 mm Hole (2)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.173 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>D</td>
<td>Position of Ø 3.4 mm Hole (3)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.184 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>E</td>
<td>Position of Ø 3.4 mm Hole (4)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.186 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>F</td>
<td>Position of Ø 4.5 mm Hole (1)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.179 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>G</td>
<td>Position of Ø 4.5 mm Hole (2)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.181 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>H</td>
<td>Position of Ø 4.5 mm Hole (3)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.175 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>I</td>
<td>Position of Ø 4.5 mm Hole (4)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.183 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
</tbody>
</table>

Notes: Inspector: Joseph Falcao
## FIRST ARTICLE INSPECTION SHEET

**NAME:** Joseph Falcao  
**PART:** Bearing Attachment Plate Leadscrew  
**DATE:** 1/10/2017  
**DRAWING NUMBER:** OP#1 FAIR Print

<table>
<thead>
<tr>
<th>Dimension ID</th>
<th>Description</th>
<th>Nominal Size</th>
<th>Limits</th>
<th>Actual</th>
<th>Device</th>
<th>Comments</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linear Height</td>
<td>10 mm</td>
<td>±0.5 mm</td>
<td>10.128 mm</td>
<td>Drop Height Indicator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>B</td>
<td>Position of Ø 3.4 mm Hole (1)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.179 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>C</td>
<td>Position of Ø 3.4 mm Hole (2)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.156 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>D</td>
<td>Position of Ø 3.4 mm Hole (3)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.181 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>E</td>
<td>Position of Ø 3.4 mm Hole (4)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.175 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>F</td>
<td>Position of Ø 3.4 mm Hole (5)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.183 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>G</td>
<td>Position of Ø 3.4 mm Hole (6)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.172 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>H</td>
<td>Position of Ø 3.4 mm Hole (7)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.178 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>I</td>
<td>Position of Ø 3.4 mm Hole (8)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.185 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>J</td>
<td>Position of Ø 4.5 mm Hole (1)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.191 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>K</td>
<td>Position of Ø 4.5 mm Hole (2)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.189 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>L</td>
<td>Position of Ø 4.5 mm Hole (3)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.184 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>M</td>
<td>Position of Ø 4.5 mm Hole (4)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.192 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
</tbody>
</table>

**Notes:**

Inspector: Joseph Falcao

---

Appendix K2: Bearing Attachment Plate Configuration 2
### Appendix K3: Gantry Top

<table>
<thead>
<tr>
<th>Circle CB 1</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>20.0487</td>
<td>20.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0487</td>
<td>✔️</td>
</tr>
<tr>
<td>Center Y</td>
<td>-15.1374</td>
<td>-15.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.1374</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>19.9467</td>
<td>20.0000</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>-0.0533</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle CB 4</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>20.0730</td>
<td>20.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0730</td>
<td>✔️</td>
</tr>
<tr>
<td>Center Y</td>
<td>-65.1491</td>
<td>-65.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.1491</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>19.9611</td>
<td>20.0000</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>-0.0389</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 1</th>
<th>Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.2625</td>
<td>0.3000</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle 12.8 thru 1</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>20.0788</td>
<td>20.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0788</td>
<td>✔️</td>
</tr>
<tr>
<td>Center Y</td>
<td>-15.1188</td>
<td>-15.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.1188</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>12.9505</td>
<td>12.8000</td>
<td>0.3000</td>
<td>-0.3000</td>
<td>0.1505</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle 12.8 thru 4</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>20.0772</td>
<td>20.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0772</td>
<td>✔️</td>
</tr>
<tr>
<td>Center Y</td>
<td>-65.1367</td>
<td>-65.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.1367</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>12.9543</td>
<td>12.8000</td>
<td>0.3000</td>
<td>-0.3000</td>
<td>0.1543</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 1</th>
<th>Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.2625</td>
<td>0.3000</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Circle M5 2</td>
<td>Actual</td>
<td>Nominal</td>
<td>Upper</td>
<td>Lower</td>
<td>Dev/Nom</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>---------</td>
<td>--------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Center X</td>
<td>112.0808</td>
<td>112.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0808</td>
</tr>
<tr>
<td>Center Y</td>
<td>-17.0234</td>
<td>-16.9300</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0934</td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.2470</td>
<td>0.3000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 4</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>64.9143</td>
<td>64.8500</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0643</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-64.1909</td>
<td>-64.0700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.1209</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.2739</td>
<td>0.3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 3</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>112.0526</td>
<td>112.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0526</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-64.1732</td>
<td>-64.0700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.1032</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.2317</td>
<td>0.3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle Actuator bore 1</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>88.4538</td>
<td>88.4300</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0238</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-40.6433</td>
<td>-40.5000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.1433</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>39.9334</td>
<td>40.0000</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>-0.0666</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.2904</td>
<td>0.3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 5</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>-23.5261</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0439</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>23.6254</td>
<td>23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0554</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.1414</td>
<td>0.3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 6</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>23.6073</td>
<td>23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0373</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>23.6021</td>
<td>23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0321</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0985</td>
<td>0.3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 7</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>23.5872</td>
<td>23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0172</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-23.5705</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0005</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0344</td>
<td>0.3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 8</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>-23.5295</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0405</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-23.5553</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0147</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0862</td>
<td>0.3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle Actuator bore 2</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diameter</td>
<td>39.9314</td>
<td>40.0000</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>-0.0686</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.1200</td>
<td>0.3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Program: GantryTop_1

**Units:** mm, dec deg

**3/2/2017 3:09:39 PM**

Rounding Enabled - Units: x.xxxx Angle: x.xx

<table>
<thead>
<tr>
<th>Circle CB 2</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>320.0176</td>
<td>320.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0176</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-15.2177</td>
<td>-15.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.2177</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>19.9350</td>
<td>20.0000</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>-0.0650</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.4369</td>
<td>0.3000</td>
<td>0.1369</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle 12.8 thru 2</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>320.0293</td>
<td>320.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0293</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-15.2294</td>
<td>-15.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.2294</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>12.9574</td>
<td>12.8000</td>
<td>0.3000</td>
<td>-0.3000</td>
<td>0.1574</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.4625</td>
<td>0.3000</td>
<td>0.1625</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle CB 3</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>319.9808</td>
<td>320.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0192</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-65.2015</td>
<td>-65.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.2015</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>19.9215</td>
<td>20.0000</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>-0.0785</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.4048</td>
<td>0.3000</td>
<td>0.1048</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle 12.8 thru 3</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>320.0189</td>
<td>320.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0189</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-65.2116</td>
<td>-65.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.2116</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>12.9536</td>
<td>12.8000</td>
<td>0.3000</td>
<td>-0.3000</td>
<td>0.1536</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.4048</td>
<td>0.3000</td>
<td>0.1048</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line Right edge</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location X</td>
<td>340.0391</td>
<td>340.0000</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>0.0391</td>
<td></td>
</tr>
<tr>
<td>Location Y</td>
<td>-80.1134</td>
<td>-80.0000</td>
<td>0.5000</td>
<td>-0.5000</td>
<td>-0.1134</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 4'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>-23.5715</td>
<td>-23.5800</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0085</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-23.5956</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0256</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0539</td>
<td>0.2500</td>
<td>0.1249</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 1'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>-23.5715</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0015</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>23.5638</td>
<td>23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0062</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0128</td>
<td>0.2500</td>
<td>0.0628</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 2'</th>
<th>Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.0434</td>
<td>0.2500</td>
<td>0.0434</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 3'</th>
<th>Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.0229</td>
<td>0.2500</td>
<td>0.0229</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Circle</td>
<td>Actual</td>
<td>Nominal</td>
<td>Upper</td>
<td>Lower</td>
<td>Dev/Nom</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>Circle M5 8'</td>
<td>-23.5625</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0075</td>
</tr>
<tr>
<td>Center X</td>
<td>-23.5824</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0124</td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0290</td>
<td>0.2500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 5'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>-23.5625</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0075</td>
<td>✓</td>
</tr>
<tr>
<td>Center Y</td>
<td>23.5982</td>
<td>23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0282</td>
<td>✓</td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0584</td>
<td>0.2500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 6'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>23.5709</td>
<td>23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0009</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>23.5784</td>
<td>23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0084</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0169</td>
<td>0.2500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M5 7'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>23.5542</td>
<td>23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0158</td>
<td></td>
</tr>
<tr>
<td>Center Y</td>
<td>-23.5942</td>
<td>-23.5700</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0242</td>
<td></td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0578</td>
<td>0.2500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle 12.8 thru 1'</th>
<th>Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.0178</td>
<td>0.2500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle 12.8 thru 4'</th>
<th>Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.0178</td>
<td>0.2500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle 12.8 thru 2'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0000</td>
<td>✓</td>
</tr>
<tr>
<td>Center Y</td>
<td>24.9911</td>
<td>25.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0089</td>
<td>✓</td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0178</td>
<td>0.2500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle 12.8 thru 3'</th>
<th>Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.0178</td>
<td>0.2500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Circle M4 1
- **Center X**: 44.8793 (Nominal: 45.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.1207)
- **Center Y**: -4.8652 (Nominal: -5.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: 0.1348)
- **Actual True Position (RFS)**: 0.3619

### Circle M4 2
- **Center X**: 66.8850 (Nominal: 67.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.1150)
- **Center Y**: -4.7946 (Nominal: -5.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: 0.2054)
- **Actual True Position (RFS)**: 0.4708

### Circle M4 3
- **Center X**: -36.0128 (Nominal: -36.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0128)
- **Center Y**: 17.0483 (Nominal: 17.1000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0517)
- **Actual True Position (RFS)**: 0.1066

### Circle M4 4
- **Center X**: -36.0139 (Nominal: -36.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0139)
- **Center Y**: -3.9297 (Nominal: -3.9000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0297)
- **Actual True Position (RFS)**: 0.0655

### Circle M4 5
- **Center X**: 192.9680 (Nominal: 193.0400, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0720)
- **Center Y**: -24.6682 (Nominal: -25.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: 0.3318)
- **Actual True Position (RFS)**: 0.6790
### Circle M4 6
- **Actual Center X**: 214.9480
- **Nominal Center X**: 215.0400
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0920
- **Out/Tol**

- **Actual Center Y**: -24.6748
- **Nominal Center Y**: -25.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: 0.3252
- **Out/Tol**

- **True Position (RFS)**: 0.6759

### Circle M4 7
- **Actual Center X**: 35.9921
- **Nominal Center X**: 36.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0079
- **Out/Tol**

- **Actual Center Y**: 17.9428
- **Nominal Center Y**: 17.1000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0328
- **Out/Tol**

- **True Position (RFS)**: 0.1262

### Circle M4 8
- **Actual Center X**: 272.9488
- **Nominal Center X**: 273.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0512
- **Out/Tol**

- **Actual Center Y**: 17.0467
- **Nominal Center Y**: 17.1000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0533
- **Out/Tol**

- **True Position (RFS)**: 0.3909

### Circle M4 9
- **Actual Center X**: 294.9419
- **Nominal Center X**: 295.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0581
- **Out/Tol**

- **Actual Center Y**: -4.8173
- **Nominal Center Y**: -5.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: 0.1827
- **Out/Tol**

- **True Position (RFS)**: 0.3834

### Circle M4 1'
- **Actual Center X**: -11.0029
- **Nominal Center X**: -11.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0029
- **Out/Tol**

- **Actual Center Y**: -11.0000
- **Nominal Center Y**: -11.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: 0.0058
- **Out/Tol**

- **True Position (RFS)**: 0.0058

### Circle M4 2'
- **Actual Center X**: -11.0029
- **Nominal Center X**: -11.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0029
- **Out/Tol**

- **Actual Center Y**: -11.0000
- **Nominal Center Y**: -11.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: 0.0058
- **Out/Tol**

- **True Position (RFS)**: 0.0058

### Circle M4 5'
- **Actual Center X**: 294.9419
- **Nominal Center X**: 295.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0581
- **Out/Tol**

- **Actual Center Y**: 17.0467
- **Nominal Center Y**: 17.1000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: 0.1827
- **Out/Tol**

- **True Position (RFS)**: 0.0200

### Circle M4 6'
- **Actual Center X**: -11.0029
- **Nominal Center X**: -11.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0029
- **Out/Tol**

- **Actual Center Y**: -11.0000
- **Nominal Center Y**: -11.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: 0.0058
- **Out/Tol**

- **True Position (RFS)**: 0.0200

### Circle M4 9'
- **Actual Center X**: 294.9419
- **Nominal Center X**: 295.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0581
- **Out/Tol**

- **Actual Center Y**: 17.0467
- **Nominal Center Y**: 17.1000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: 0.1827
- **Out/Tol**

- **True Position (RFS)**: 0.0200

### Circle M4 10'
- **Actual Center X**: -11.0029
- **Nominal Center X**: -11.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0029
- **Out/Tol**

- **Actual Center Y**: -11.0000
- **Nominal Center Y**: -11.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: 0.0058
- **Out/Tol**

- **True Position (RFS)**: 0.0200

### Circle M4 3'
- **Actual Center X**: -35.9962
- **Nominal Center X**: -36.0000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: 0.0038
- **Out/Tol**

- **Actual Center Y**: 10.4886
- **Nominal Center Y**: 10.5000
- **Upper Limit**: 1.0000
- **Lower Limit**: -1.0000
- **Dev/Nom**: -0.0114
- **Out/Tol**

- **True Position (RFS)**: 0.0241
Program: GantryTop_2
Units: mm, dec deg

<table>
<thead>
<tr>
<th>Circle M4 4' Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.0225</td>
<td>0.2500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M4 7' Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.0287</td>
<td>0.2500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M4 8' Actual</th>
<th>Angle</th>
<th>Tol/Zone</th>
<th>Reference</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Position (RFS)</td>
<td>0.0394</td>
<td>0.2500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Circle M4 1
- **Center X**: Actual: 70.0136, Nominal: 70.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: 0.0136
- **Center Y**: Actual: 19.9525, Nominal: 19.9600, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0075
- **True Position (RFS)**: Actual: 0.0310, Nominal: 0.3000

### Circle M4 2
- **Center X**: Actual: 140.0098, Nominal: 140.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: 0.0098
- **Center Y**: Actual: 19.9515, Nominal: 19.9600, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0085
- **True Position (RFS)**: Actual: 0.0260, Nominal: 0.3000

### Circle M4 3
- **Center X**: Actual: 200.0036, Nominal: 200.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: 0.0036
- **Center Y**: Actual: 19.9396, Nominal: 19.9600, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0204
- **True Position (RFS)**: Actual: 0.0414, Nominal: 0.3000

### Circle M4 4
- **Center X**: Actual: 269.9510, Nominal: 270.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0490
- **Center Y**: Actual: 19.9475, Nominal: 19.9600, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0125
- **True Position (RFS)**: Actual: 0.1011, Nominal: 0.3000

### Circle M5
- **Center X**: Actual: 299.9654, Nominal: 300.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0346
- **Center Y**: Actual: 9.9961, Nominal: 10.0000, Upper: 1.0000, Lower: -1.0000, Dev/Nom: -0.0039
- **True Position (RFS)**: Actual: 0.0697, Nominal: 0.5000
### Program: GantryTop_3

**Units:** mm, dec deg

**Rounding Enabled - Units: x.xxxx Angle: x.xx**

<table>
<thead>
<tr>
<th>Circle M4 1'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>-99.9809</td>
<td>-100.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0191</td>
<td>✔</td>
</tr>
<tr>
<td>Center Y</td>
<td>0.0008</td>
<td>0.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0008</td>
<td>✔</td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0382</td>
<td>0.2500</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M4 2'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>-29.9847</td>
<td>-30.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0153</td>
<td>✔</td>
</tr>
<tr>
<td>Center Y</td>
<td>0.0025</td>
<td>0.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0025</td>
<td>✔</td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0310</td>
<td>0.2500</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M4 3'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>30.0091</td>
<td>30.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0091</td>
<td>✔</td>
</tr>
<tr>
<td>Center Y</td>
<td>-0.0070</td>
<td>0.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0070</td>
<td>✔</td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0230</td>
<td>0.2500</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circle M4 4'</th>
<th>Actual</th>
<th>Nominal</th>
<th>Upper</th>
<th>Lower</th>
<th>Dev/Nom</th>
<th>Out/Tol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center X</td>
<td>99.9565</td>
<td>100.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>-0.0435</td>
<td>✔</td>
</tr>
<tr>
<td>Center Y</td>
<td>0.0037</td>
<td>0.0000</td>
<td>1.0000</td>
<td>-1.0000</td>
<td>0.0037</td>
<td>✔</td>
</tr>
<tr>
<td>True Position (RFS)</td>
<td>0.0873</td>
<td>0.2500</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### FIRST ARTICLE INSPECTION SHEET

**NAME:** Joseph Falcao  
**PART:** Hard Stop  
**DATE:** 3/22/2017  
**DRAWING NUMBER:** OP#1 FAIR Print

<table>
<thead>
<tr>
<th>Dimension ID</th>
<th>Description</th>
<th>Nominal Size</th>
<th>Limits</th>
<th>Actual</th>
<th>Device</th>
<th>Comments</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linear Height</td>
<td>11 mm</td>
<td>±0.5 mm</td>
<td>11.469 mm</td>
<td>Drop Height Indicator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>B</td>
<td>Position of Ø 4.5 mm Hole (1)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.226 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>C</td>
<td>Position of Ø 4.5 mm Hole (2)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.249 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>D</td>
<td>Position of Ø 3.3 mm Hole</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.237 mm</td>
<td>Optical Comparater</td>
<td></td>
<td>Pass</td>
</tr>
</tbody>
</table>

**Notes:** Inspector: Joseph Falcao
## Appendix K5: Leadscrew Support Raiser

### FIRST ARTICLE INSPECTION SHEET

**NAME:** Joseph Falcao  
**PART:** Lead Screw Support Raiser  
**DATE:** 4/19/2017  
**DRAWING NUMBER:** OP#1 FAIR Print

<table>
<thead>
<tr>
<th>Dimension ID</th>
<th>Description</th>
<th>Nominal Size</th>
<th>Limits</th>
<th>Actual</th>
<th>Device</th>
<th>Comments</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linear Height</td>
<td>68.70 mm</td>
<td>±0.5 mm</td>
<td>68.963 mm</td>
<td>Drop Height Indicator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>B</td>
<td>Position of Ø 6.0 mm Hole (1)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.158 mm</td>
<td>Optical Comparator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>C</td>
<td>Position of Ø 6.0 mm Hole (2)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.162 mm</td>
<td>Optical Comparator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>D</td>
<td>Position of Ø 5.0 mm Hole (1)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.214 mm</td>
<td>Optical Comparator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>E</td>
<td>Position of Ø 5.0 mm Hole (2)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.237 mm</td>
<td>Optical Comparator</td>
<td></td>
<td>Pass</td>
</tr>
</tbody>
</table>

**Notes:** Inspector: Joseph Falcao
# FIRST ARTICLE INSPECTION SHEET

**NAME:** Joseph Falcao  
**PART:** Line Constraint  
**DATE:** 3/22/2017  
**DRAWING NUMBER:** OP#1 FAIR Print

<table>
<thead>
<tr>
<th>Dimension ID</th>
<th>Description</th>
<th>Nominal Size</th>
<th>Limits</th>
<th>Actual</th>
<th>Device</th>
<th>Comments</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Radius</td>
<td>12 mm</td>
<td>±0.5 mm</td>
<td>12.374 mm</td>
<td>Drop Height Indicator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>B</td>
<td>Position of Ø 4.5 mm Hole (1)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.157 mm</td>
<td>Optical Comparator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>C</td>
<td>Position of Ø 4.5 mm Hole (2)</td>
<td>0.0</td>
<td>±0.3 mm</td>
<td>0.162 mm</td>
<td>Optical Comparator</td>
<td></td>
<td>Pass</td>
</tr>
<tr>
<td>D</td>
<td>Flatness</td>
<td>0.000</td>
<td>±0.004 mm</td>
<td>0.003 mm</td>
<td>Test Type Indicator</td>
<td></td>
<td>Pass</td>
</tr>
</tbody>
</table>

**Notes:** Inspector: Joseph Falcao
Appendix L: Electronics Basic Schematic
<table>
<thead>
<tr>
<th>LINE</th>
<th>SOURCE</th>
<th>SUB-SUB-ASSEMBLY OR PART DESCRIPTION</th>
<th>SOURCE PART NO.</th>
<th>LIST PRICE</th>
<th>QTY.</th>
<th>TOTAL</th>
<th>LINK</th>
<th>MICRO VU ITEM NO.</th>
<th>NEED BY DATE</th>
<th>MAP LOCATION</th>
<th>PURPOSE</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Digi Key</td>
<td>Run Button Green non-illuminated button</td>
<td>EG1935-N0</td>
<td>$3.13</td>
<td>ea.</td>
<td>1</td>
<td>$3.13</td>
<td>EG1935-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Big green button for telling the machine to start the selected mode</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Digi Key</td>
<td>Emergency Stop Push pull red button</td>
<td>679-3575-N0</td>
<td>$24.48</td>
<td>ea.</td>
<td>1</td>
<td>$24.48</td>
<td>679-3575-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Big red button for emergency stop. Push pull head</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Digi Key</td>
<td>State Selector 3 position turn switch</td>
<td>CN9823-N0</td>
<td>$4.81</td>
<td>ea.</td>
<td>1</td>
<td>$4.81</td>
<td>CN9823-N0</td>
<td>2/24/2017</td>
<td></td>
<td>3 Position switches for selecting one of three modes: OFF, Assemble, and Calibration</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Digi Key</td>
<td>Piezo Buzzer Warning in case of fault</td>
<td>668-1463-N0</td>
<td>$0.91</td>
<td>ea.</td>
<td>1</td>
<td>$0.91</td>
<td>668-1463-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Audio signal for a critical error or completion</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Digi Key</td>
<td>Proto Board For any additional wiring stops</td>
<td>1568-1082-N0</td>
<td>$4.95</td>
<td>ea.</td>
<td>1</td>
<td>$4.95</td>
<td>1568-1082-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Wiring for piezo buzzer</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Digi Key</td>
<td>DC/DC Buck Converter Powers SV components</td>
<td>1597-1243-N0</td>
<td>$8.50</td>
<td>ea.</td>
<td>1</td>
<td>$8.50</td>
<td>1597-1243-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Powers and protects SV components</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Digi Key</td>
<td>Terminal Block Organize Power Distribution</td>
<td>WM5765-N0</td>
<td>$4.32</td>
<td>ea.</td>
<td>1</td>
<td>$4.32</td>
<td>WM5765-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Organize power distribution with screw terminals</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Digi Key</td>
<td>Level Shifter 5V-3.3V</td>
<td>1568-1209-N0</td>
<td>$2.95</td>
<td>ea.</td>
<td>2</td>
<td>$5.90</td>
<td>1568-1209-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Connects SV to 3.3V</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Digi Key</td>
<td>Connector 20 INCH LBS MINIATURE TORQUE LIMITER</td>
<td>S7H4A-3.25-815</td>
<td>$262.91</td>
<td>ea.</td>
<td>1</td>
<td>$262.91</td>
<td>S7H4A-3.25-815</td>
<td>2/24/2017</td>
<td></td>
<td>Stepper motor controllers for rail actuators, beam actuator, and gantry actuator</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Digi Key</td>
<td>Empty Gun Mount Green Light to indicate the machine to start the machine Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Digi Key</td>
<td>Relay Relays for Powering screwdriver components</td>
<td>GH7019-N0</td>
<td>$13.05</td>
<td>ea.</td>
<td>4</td>
<td>$52.20</td>
<td>GH7019-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Relays for high power components, allows microcontroller to turn on solenoids and motors for screwdriver</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Digi Key</td>
<td>Microstepping Driver Microstepping driver for high precision movements</td>
<td>497-16251-N0</td>
<td>$15.96</td>
<td>ea.</td>
<td>2</td>
<td>$31.92</td>
<td>497-16251-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Stepper motor controllers for rail actuators, beam actuator, and gantry actuator</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Digi Key</td>
<td>Relay Board Holds Relays for the machine to start the machine Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Digi Key</td>
<td>Green Panel Light Green LED</td>
<td>350-3981-N0</td>
<td>$5.30</td>
<td>ea.</td>
<td>1</td>
<td>$5.30</td>
<td>350-3981-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Green light to indicate the machine is ready to start or has finished its current run</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Digi Key</td>
<td>Amber Panel Light Amber LED</td>
<td>350-3982-N0</td>
<td>$5.30</td>
<td>ea.</td>
<td>1</td>
<td>$5.30</td>
<td>350-3982-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Amber light indicates the machine is currently running</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Digi Key</td>
<td>Red Panel Light Red LED</td>
<td>350-3980-N0</td>
<td>$5.30</td>
<td>ea.</td>
<td>1</td>
<td>$5.30</td>
<td>350-3980-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Red light indicates a fatal error detected by the machine and the machine stops</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Digi Key</td>
<td>18 Gage Wire Ferrules Wire Ferrules - Black</td>
<td>277-2186-N0</td>
<td>$0.19</td>
<td>ea.</td>
<td>30</td>
<td>$5.61</td>
<td>277-2186-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Wire ferrules for control cables.</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Digi Key</td>
<td>18 Gage Wire Ferrules Wire Ferrules - Red</td>
<td>288-1015-N0</td>
<td>$0.11</td>
<td>ea.</td>
<td>30</td>
<td>$3.36</td>
<td>288-1015-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Wire ferrules for power distribution.</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Digi Key</td>
<td>Screw Terminals 6 position</td>
<td>A68337-N0</td>
<td>$4.23</td>
<td>ea.</td>
<td>4</td>
<td>$16.92</td>
<td>A68337-N0</td>
<td>2/24/2017</td>
<td></td>
<td>Wires for piezo buzzer wiring steps</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Dr. Ridgely Shoe of Brian Breakout Board (Dr. Ridgely)</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
<td>ea.</td>
<td>1</td>
<td>$0.00</td>
<td>NONE</td>
<td>2/24/2017</td>
<td></td>
<td>Breakout Board for Controller &amp; Programmer</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Fix it Sticks Screwdriver Actuator 20 INCH LBS MINIATURE TORQUE LIMITER</td>
<td>20 INCH LBS MINIATURE TORQUE LIMITER</td>
<td>$40.00</td>
<td>ea.</td>
<td>2</td>
<td>$80.00</td>
<td>20 INCH LBS MINIATURE TORQUE LIMITER</td>
<td>2/24/2017</td>
<td></td>
<td>Subassembly for torque screws</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Grainger Screwdriver Actuator Insert BLT 1/4&quot;, Hex, 4mm</td>
<td>30Y909</td>
<td>$0.94</td>
<td>ea.</td>
<td>2</td>
<td>$1.88</td>
<td>30Y909</td>
<td>2/24/2017</td>
<td></td>
<td>Subassembly for torque screws</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Haydon Kerk Rail Leveling Actuators Linear Actuator</td>
<td>5TH4A-3.25-815</td>
<td>$162.91</td>
<td>ea.</td>
<td>2</td>
<td>$325.82</td>
<td>5TH4A-3.25-815</td>
<td>2/24/2017</td>
<td></td>
<td>Actuators that push target rails down against the granite block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Haydon Kerk Non-Captive NEMA 14 Stepper Motor</td>
<td>35FNV-2.33-906</td>
<td>$125.57</td>
<td>ea.</td>
<td>1</td>
<td>$125.57</td>
<td>35FNV-2.33-906</td>
<td>2/24/2017</td>
<td></td>
<td>Levels the X-beam</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Heidenhain Heidenhain Profe X7- 60M</td>
<td>359341-01</td>
<td>$2,781.00</td>
<td>ea.</td>
<td>1</td>
<td>$2,781.00</td>
<td>Contacted Via Email</td>
<td>2/24/2017</td>
<td></td>
<td>Measure level of the X-Beam</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Heidenhain Heidenhain Interpolation Electronics</td>
<td>536397-01</td>
<td>$808.00</td>
<td>ea.</td>
<td>1</td>
<td>$808.00</td>
<td>536397-01</td>
<td>2/24/2017</td>
<td></td>
<td>Interpolation electronics board translates signal into readable encoder ticks for our system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Heidenhain Heidenhain MT60 Switch Board</td>
<td>317436-02</td>
<td>$361.00</td>
<td>ea.</td>
<td>1</td>
<td>$361.00</td>
<td>NONE</td>
<td>2/24/2017</td>
<td></td>
<td>Switch board allows us to activate the MT60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>KRO Linear rails and bearings M12/4</td>
<td>LWS124 H1000 BCS</td>
<td>0.99</td>
<td>ea.</td>
<td>2</td>
<td>$1.98</td>
<td>LWS124 H1000 BCS</td>
<td>2/24/2017</td>
<td></td>
<td>The linear rails</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Msimu USA Granite Parallel Bolts Bolts that attach the granite parallel to the granite block</td>
<td>5CB12-60</td>
<td>$1.10</td>
<td>ea.</td>
<td>6</td>
<td>$6.60</td>
<td>5CB12-60</td>
<td>4/3/2017</td>
<td></td>
<td>The base of the device where the machine will be built on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Msimu USA NEMA 23 High Torque Stepper Motor The driving motor of the actuator</td>
<td>H723-601</td>
<td>$105.00</td>
<td>ea.</td>
<td>1</td>
<td>$105.00</td>
<td>H723-601</td>
<td>2/24/2017</td>
<td></td>
<td>Actuator Subassembly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
<td>Part Number</td>
<td>Quantity</td>
<td>Unit Cost</td>
<td>Total Cost</td>
<td>Date</td>
<td>Subassembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
<td>------------</td>
<td>------</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Misumi USA</td>
<td>NEMA 23 Bolts</td>
<td>Attatch the motor to the main housing</td>
<td>SCB5-20</td>
<td>6 ea.</td>
<td>$0.36 ea.</td>
<td>2/24/2017</td>
<td>Actuator Subassembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Misumi USA</td>
<td>Lead Screw Nuts - Anti-Backlash Type</td>
<td>Nut that connects the leadscrew to the motor</td>
<td>MTRH10</td>
<td>1 ea.</td>
<td>$136.42 ea.</td>
<td>2/24/2017</td>
<td>Actuator Subassembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Misumi USA</td>
<td>Coupling - Sit, Clamping</td>
<td>Connects the shaft of the motor to the leadscrew</td>
<td>CPLSC20-6-6.35</td>
<td>1 ea.</td>
<td>$59.60 ea.</td>
<td>2/24/2017</td>
<td>Actuator Subassembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Misumi USA</td>
<td>Lead Screw - Both Ends Stopped</td>
<td>Lead screw with both ends stopped</td>
<td>MTSRW12-950-F20-V8-S20-Q6</td>
<td>1 ea.</td>
<td>$65.39 ea.</td>
<td>2/24/2017</td>
<td>Actuator Subassembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Misumi USA</td>
<td>Lead Screw Support Side</td>
<td>Support for the free end of the lead screw</td>
<td>MTFU8</td>
<td>1 ea.</td>
<td>$32.76 ea.</td>
<td>2/24/2017</td>
<td>Actuator Subassembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Misumi USA</td>
<td>Support Bolts</td>
<td>Bolts from the support to the raiser</td>
<td>SCB6-45</td>
<td>2 ea.</td>
<td>$0.46 ea.</td>
<td>2/24/2017</td>
<td>Actuator Subassembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Misumi USA</td>
<td>Nailer Bolts</td>
<td>Bolts from the raiser to the granite plate</td>
<td>SCN4-10</td>
<td>87 ea.</td>
<td>$0.30 ea.</td>
<td>2/24/2017</td>
<td>Actuator Subassembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Misumi USA</td>
<td>Probe Bolts</td>
<td>1 ea.</td>
<td>SCB4-40</td>
<td>$0.59 ea.</td>
<td>2 ea.</td>
<td>$1.18</td>
<td>2/24/2017</td>
<td>Measure level of the X-Beam</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Misumi USA</td>
<td>Rail Leveling Actuators</td>
<td>Precision Rods - fli Tolerance / f7</td>
<td>RGDCG14-35-MC6</td>
<td>2 ea.</td>
<td>$16.12 ea.</td>
<td>2/24/2017</td>
<td>Actuators that push target rails down against the granite</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Misumi USA</td>
<td>Gantry Frame Bolts</td>
<td>Bolts from to attach bearing to afore mentioned plate (AS)</td>
<td>SCSB3-10</td>
<td>12 ea.</td>
<td>$3.60 ea.</td>
<td>2/24/2017</td>
<td>The linear rails</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Ball Splines - Both Ends Steooed</td>
<td>BSJM10-80-F5-E15-P5-Q8</td>
<td>2 ea.</td>
<td>$84.24 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Pillow Blocks - Diamond Shape Flanged</td>
<td>HRR8</td>
<td>2 ea.</td>
<td>$48.24 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Compression Springs - Round Wires, Standard Lengths, Stainless Steel</td>
<td>UF8-15</td>
<td>4 ea.</td>
<td>$4.52 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Steel Spur Gears - Pressure</td>
<td>GEABBl.0-20-6-B-6.35</td>
<td>2 ea.</td>
<td>$42.64 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Steel Spur Gears - Pressure</td>
<td>GEABBl.0-20-6-B-6.35</td>
<td>2 ea.</td>
<td>$42.64 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Steel Spur Gears - Pressure</td>
<td>GEABBl.0-20-6-B-6.35</td>
<td>2 ea.</td>
<td>$42.64 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Steel Spur Gears - Pressure</td>
<td>GEABBl.0-20-6-B-6.35</td>
<td>2 ea.</td>
<td>$42.64 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Steel Spur Gears - Pressure</td>
<td>GEABBl.0-20-6-B-6.35</td>
<td>2 ea.</td>
<td>$42.64 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Steel Spur Gears - Pressure</td>
<td>GEABBl.0-20-6-B-6.35</td>
<td>2 ea.</td>
<td>$42.64 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Steel Spur Gears - Pressure</td>
<td>GEABBl.0-20-6-B-6.35</td>
<td>2 ea.</td>
<td>$42.64 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>Misumi USA</td>
<td>Screwdriver Actuator Steel Spur Gears - Pressure</td>
<td>GEABBl.0-20-6-B-6.35</td>
<td>2 ea.</td>
<td>$42.64 ea.</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>Misumi USA</td>
<td>Steel Bushings with Pillow Blocks - Double Bushings</td>
<td>n/a</td>
<td>2 ea.</td>
<td>$62.70 ea.</td>
<td>2/24/2017</td>
<td>Levels the X-beam</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>Misumi USA</td>
<td>Precision Linear Shaft</td>
<td>n/a</td>
<td>2 ea.</td>
<td>$62.70 ea.</td>
<td>2/24/2017</td>
<td>Levels the X-beam</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>Misumi USA</td>
<td>D- Rings for angular misalignment</td>
<td>n/a</td>
<td>2 ea.</td>
<td>$62.70 ea.</td>
<td>2/24/2017</td>
<td>Levels the X-beam</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>Misumi USA</td>
<td>Steel Bushings with Pillow Blocks - Double Bushings</td>
<td>n/a</td>
<td>2 ea.</td>
<td>$62.70 ea.</td>
<td>2/24/2017</td>
<td>Levels the X-beam</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>Misumi USA</td>
<td>Precision Linear Shaft</td>
<td>n/a</td>
<td>2 ea.</td>
<td>$62.70 ea.</td>
<td>2/24/2017</td>
<td>Levels the X-beam</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>Misumi USA</td>
<td>Steel Bushings with Pillow Blocks - Double Bushings</td>
<td>n/a</td>
<td>2 ea.</td>
<td>$62.70 ea.</td>
<td>2/24/2017</td>
<td>Levels the X-beam</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>Misumi USA</td>
<td>Steel Bushings with Pillow Blocks - Double Bushings</td>
<td>n/a</td>
<td>2 ea.</td>
<td>$62.70 ea.</td>
<td>2/24/2017</td>
<td>Levels the X-beam</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Total:** 151 items
<table>
<thead>
<tr>
<th></th>
<th>Company</th>
<th>Part Number</th>
<th>Description</th>
<th>Quantity</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>Misumi USA</td>
<td>Cable Carrier</td>
<td>Gantry Cable Management MHPKS206-50-32-A</td>
<td>1</td>
<td>$62.32</td>
<td>$62.32</td>
</tr>
<tr>
<td>60</td>
<td>Misumi USA</td>
<td>4 core stranded 18 gage wire</td>
<td>MVVS-A-0.75-6-4</td>
<td>1</td>
<td>$27.58</td>
<td>$27.58</td>
</tr>
<tr>
<td>61</td>
<td>MiscoDirect Screwdriver Actuator</td>
<td>Full Solenoid 74097841</td>
<td>$50.32 ea. 4 $201.28 74097841</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>n/a</td>
<td>Granite Block, 100mm x 400mm x 100mm</td>
<td>n/a $450.00 ea. 1 $450.00 n/a</td>
<td>4/3/2017</td>
<td>The base of the device where the machine will be built on</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>n/a</td>
<td>Granite Parallel Constraints, 800mm x 50mm x 50mm</td>
<td>n/a $230.00 ea. 2 $460.00 n/a</td>
<td>4/3/2017</td>
<td>The base of the device where the machine will be built on</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Online Metals</td>
<td>Lead Screw Support Raiser</td>
<td>A raiser for the lead screw support EXTRUDED ALUMINUM BARE RECTANGLE 6061 T6511 - Random Length</td>
<td>1</td>
<td>$21.43</td>
<td>$21.43</td>
</tr>
<tr>
<td>65</td>
<td>Online Metals</td>
<td>Metal Stock for attaching Bearing to Gantry</td>
<td>n/a Part of the Gantry Order $0.00 ea. 2 $0.00 n/a Gantry Order</td>
<td>2/24/2017</td>
<td>The linear rails</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>Online Metals</td>
<td>Gantry Steel</td>
<td>n/a COLD FINISH MILD STEEL RECTANGLE 1018 $313.20 4 feet 1 $313.20 n/a COLD FINISH MILD STEEL RECTANGLE 1018</td>
<td>2/24/2017</td>
<td>The linear rails</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>Online Metals</td>
<td>50.8 mm Cylindrical Stock (Hardstop)</td>
<td>Random Length should provide enough materials for two- three- tries 50.8 mm Dia 6061 Al rod $51.66 ea. 1 $51.66 n/a &amp; 84 Dia 6061 Al rod</td>
<td>4/3/2017</td>
<td>Keeps the X-Beam in a predefined position and protects the actuator</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>Online Metals</td>
<td>Sheet Metal</td>
<td>Material to make the housing structure with ALUMINUM BARE SHEET 6061 T6 $21.52 24x36&quot; 1 $21.52 ALU SHEET 6061 T6</td>
<td>4/3/2017</td>
<td>Protection for the gantry subassembly when the machine is not running</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>Online Metals</td>
<td>Sheet Metal</td>
<td>Material to make the housing structure with ALUMINUM BARE SHEET 6061 T6 $21.52 24x36&quot; 1 $21.52 ALU SHEET 6061 T6</td>
<td>4/3/2017</td>
<td>Attachment to the housing for the gantry that covers the electronics</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>Pittman Motors</td>
<td>Screwdriver Actuator</td>
<td>DC Gearmotor GMS9345023-R1-6P $176.67 ea. 2 $353.34 GMS9345023-R1-6P</td>
<td>2/24/2017</td>
<td>Subassembly for torquing screws</td>
<td></td>
</tr>
</tbody>
</table>

Total 275 parts $8,605.17
Appendix N: Electronics Diagram
Appendix O: Final Program Flow Chart List

Appendix O1: BeamActuator.Home() .................................................................................. 155
Appendix O2: BeamActuator.Move() .................................................................................. 156
Appendix O3: Gantry.Home() ............................................................................................ 157
Appendix O4: Gantry.Move() .............................................................................................. 158
Appendix O5: Import.BoltPattern() ..................................................................................... 159
Appendix O6: ImportCalibration ......................................................................................... 160
Appendix O7: Import.Song() .............................................................................................. 161
Appendix O8: main.Assembly_Mode() ............................................................................... 162
Appendix O9: main.Calibration_Mode() ............................................................................. 163
Appendix O10: main.ErrorHandler() ................................................................................ 164
Appendix O11: main.Home() ............................................................................................. 165
Appendix O12: main.Leveling_Mode() .............................................................................. 166
Appendix O13: main.Lights_Sound_Action() ................................................................... 167
Appendix O14: main.Sleep_Mode() .................................................................................. 168
Appendix O15: main.TorqueDown() ................................................................................ 169
Appendix O16: Probe.Home() ........................................................................................... 170
Appendix O17: Probe.Probe() .......................................................................................... 171
Appendix O18: RailAct.Move() ........................................................................................ 172
Appendix O19: RailAct.Home() ......................................................................................... 173
Appendix O20: setup.FileCheck() .................................................................................... 174
Function Initialization

Check if limit switch is pressed

Back off switch i.e. ReleaseSW

Switch released?

Move Beam Actuator towards limit switch using GoUntil (which stops at switch)

While loop

Beam Actuator busy/still moving

HardHiZ/Turn off Beam Actuator motor

Return/exit function Return("Done")

Beam Actuator stalled

Set Beam Actuator Error Flag True

HardHiZ/Turn off Beam Actuator motor

Return/exit function Return("Error Occurred")

Back off switch i.e. ReleaseSW

while Beam Actuator busy/still moving do

Beam Actuator stalled

true

false

false

true

true

false

Set Beam Actuator Error Flag True

HardHiZ/Turn off Beam Actuator motor

Return/exit function Return("Error Occurred")

Return/exit function Return("Done")

Note: Realized after the fact that this can be replaced with a normal run function and the gantry busy check replaced with a check for the limit switch.
Function Initialization

Set Variables
TicksPerDistance
Offset
Limit

Destination + Offset <= 0

Set Beam Actuator Error
Flag True

HardHiZ/Turn off Beam Actuator motor

Return/exit function
Return("Error Occurred")

Destination + Offset >= Limit

Set Beam Actuator Error
Flag True

HardHiZ/Turn off Beam Actuator motor

Return/exit function
Return("Error Occurred")

Convert Destination from a distance to #steps on the stepper motor using TicksPerDistance and Offset
IE
Destination = (Destination + Offset) * TicksPerDistance

Move Beam Actuator to Destination

While loop

Did the Beam Actuator reach position

HardHiZ/Turn off Beam Actuator motor

Exit while loop
IE break

Did stall occur

Set Beam Actuator Error
Flag True

HardHiZ/Turn off Beam Actuator motor

Return/exit function
Return("Error Occurred")

Probe == True

Return/exit function
Return("Done")

If Probe function returns error
IE "Error Occurred"

Return/exit function
Return("Error Occurred")
Function Initialization

Check if limit switch is pressed

True

Back off switch 
le ReleaseSW

Switch released?

HardHiZ/Turn off Gantry motor

False

Move Gantry towards limit switch using GoUntil (which stops at switch)

While loop

Gantry busy/still moving

true

Gantry stalled

Set Gantry Error Flag True

false

Back off switch 
le ReleaseSW

HardHiZ/Turn off Gantry motor

Return/exit function
Return("Done")

true

Return/exit function
Return("Error Occurred")

false

While loop

Gantry busy/still moving

true

Gantry stalled

Set Gantry Error Flag True

false

HardHiZ/Turn off Gantry motor

Return/exit function
Return("Done")

Note: Realized after the fact that this can be replaced with a normal run function and the gantry busy check replaced with a check for the limit switch.
Function Initialization

Set Variables: DistancePerStep, xOffset, xLimit

Distance + xOffset = 0

Distance + xOffset = xLimit

Move Gantry to Destination:

While loop

Did stall occur?

Did the gantry reach position?

Convert Destination from a distance to #steps on the stepper motor using DistancePerStep and xOffset

I.e.

Destination = (Destination + xOffset)/DistancePerStep

Did Probe function return an error?

If Leveling or Assembly flag true

Call Probe with limits on measurement

F Probe function returns error i.e. "Error Occurred"

Return ("Error Occurred")

Call Probe with no limits on measurement

F Probe function returns error i.e. "Error Occurred"

Return ("Error Occurred")

Note: Calling Probe function with no limits means that the user can use any gage block for calibration and the probe will accept all readings. On the downside, the user can mess up measurements (i.e., stop probe prematurely with your hand). If this happens, re-run calibration.

Note: Calling Probe function with limits in case something interferes with probe measurements while beam is being leveled and if no one is there to catch it. May not be necessary as we noticed that it is hard to interfere with the probe. Recommend further testing.

Return/exit function Return("Done")

If Calibration flag True

Call Probe with no limits on measurement

If Probe function returns error i.e. "Error Occurred"

Note: Check Error Flags to determine how to call the Probe function from the Probe file

Call Probe with no limits on measurement

Return (reading)

Exit while loop i.e. break

HardHiZ/Turn off Gantry motor

Set Gantry Error Flag True

Return/exit function Return("Error Occurred")

HardHiZ/Turn off Gantry motor

Set Gantry Error Flag True

Return/exit function Return("Error Occurred")

HardHiZ/Turn off Gantry motor

Set Gantry Error Flag True

Return/exit function Return("Error Occurred")

Return/exit function Return("Error Occurred")

File: Gantry.py
Function: Move(Destination, Probe = False)
Init

Input = Xbeam.get()

Create file name
FileName = "BoltPattern"+str(Input)+".csv"

error = 0, i = 0

Try the following
Process lines, remove certain string bits, store data

For Loop checking items in BoltSide

If item = "R", "L", or "B"
ErrBoltCsv.put(1)
String calling error out
error = 1

If item != R, L, or B
ErrBoltCsv.put(1)
String calling error out
error = 1

i += 1

End of for loop?

For Loop checking items in BoltData

Try to float item

ErrBoltCsv.put(1)
String calling error out
error = 1

Value Error Occurred
No Error

i += 1

End of for loop?

error == 0
return([BoltSide, BoltData])

error != 0
return("And Error Occurred")

File: Import.py
Function: BoltPattern()
Init

Input = Xbeam.get()

error = 0

Create File Name
i.e. For Input = 500,
FileName = Calibration500.csv

Try to open file with file name given in FileName

Store lines read from file into String

Try to split String by as the commas

Store output into List

Counter Variable i = 0

for item in List

Try to float item in List

Store lines read from file into String

Try to split String by as the commas

No Error

Replace item in List with the floated value
cf[0] = float(item)

End of List String

End of List?

error == 1

Add an string describing the error to list String

error != 0

Add an string describing the error to String as a list with one item

Error

Store output into List

No Error

Counter Variable i = 0

i = i+1

error = 1

Error

for item in List

Open Error Report text file for writing

for item in String

Write item to Error Report

Close Error Report

End of List String

Write item to Error Report

Error

Return(List)

Return("Error Occurred")

File: Import.py
Function: Calibration()
Calculate the piano switches as a binary number stored into variable Number:

\[ Number = a \times 2^3 + b \times 2^2 + c \times 2^1 + d \times 2^0 + 1 \]

- Number == 1
- Number == 2
- Number == 4
- Number == 3
- Number == 5
- Number == 6
- Number == 7
- Number == 9
- Number == 8
- Number == 10
- Number == 11
- Number == 12
- Number == 13
- Number == 15
- Number == 14
- Number == 16

File: Import.py
Function: Song()
while loop start
Output = ImportBoltPattern()
if (Output == "str")
ErrorHandler()
break
ErrAssemble.put(1)
while loop end

For Loop
for counter in range(0, len(BoltData))
Output = Move_Gantry_To(BoltData[counter])
Output == "Error Occurred"
break
TorqueDown(BoltSide[counter])
end of for loop

while loop start
Output = Move_Gantry_To(BoltSide[counter])
Output == "Error Occurred"
break
ErrAssemble.put(1)
while loop start

Error Check
ErrorHandler()
break
ErrAssemble.put(1)
end of while loop

while loop start
Home
Error Check
ErrorHandler()
break
ErrAssemble.put(1)
end of while loop

while loop start
zero Flags()
Lights_Sound_Action()
return
```python
init
Input = Xbeam.get()
error = 0
Create File Name
i.e. For Input = 500, FileName = Calibration500.csv
Try to open file with file name given in FileName
Store lines read from file into String
Try to split String by as the commas
Store output into list
Counter Variable i = 0
for item in list
Try to float item in list
Replace item in list with the floated value if float(item)
End of list?
Add an string describing the error in String as a list with one item
error = 1
Add an string describing the error to the list String
error = 1
No Error
Return(List)
error != 0
false
Return("Error Occurred")
for item in String
Write item to Error Report
End of List String
return("Error Occurred")
Close Error Report
error != 0
false
Return(List)
for loop beginning
return("Error Occurred")
Open Error Report text file for writing
Write item to Error Report
End of List String
```

File: main.py
Function: Calibration_Mode()
Err.FileCheck.get() == 0
ErrSong.get() == 0
ErrBoltCsv.get() == 0
ErrCalCsv == 0
Err.FileCheck.get() == 1
ErrSong.get() == 1
ErrBoltCsv.get() == 1
ErrCalCsv == 1
ErrInit.get() == 1
ErrSleep.get() == 1
ErrLeveling.get() == 1
ErrAssembly.get() == 1
ErrProbe.get() == 1
ErrGantry.get() == 1
ErrBeamAct.get() == 1

For now, do nothing

Open "Error Report.txt" for writing to:

ErrInit.get() == 1
ErrSleep.get() == 1
ErrLeveling.get() == 1
ErrAssembly.get() == 1
ErrProbe.get() == 1
ErrGantry.get() == 1
ErrBeamAct.get() == 1

Write which actuator has attempted to run past maximum stroke length allowed

Write "A possible issue has occurred where the system has not been accounted for all possible errors. Please identify and name the new error"

Toggle Lights and Sound
Lights_Sound_Action()

zero_flag[]

Program Ends

Write "There was an error with the sleep mode"

Write "There was an error with leveling the X-Beam during the Assembly mode"

Write "There was an error with assembling the X-Beam during the Assembly mode"

Write "Probe was unable to take valid measurements"

Write "Gantry was unable to move to the destination"

Write "Beam Actuator was unable to move to the destination"

Write "There was an error with initiating the system initialization"
File: main.py
Function:
Home_Mode(*arg)

1. **if Probe or All in args**
   - **True** → Probe.Home()
   - **False**

2. **if RailActR or All or RailAct in args**
   - **True** → RailAct.Home(Right)
   - **False**

3. **if RailActL or All or RailAct in args**
   - **True** → RailAct.Home(Left)
   - **False**

4. **if Screwdrivers or All in args**
   - **True** → Turn off both DC motors and Solenoids
   - **False**

5. **if Gantry or All in args**
   - **True** → Gantry.Home()
   - **False**

6. **if Bact or All in args**
   - **True** → BeamActuator.Home()
   - **False**

7. **exit**
Init

Lights_Sound_Off()

Create Green, Yellow, Red, Blink, Switch, and Stop Variables

ErrFileCheck.get() == 1

ErrProbe.get() == 1

ErrBeamAct.get() == 1

ErrSong.get() == 1

Green = 1

Yellow = 1

Red = 1

Blink = 0

True

True

False

False

False

False

False

True

True

True

Turn Buzzer On

GreenLED.High

Sleep 1 second

Turn Buzzer Off

return()

While Loop Start

Turn On Buzzer

Switch == 0

Green == 1

Yellow == 1

Red == 1

Blink == 1

GreenLED.High()

YellowLED.High()

RedLED.High()

True

False

Go() == 1

(Current-Start) >= 500

False

False

True

Stop == 0

Stop = 1

Stop == 2

Stop == 0

Stop = 1

Stop = 2

Set Start (Time Reference Point)

Set Current (Time relative to Start)

While Loop Start

Go() == 1

(Current-Start) >= 500

False

False

True

Stop == 0

Stop = 1

Stop == 2

Stop == 0

Stop = 1

Stop = 2

Set Current (Time relative to Start)

True

Stop == 0

False

break

End of While Loop

Stop == 2

Stop = 1

Stop = 2

Stop == 0

Stop = 1

Stop = 2

Set Start (Time Reference Point)

Set Current (Time relative to Start)

While Loop Start

Go() == 1

(Current-Start) >= 500

False

False

True

Stop == 0

Stop = 1

Stop == 2

Stop == 0

Stop = 1

Stop = 2

Set Current (Time relative to Start)

True

Stop == 0

False

break

End of While Loop

Stop == 2

Stop = 1

Stop = 2

Stop == 0

Stop = 1

Stop = 2

Return

Lights_Sound_Off()
While Loop 1st Layer

Input == L
Input == R

Move to left actuator point above rail
Move to right actuator point above rail

While Loop 2nd Layer

False
False

Left Solenoid On
Right Solenoid On

sleep 0.5 s

False

Input == L or B
Input == R or B

Left Motor On
Right Motor On

sleep X s

False

Turn off Motors
Turn off Solenoids

sleep 0.5 s

Input == L or B
Input == R or B

Home Left Actuator
Home Right Actuator

False

return

Set Low Torque

Stall Check

Destination Check

False

error = 1
error = 0

True
True

break

Error check

error == 1
error == 0

Input == L
Input == R

Move to left actuator point above granite
Move to right actuator point above granite

True
True

False

while 3rd layer

Destination Check

Stall Check

False

error = 1
error = 0

True
True

break

Tell Actuator to Approach Rail at low torque

Tell actuator to approach destination at high torque

wait until actuator reached destination or stall

0.169
File: Probe.py
Function: Home()

- **Init**
  - **Probe not at reference tick**
    - **False** → **Return/exit function**
    - **True** → **Retract Probe**
  - **While loop**
    - **Probe at reference tick**
      - **True**
      - **Time spent Retracting probe > 2sec**
        - **False**
        - **Zero out encoder**
        - **True** → **Return/exit function**
    - **False**
File: Probe.py
Function: Probe(Limit = False, UpperLimit = 0, LowerLimit = 0)
Init

Set stall to stall input

Get motor from checkSide(side)

Run motor

While loop

Motor still busy?

Motor stalled?

Break/exit loop

Break/exit loop

Set stall to original level

HardHiZ/Turn off Beam Actuator motor

Return/exit function
Return('Completed move command')
File: RailAct.py
Function: Home(Side)

- **Init**
  - Get motor from checkSide(side)
  - At Home?
    - True: Return/exit function Return('Done')
    - False: If motor == 1 (right side)
      - True: Run motor 1
      - False: Run motor 2
  - While loop
    - False: At Home?
    - True: Return/exit function Return('Done')

- Return/exit function
  - Return('Done')
For loop starts:

Create FileList: List of file names (strings) that system checks for.

Print Error Message Intro

There was an error during...

Check Switch == 0

Print File Error Message

The file was missing

Check Switch != 0

return(Error Message prompting user to read ErrorReport.txt)

return(0)

ErrFileCheck.put(1)

GreenLED.High()

YellowLED.High()

RedLED.High()

While Loop Start

Turn On Buzzer

Switch == 0

Green == 1

Yellow == 1

Red == 1

GreenLED.High()

YellowLED.High()

RedLED.High()

True

Turn Buzzer Off

Green == 1

Yellow == 1

Red == 1

GreenLED.Low()

YellowLED.Low()

RedLED.Low()

False

Set Start (Time Reference Point)

Set Current (Time relative to Start)

While Loop Start

Go() == 1

(Current - Start) >= 500

False

False

True

True

Stop == 0

Stop = 1

Stop == 2

Stop == 0

break

End of While Loop

Stop == 0

Switch == 0

False

Switch == 1

False

Switch = 1

True

Switch = 0

True

Stop == 2

False

break

End of while loop

Blink == 1

Lights_Sound_Off()
Appendix P: Final Program Script

Appendix P: Final Program Script................................................................. 175
Appendix P1: BeamActuator.py .................................................................. 176
Appendix P2: encoder.py .......................................................................... 180
Appendix P3: Gantry.py ........................................................................... 182
Appendix P4: Import.py .............................................................................. 186
Appendix P5: l4670nucleo.py .................................................................. 192
Appendix P6: main.py.................................................................................. 204
Appendix P7: Probe.py .............................................................................. 222
Appendix P8: RailAct.py ............................................................................ 225
Appendix P9: setup.py ............................................................................... 228
Appendix P10: task_share.py ..................................................................... 237
Appendix P1: BeamActuator.py

```python
# -*- coding: utf-8 -*-

File: BeamActuator.py
@author: Robert Tam

def Move(Destination, probe = False):
    '''Function which utilizes code from the 16470nucleo.py file to drive the stepper motors.

    Function runs the Beam Actuator till it stalls (in which case the system throws an error and waits for user input) or it reaches the destination in which case it stops and exits the function.

    @param Destination is the input distance x in millimeters from the bottom edge of the X-Beam
    @param probe is True or False, defaulting to False if not called. If True, the function will call the probe function to do one measurement at the end of the move. If False, it will not do that measurement.
    @return Returns one of three things. If the probe == True, the system will return the value read by the probe at the end. If not, then the function will return “Done”. In both cases, if an error occurred, function will return “Error Occured”.''

    TickPerDistance = 111000/51.19 # ticks per mm determined experimentally
    Offset = 3 # Distance from home position of beam actuator to bottom edge of the X-Beam in mm
    Limit = 10 # Maximum travel of the beam actuator before something collides in mm. This is without the offset
    # ISSUE See above 2 variables

    print("Moving Beam Actuator to position x: " + str(Destination))
    print("Probe at the end of the move? " + str(probe))
    # Check that Destination is not outside of the limits
    if Destination + Limit <= 0:
        # If the destination is less than -xOffset, then the destination is behind the home position which would push the system against the housing and possible break the switch.
        print("Error Occured moving the Beam Actuator, Destination behind home")
        ErrBeamAct.put(1)
        return("Error Occured")
    elif Destination+Offset >= (Limit):
        # If the destination is greater than the xLimit, the beam actuator is running the risk of crashing against the leadscrew bearing end support raiser.
        print("Error Occured moving the Beam Actuator, destination past" + \" max limit")
        ErrBeamAct.put(1)
        return("Error Occured")
    print('''Destination of Beam Actuator within limits''')

    # Convert Destination in mm to revolutions to steps
    Destination = int((Destination+Offset)*TickPerDistance)
    print("Destination converted to " + str(Destination)+" number of steps")

    # Move to the new value of Destination
    print("Moving Beam Actuator to Destination")
```

176
Board1.GoTo(1,Destination)

time.sleep_ms(3000)

# Wait for stall or finish flag
print(" waiting for stall or completion of move command")
while True:
    #
    print("Busy? : "+str(Board1.isBusy(1)))
    print("Stalled? : "+str(Board1.isStalled(1)))
    if Board1.isBusy(1) == False:
        # if finish, exit the function
        print(" Beam Actuator in position, exit loop")
        Board1.HardHiZ(1)
        Board1.GetStatus(1,verbose = 0)
        break

    elif Board1.isStalled(1) == True:
        # if stall, stop beam actuator, throw error and return
        print(" Error Occured moving the Beam Actuator to position. "+
        "Beam Actuator stalled out")
        ErrBeamAct.put(1)
        Board1.HardHiZ(1)
        Board1.GetStatus(1,verbose = 0)
        return("Error Occured")

    # If done moving beam actuator and probe option is true, take measurement
    if probe == True:
        print(" Probe part of BeamActuator.Move() function called")
        reading = Probe.Probe()

        if type(reading) == "Error Occured":
            # Error occured, but should have been solved as the Probe function
            # shouldn't be able to finish if there was an error.
            print(" Error occured with probe in the Move_beam actuator_to()")
            return("Error Occured")
        else:
            print(" No error occured with probe in the BeamActuator.Move()")
            return(reading)

    # Else if the beam actuator is done moving and probe option in false, return
    elif probe == False:
        Board1.HardHiZ(1)
        Board1.GetStatus(1,verbose = 0)
        print(" Probe part of BeamActuator.Move() function not called")
        return("Done")

def Home():
    '''Home the Beam Actuator.
    Function input:
    Board1 is the l6470nucleo.Dual6470 class object that the beam actuator
    is a member of.
    Function output:
    outputs and "Error Occured" if there is an error.
    ..."

    if Board1.isHome(1)==True:
        # if at switch, exit.
        # Note, need to add while loop to release switch
        print(' Beam Actuator already homed')
        return
# Home beam actuator Code. +/- 400 is the max speed of the beam actuator

print("  Beam Actuator GoUntil switch command")
Board1.GetStatus(1,verbose = 0)
Board1.GoUntil(1,-400)
ute.time.sleep_ms(250)

# Check home status in while loop
while True:
    if Board1.isBusy(1) == False:
        # beam actuator at switch, continue.
        print("  Board isn't busy anymore")
        #
        Board1.GetStatus(1)
        Board1.HardHiZ(1)
        Board1.GetStatus(1,verbose = 0)
        break
    elif Board1.isStalled(1) == True:
        # motor stalled, return an error and set flag
        print("  Board stalled: "+str(Board1.isStalled(1)))
        Board1.HardHiZ(1)
        Board1.GetStatus(1,verbose = 0)
        print("  Beam Actuator Stalled during Home command")
        print("  ERROR ERROR ERROR, return error")
        ErrBeamAct.put(1)
        return("Error Occured")

# Release switch for beam actuator
print("  releasing switch for Beam Actuator")
Board1.ReleaseSW(1,1)
ute.time.sleep_ms(500)

# Check home status in while loop
start = ute.time.ticks_ms()
while True:
    if Board1.isBusy(1) == False:
        # beam actuator homed, continue.
        print("  Beam Actuator completed ReleaseSW Command")
        Board1.HardHiZ(1)
        status = Board1.GetStatus(1,verbose = 0)
        if status[1-1] & (1<<2) == False:
            break
    else:
        Board1.ReleaseSW(1,1)
    if Board1.isHome(1) == False:
        break
    elif Board1.isStalled(1) == True:
        # motor stalled, return an error and set flag
        Board1.HardHiZ(1)
        Board1.GetStatus(1,verbose = 0)
        print("  Beam Actuator Stalled during Home command")
        print("  ERROR ERROR ERROR, return error")
        ErrBeamAct.put(1)
        return("Error Occured")
        
def Status():
            '''Print information about the board's status for the Beam actuator
to the repl

Function Inputs:
Board1 is the 16470.Dual6470 class object which correlates with the beam actuator

Function Outputs:
None'''

Board1.GetStatus(1,verbose = 1)

# importing modules and objects needed
import Probe
from setup import ErrBeamAct, Board1
import utime
Appendix P2: encoder.py

# -*- coding: utf-8 -*-
#
# @file encoder.py
# File containing class to control a Quad Encoder
# @author Robert Tam
# @author Tommy Yath
# @author Berizohar Padilla
# @copyright GPL Version 3.0

# Pin Callouts

class Quad_Encoder:
    """Class of functions for a DC motor Quadratic encoder
    This class has two functions
    1: .read() returns the current position
    2: .zero() sets the position to zero
    ""
    def __init__(self,pin1,pin2,timer):
        """Initializes the Encoder class by taking inputs and creating channels for said inputs.
        @ param pin1  : Pin location of channel A of the Encoder
        @ param pin2  : Pin location of channel B of the Encoder
        @ param timer : Timer channel associated with pin1 and pin2
        ""
        # Module Importing
        import pyb

        self.timer=timer
        self.pin1=pin1
        self.pin2=pin2
        self.position = 0
        self.current = 0
        self.previous = 0
        self.delta = 0

        # Create Timer Channels
        self.ch1 = self.timer.channel(1,pyb.Timer.ENC_AB,pin=self.pin1)
        self.ch2 = self.timer.channel(2,pyb.Timer.ENC_AB,pin=self.pin2)

    def read(self):
        """Updates and returns the current postion of the encoder
        Updates by taking the difference in counts since the last read and adding them to self.position. Thus, this function has to be run often enough that the encoder doesn't roll over more than twice
        ""
        self.previous = self.current
        self.current = self.timer.counter()
        self.delta = self.current - self.previous

        # if statements to account for encoder roll over
        if self.delta < -10000:
            self.delta += 2**14
        elif self.delta > 10000:
            self.delta -= 2**14
self.position = self.position + self.delta
return(self.position)

def zero(self):
    """Sets the encoder position and other values to zero
    """
    self.position = 0
    self.current = 0
    self.previous = 0
    self.delta = 0
def Move(Destination, probe = False):
    '''Function which utilizes code from the l6470nucleo.py file to drive the stepper motors.

    Function runs the gantry till it stalls (in which case the system throws an error and waits for user input) or it reaches the destination in which case it stops and exits the function.

    @param Destination is the input distance x in millimeters from the end of the X-Beam you want to run to.
    @param probe is True or False, defaulting to False if not called. If True, the function will call the probe function to do one measurement at the end of the move. If False, it will not do that measurement.
    @return Returns one of two things. If the probe == True, the system will return the value read by the probe at the end. If not, then the function will return "Done".''

    # The following three variables (DistancePerStep, xOffset, and xLimit) were all experimentally determined in a not so accurate way. Recommend using these numbers as ball park estimates.
    DistancePerStep = 1/800
    xOffset = 3.2
    xLimit = 775

    # Print information to repl about where the gantry is moving to and if the function will use the probe at the end
    print("Moving Gantry to position x: " + str(Destination))
    print("  Probe at the end of the move? " + str(probe))

    # Check that Destination is not outside of the limits
    if Destination + xOffset <= 0:
        # If the destination is less than -xOffset, then the destination is behind the home position which would push the system against the housing and possible break the switch. Also print an error message to the repl
        print("Error Occured moving the Gantry, Destination behind home")
        ErrGantry.put(1)  # set error flag
        Board1.HardHiZ(2)  # stop the motor
        Board1.GetStatus(2,verbose=0)  # get status to clear the HardHiZ
        return("Error Occured")  # exit function
    elif Destination+xOffset >= xLimit:
        # If the destination is greater than the xLimit, the gantry is running the risk of crashing against the leadscrew bearing end support raiser. Also print an error message to the repl
        print("Error Occured moving the Gantry, destination beyond max")
        ErrGantry.put(1)  # set error flag
        Board1.HardHiZ(2)  # turn off motor
        Board1.GetStatus(2,verbose=0)  # get status to clear the HardHiZ

return("Error Occured")  # exit function

# if no error, print message saying no error to repl
print('" Destination of Gantry within limits"

# Convert Destination in mm to revolutions to steps and print number of
# steps on the repl
Destination = int((Destination+xOffset)/DistancePerStep)
print(" Destination converted to "+str(Destination)+" number of steps")

# Move to the new value of Destination, tell user by printing to repl that
# gantry should be moving
print(" Moving Gantry to Destination")
Board1.GetStatus(2,verbose=0)  # clear errors before telling gantry to go
Board1.GoTo(2,Destination)

# sleep for 1 second to let the system get going, clear errors during that
# 1 second because sometimes the machine thought it stalled out during
# acceleration
utime.sleep(1)
Board1.GetStatus(2,verbose=0)

# Wait for stall or finish flag, print message to repl saying as much
print(" waiting for stall or completion of move command")
while True:
    if Board1.isBusy(2) == False:
        # if finish, exit the loop and print message to repl saying that
        print(" Gantry in position, exit loop")
        Board1.HardHiZ(2)  # stop motor
        Board1.GetStatus(2,verbose=0)  # clear errors if any
        break

    elif Board1.isStalled(2) == True:
        # if stall: print message, stop gantry, throw error and return
        print(" Error Occured moving the Gantry to position. "+\n             "Gantry stalled out")
        ErrGantry.put(1)  # throw error flag
        Board1.HardHiZ(2)  # stop motor
        Board1.GetStatus(2,verbose=0)  # clear error
        return("Error Occured")  # return

# If done moving gantry and probe option is true, take measurement
if probe == True:
    print(" Probe part of Move function called")
    # check the error flags to determine what mode the machine is in.
    # if it is in calibration, then the probe does not need limits on
    # its measurement. If it is leveling, then it does.
    if ErrCalibration.get() == 1:
        # print message saying function is calling Probe.Probe() func for
        # the calibration mode which does not need limits on the probe
        # reading
        print(" Probe called for Calibration Mode")
        reading = Probe.Probe()  # take measurement

    elif ErrAssembly.get() == 1 or ErrLeveling.get() == 1:
        # print message saying func is calling Probe.Probe() for the
        # leveling or assembly mode and that limits are needed
        print(" Probe called for Leveling and Assembly Mode")
# limits were not experimentally determined due to time and that
  # they may be unnecessary as we could not get fingers under the probe
  # while the X-Beam was loaded to interfere with the probe reading.
  # (the limits were put into the program to prevent people's fingers
  # or other stuff from messing up probe readings)
UpprLimit = 1000000
LwrLimit = -1000000
reading = Probe(Limit = True, UpperLimit = UpprLimit, 
                LowerLimit = LwrLimit)

if type(reading) == "Error Occured":
    # Error occurred, but should have been solved as the Probe function
    # shouldn't be able to finish if there was an error.
    print("  Error occurred with probe in the Move()")
    return("Error Occured")
else:
    # no error, return reading
    print("  No error occurred with probe in the Mov()")
    return(reading)

# Else if the gantry is done moving and probe option in false, return
elif probe == False:
    print("  Probe part of Move function not called")
    return("Done")

def Home():
    '''Home the Beam Actuator.
    Function input:
    Board1 is the 16470nucleo.Dual6470 class object that the beam actuator
    is a member of.
    Function output:
    outputs and "Error Occured" if there is an error.
    ...'''
    if Board1.isHome(2)==True:
        # Gantry is homed, exit func. Should add a while loop to release switch
        # and wait till switch is released
        print('  Gantry is already homed')
        return
    # Home Gantry Code. +/- 400 is the max speed of the gantry
    print("  Gantry GoUntil switch command")
    Board1.GoUntil(2,-500)
    utime.sleep(5) # wait for system to get going before making checks
    Board1.GetStatus(2, verbose = 0)
    # Check home status in while loop
    while True:
        if Board1.isBusy(2) == False:
            # gantry at switch, continue.
            break
        elif Board1.isStalled(2) == True:
            print("  Board stalled: "+str(Board1.isStalled(2)))
            # motor stalled, return an error and set flag
            Board1.HardHiZ(2)
            Board1.GetStatus(2,verbose=0)
            print("  Gantry Stalled during Home command")
            print("  ERROR ERROR ERROR, return error")
            ErrGantry.put(1)
            return("Error Occured")
# Release switch for gantry

```
print(" releasing switch for gantry")
Board1.ReleaseSW(2,1)
```

# Check home status in while loop

```
while True:
    if Board1.isBusy(2) == False:
        # Both rail actuators are homed, continue.
        print(" Gantry completed ReleaseSW Command")
        Board1.HardHiZ(2)
        Board1.GetStatus(2,verbose=0)
        return('Done')
    elif Board1.isStalled(2) == True:
        # motor stalled, return an error and set flag
        Board1.HardHiZ(2)
        Board1.GetStatus(2,verbose=0)
        print(" Gantry Stalled during Home command")
        print(" ERROR ERROR ERROR, return error")
        ErrGantry.put(1)
        return("Error Occured")
```

```
def Status():
    '''Function prints out the Gantry's status on the repl
    Function Input:
        The 16470.Dual6470 class object that correlates to the
        Gantry
    Function Output:
        None'''
    Board1.GetStatus(2,verbose = 1)
```

# Import Modules and Objects used in this file when the file is imported
```
import time
from setup import ErrGantry , ErrCalibration, ErrLeveling, ErrAssembly, Board1
import Probe
```
def Song():
    # This function checks the piano board's last four switches (the pins called Note1 to Note4) and converts those four binary inputs into an integer. With four switches available, the range of corresponding integer inputs should be 0 to 15 (1-16 if you add 1 to the final number)
    # Amendment: due to issues with the piano switch board, this function was basically cut out. If the user still wishes to use the piano switch board, just delete XBeam.put(500) and uncomment the rest of the function
    XBeam.put(500)

    # Get Values from the switches and store into variables a, b, c, and d
    # NOTE: as with Note0, reading 0 means true on the switch
    # if Note0.value() == 0:
    #     a = 1
    # else:
    #     a = 0
    # if Note1.value() == 0:
    #     b = 1
    # else:
    #     b = 0
    # if Note2.value() == 0:
    #     c = 1
    # else:
    #     c = 0
    # if Note3.value() == 0:
    #     d = 1
    # else:
    #     d = 0
    # if Note4.value() == 0:
    #     e = 1
    # else:
    #     e = 0

    # Calculating a binary number from four binary input
    # Number = (a*2**4 + b*2**3 + c*2**2 + d*2**1 + e*2**0) + 1

    # If the user has selected the combination of switches which correspond to 1-16, than input the X-Beam Length into the buffer XBeam that corresponds to that input. Additionally, if there is no input, than the system will return an error.
    # if Number == 1:
    #     Length = 500
    # elif Number == 2:
    #     ErrSong.put(1)
    # elif Number == 3:
    #     ErrSong.put(1)
#  elif Number == 4:
#    ErrSong.put(1)
#  
#  elif Number == 5:
#    ErrSong.put(1)
#  
#  elif Number == 6:
#    ErrSong.put(1)
#  
#  elif Number == 7:
#    ErrSong.put(1)
#  
#  elif Number == 8:
#    ErrSong.put(1)
#  
#  elif Number == 9:
#    ErrSong.put(1)
#  
#  elif Number == 10:
#    ErrSong.put(1)
#  
#  elif Number == 11:
#    ErrSong.put(1)
#  
#  elif Number == 12:
#    ErrSong.put(1)
#  
#  elif Number == 13:
#    ErrSong.put(1)
#  
#  elif Number == 14:
#    ErrSong.put(1)
#  
#  elif Number == 15:
#    ErrSong.put(1)
#  
#  elif Number == 16:
#    ErrSong.put(1)
#  
#  # Note that with 5 switches, you can have up to 32 options. Add more elif statements.
#  
if ErrSong.get() == 0:
    XBeam.put(Length)
    return("No Error")
#  
elif ErrSong.get() == 1:
    f = open("Error Report.txt","w")
    f.write('''An error occured during the initialization phase\n\rThe switch combination of the piano switch board does not correspond to any of\n\rregistered X-Beam lengths.\n\rRecommendations:\n\r1) Check your switch combination\n\r2) Open ImportSong.py and check that the switch combination you've selected\n\rhas a corresponding X-Beam length.\n\rNote: Please remember to check that if you add an X-Beam Length to\n\rImportSong.py, check that the corresponding Calibration and BoltPattern\n\r''')
def BoltPattern():
    '''This function imports a bolt pattern file based on the value of buffer X-Beam. This is done by using the Python try function to attempt to import the bolt pattern file associated with the input given which should be the length of the X-Beam being worked upon. The input is from the ImportSwitchBoard() function. Example of how this would work is shown below.

    # Value in XBeam buffer is 500
    Input = XBeam.get()
    String = "BoltPattern"+str(Input)+".csv"
    i.e. BoltPattern500.csv

    After figuring out the file name, it tries to import the file. If the file exists, the function returns a list of integers. If the file does not exist, then the program returns an error message

    @input Function reads buffer X-Beam for the necessary input which should be the length of the X-Beam
    @return Function returns a string if there was an error and writes a message to the Error Report text file. If there was no error, the function returns a list with two sub-lists, one containing strings of which side is to be bolted and the other containing positions relative to the gantry's zero position of where to move to.
    ...""

    print("Beginning to import BoltPatternXXX.csv")
    # Getting length value of the X-Beam from the buffer XBeam
    Input = XBeam.get()
    print("Working X-Beam "+str(Input))

    # Creating the name of the file to be imported
    FileName = "BoltPattern"+str(Input)+".csv"
    print("File name associated with X-Beam: "+FileName)

    # a function unique binary error flag indicating which error has occurred.
    error = 0

    '''Open Error Report for editing'''
    f = open("Error Report.txt","w")
    print("File Opened, checking values")

    try:
        '''Attempt to import the BoltPattern file indicated by the input.'''
        with open(FileName,'r') as file:
            '''File does exist, process the data as indicated'''
            i = 1
            print("File Opened, splitting values into two separate lists")
            for line in file:
                line = line.split(',')
                line[-1] = (line[-1].replace("\n",""))
                line[-1] = (line[-1].replace("\r",""))
                if i == 1:
BoltSide = line

elif i == 2:
    BoltData = line
    print("  Line "+str(i)+" processed")
    i = i+1

except OSError:
    '''File doesn't exist, write to Error Report'''
    print("  Unable to open BoltPatternXXX.csv file")
    ErrBoltCsv.put(1)
    string = "Error, Missing "+FileName
    f.write(string)
    f.close()
    return("Missing file error")

'''Check the values in the BoltSide list. If there is an error, write to the Error Report text file.''
print("Checking items indicating sides in the BoltSide list")
i = 1
for item in BoltSide:
    if item != "R" and item != "L":
        ErrBoltCsv.put(1)
        string = "Error in "+FileName", row 1 item "+str(i)+": "+\str(item)+" is not a valid input"
        f.write(string)
        f.write('\r\n')
        error = 1
        print("  Item "+str(i)+" checked, "+string)
    else:
        print("  Item "+str(i)+" checked, no errors")
        i = i+1

'''Check the values in the BoltData list. If there is an error, write to the Error Report text file.''
print("Checking items indicating positions in the BoltData list")
i = 1
for item in BoltData:
    try:
        float(item)
    except ValueError:
        ErrBoltCsv.put(1)
        string = "Error in "+FileName", row 2 item "+str(i)+": "+\str(item)+" is not a valid input"
        f.write(string)
        f.write('\r\n')
        error = 1
        print("  Item "+str(i)+" checked, "+string)
    else:
        print("  Item "+str(i)+" checked, no errors")
        i = i+1

'''If there were no issues in the lists, return the two lists'''
if error == 0:
    print("  No Errors importing "+FileName)
    return([BoltSide,BoltData])
elif error != 0:
    print("  An error has occured, exiting function")
def Calibration():
    '''This function imports a calibration file based on the value stored in
    the XBeam buffer. This is done by using the Python try function to attempt
    to import the calibration file associated with the input given which should
    be the length of the X-Beam being worked upon. The input is from the
    ImportSwitchBoard() function. Example of how this would work is shown
    below.

    # Value in XBeam buffer is 500
    Input = XBeam.get()
    String = "Calibration"+str(Input)+".csv"
    i.e. Calibration500.csv

    After figuring out the file name, it tries to import the file. If the file
    exists, the function returns a list of integers. If the file does not
    exist, then the program returns an error message

    @input This function takes inputs in by reading the buffer XBeam for the
    necessary information which is an integer indicating XBeam length
    @return This function returns a string if there was an error. If not, the
    function returns a list of values'''

    print("Beginning to import CalibrationXXX.csv")

    # Getting Length value of the X-Beam from the buffer XBeam
    Input = XBeam.get()
    print("Working X-Beam "+str(Input))

    # Creating the name of the file to be imported
    FileName = "Calibration"+str(Input)+".csv"
    print("File name associated with X-Beam: "+FileName)

    # a function unique binary error flag indicating which error has occurred.
    error = 0

    try:
        '''Attempt to import the Calibration file indicated by the input.''
        with open(FileName,'r') as file:
            '''File does exist, process the data as indicated'''
            print("File Opened, processing data and checking items")
            for line in file:
                String = str(line)  # Bring in line of info from csv
                List = String.split(',')  # Split Line of info at the comma
                String = []
                i = 0
                for item in List:
                    '''Go through each item in List of info and try to make each
                    item a float. If not, return an error message indicating
                    which item was invalid.''
                    try:
                        '''Try to turn the item into a float'''
                        List[i] = float(item)
                        print("Item "+str(i)+" checked and is valid")
except ValueError:
    '''Error indicating invalid input'''
    String.append("Error in "+FileName+ " Line "+str(i+1)+ ": "+item+" is not a valid input")
    ErrCalCsv.put(1)
    print("Error in "+FileName+ " Line "+str(i+1)+ ": "+item+" is not a valid input")
    error = 1
except TypeError:
    '''Error indicating invalid input'''
    String.append("Error in "+FileName+ " Line "+str(i+1)+ ": "+item+" is not a valid input")
    ErrCalCsv.put(1)
    print("Error in "+FileName+ " Line "+str(i+1)+ ": "+item+" is not a valid input")
    error = 1

i = i+1
except OSError:
    '''Exception to report that the file is missing'''
    print("Unable to open CalibrationXXX.csv file")
    String = ["Error,Missing Calibration"+str(Input)+".csv File"]
    ErrCalCsv.put(1)
    error = 1

if error == 0:
    print("No error importing "+FileName)
    return(List) # return the final list of values

elif error != 0:
    f = open("Error Report.txt","w")
    for item in String:
        f.write(item)
    f.close()
    print("Error Occured importing "+FileName)
    return("Error Occured")
else:
    return('''Error Occured, error flag in ImportCalibration was not 0 or 1
    at the end of the function''')

from setup import ErrCalCsv, ErrBoltCsv, XBeam
This file contains a driver for a dual L6470 stepper driver board which is part of the Nucleo package from ST Micro. It can control each of the two L5470 chips which are on the board.

In order to use a stepping motor, the constants @c MAX_SPEED, @c ACCEL, @e etc. need to be tuned for that motor using the methods shown at @c http://www.st.com/resource/en/application_note/dm00061093.pdf.

@author John Ridgely, John Barry, Anthony Lombardi, Whittaker Hamill, Sam Artho-Bentz, and Robert Tam

```
import pyb
import time

# Maximum speed for the motor; default 65, or ~992 steps/s
MAX_SPEED = 0

# Acceleration for the motor; default 138, or 2008 steps/s^2
ACCEL = const (5)

# Deceleration for the motor; default 138, or 2008 steps/s^2
DECEL = const (12)

# The K_val constant for registers 0x09, 0x0A, 0x0B, 0x0C; default 0x29 = 41
K_VAL = const (90)  # Calculated 200??

# Intersection speed for register 0x0D; default 0x0408 = 1032
INT_SPEED = const (1032)

# Startup slope for register 0x0E; default 0x19 = 25
ST_SLP = const (25)

# Final slope for registers 0x0F, 0x10; default 0x29 = 41
FN_SLP = const (397)

# Number of Microsteps, num which are powers of 2 up to 128 are acceptable.
STEP_SEL = const (8)

# Define SYNC_ENable bitmask. 0x80 for High, 0x00 for Low
SYNC_EN = const(0x00)

# SYNC_SEL modes. Datasheet pg 46 for full information
SYNC_SEL = const(0x10)

# STALL_TH. Default of 2.03A
STALL_TH = const(16)

class Dual6470:
    # This class implements a driver for the dual L6470 stepping motor driver
    # chips on a Nucleo IHM02A1 board. The two driver chips are connected in
    # SPI daisy-chain mode, which makes communication a bit convoluted.
    #
    # NOTE: One solder bridge needs to be moved for the IHM02A1 to work
```
# Elec
# acce
# RESER
# ther
# dece
# OCD
# Alar
# STAL
# star
# RESE
# ADC
# acce
# Step
# cons
# dece
# hold

### DICTIONARIES ===

The board is shipped is not the one which MicroPython uses unmodified MicroPython. Bridge SB34 must be disconnected, and

\[ \text{self} \]

```

def __init__ (self, spi_object, cs_pin, stby_rst_pin):
    # === DICTIONARIES ===
    """ Dictionary of available registers and their addresses. ""
    self.REGISTER_DICT = {} # ADDR | LEN | DESCRIPTION | xRESET | Write
    self.REGISTER_DICT[ 'ABS_POS' ] = [0x01, 22] # current pos 000000 | S
    self.REGISTER_DICT[ 'EL_POS' ] = [0x02, 9] # Electrical pos 00 | S
    self.REGISTER_DICT[ 'MARK' ] = [0x03, 22] # mark position 000000 | W
    self.REGISTER_DICT[ 'SPEED' ] = [0x04, 20] # current speed 000000 | R
    self.REGISTER_DICT[ 'ACC' ] = [0x05, 12] # accel limit 08A | W
    self.REGISTER_DICT[ 'DEC' ] = [0x06, 12] # decel limit 08A | W
    self.REGISTER_DICT[ 'MAX_SPEED' ] = [0x07, 10] # maximum speed 041 | W
    self.REGISTER_DICT[ 'MIN_SPEED' ] = [0x08, 13] # minimum speed 0 | S
    self.REGISTER_DICT[ 'FS_SPD' ] = [0x09, 10] # full-step speed 027 | W
    self.REGISTER_DICT[ 'KVAL_HOLD' ] = [0x0A, 8] # holding Kval 29 | W
    self.REGISTER_DICT[ 'KVAL_RUN' ] = [0x0A, 8] # const speed Kval 29 | W
    self.REGISTER_DICT[ 'KVAL_ACC' ] = [0x0B, 8] # accel start Kval 29 | W
    self.REGISTER_DICT[ 'KVAL_DEC' ] = [0x0C, 8] # decel start Kval 29 | W
    self.REGISTER_DICT[ 'INT_SPEED' ] = [0x0D, 14] # intersect speed 0408 | H
    self.REGISTER_DICT[ 'ST_SLP' ] = [0x0E, 8] # start slope 19 | H
    self.REGISTER_DICT[ 'FN_SLP_ACC' ] = [0x0F, 8] # accel end slope 29 | H
    self.REGISTER_DICT[ 'FN_SLP_DEC' ] = [0x10, 8] # decel end slope 29 | H
    self.REGISTER_DICT[ 'K_THERM' ] = [0x11, 4] # therm comp factor 0 | H
    self.REGISTER_DICT[ 'ADC_OUT' ] = [0x12, 5] # ADC output XX | S
    self.REGISTER_DICT[ 'OCD_TH' ] = [0x13, 4] # OCD threshold 8 | W
    self.REGISTER_DICT[ 'STALL_TH' ] = [0x14, 7] # STALL threshold 40 | W
    self.REGISTER_DICT[ 'STEP_MODE' ] = [0x16, 8] # Step mode 7 | H
    self.REGISTER_DICT[ 'ALAR_DM' ] = [0x17, 8] # Alarm enable FF | S
    self.REGISTER_DICT[ 'CONFIG' ] = [0x18, 16] # IC configuration 2E88 | H
    self.REGISTER_DICT[ 'STATUS' ] = [0x19, 16] # Status XXXX | X
    self.REGISTER_DICT[ 'RESERVED A' ] = [0x1A, 0] # RESERVED | X
    self.REGISTER_DICT[ 'RESERVED B' ] = [0x1B, 0] # RESERVED | X
    # Write: X = unreadable, W = Writable (always),
    # S = Writable (when stopped), H = Writable (when Hi-Z)
    """ Dictionary for the STATUS register. Contains all error flags,
    as well as basic motor state information. ""
    self.STATUS_DICT = {} # [ NAME | OK/DEFAULT VALUE ]
    self.STATUS_DICT[14] = [ 'STEP_LOSS_B',1 ] # stall detection on bridge B
    self.STATUS_DICT[13] = [ 'STEP_LOSS_A',1 ] # stall detection on bridge A
    self.STATUS_DICT[12] = [ 'OVERCURRENT',1 ] # OCD, overcurrent detection
    self.STATUS_DICT[11] = [ 'HEAT_SHUTDOWN',1 ] # TH_SD, thermal shutdown
    self.STATUS_DICT[10] = [ 'HEAT_WARN ',1 ] # TH_WN, thermal warning
    self.STATUS_DICT[ 9 ] = [ 'UNDEPVOLT ',1 ] # UVLO, low drive supply voltage
    self.STATUS_DICT[ 8 ] = [ 'WRONG_CMD ',0 ] # Unknown command
    self.STATUS_DICT[ 7 ] = [ 'NOTPERF_CMD',0 ] # Command can't be performed
    self.STATUS_DICT[ 3 ] = [ 'SWITCH_EDGE',0 ] # SW_EVN, signals switch falling edge
    self.STATUS_DICT[ 2 ] = [ 'SWITCH_FLAG',0 ] # switch state. 0=open, 1=grounded
    self.STATUS_DICT[15] = [ 'STEP_MODE',0 ] # 1=step-clock mode, 0=normal
```

---

193
self.STATUS_DICT[4] = ['DIRECTION', 1]  # 1=forward, 0=reverse
self.STATUS_DICT[6] = ['MOTOR_STAT', 0]  # two bits: 00=stopped, 01=accel
               #        10=decel,    11=const spd
self.STATUS_DICT[1] = ['BUSY', 1]  # low during movement commands
self.STATUS_DICT[0] = ['Hi-Z', 1]  # 1=hi-Z, 0=motor active

""" Dictionary for Application Commands. See L6470 Programming manual Pg 56
for information on usage. These commands must be OR'd with the
values for use.
"""

self.COMMAND_DICT = {}  # [ NAME       | Command Hex Code | Action ]
self.COMMAND_DICT['SetParam'] = 0x00  # Writes VALUE in PARAM register
self.COMMAND_DICT['GetParam'] = 0x20  # Returns the stored value in PARAM register
self.COMMAND_DICT['Run'] = 0x50  # Sets the target speed and direction
self.COMMAND_DICT['StepClock'] = 0x58  # Puts the device in Step-clock mode and imposes di
self.COMMAND_DICT['GoTo'] = 0x60  # Goes to specified ABS_POS (min path)
self.COMMAND_DICT['GoTo_DIR'] = 0x68  # Goes to specified ABS_POS (forced direction)
self.COMMAND_DICT['GoUntil'] = 0x82  #
self.COMMAND_DICT['ReleaseSW'] = 0x92  #
self.COMMAND_DICT['GoHome'] = 0x70  #
self.COMMAND_DICT['GoMark'] = 0x78  #
self.COMMAND_DICT['ResetPos'] = 0xD8  #
self.COMMAND_DICT['ResetDevice'] = 0xC0  #
self.COMMAND_DICT['SoftStop'] = 0xB0  #
self.COMMAND_DICT['HardStop'] = 0xB8  #
self.COMMAND_DICT['SoftHiZ'] = 0xA0  #
self.COMMAND_DICT['HardHiZ'] = 0xA8  #
self.COMMAND_DICT['GetStatus'] = 0xD0  #

## Initialize the Dual L6470 driver. The modes of the @c CS and
# @c STBY/RST pins are set and the SPI port is set up correctly.
# @param spi_object A SPI object already initialized. used to talk to the
#     driver chips, either 1 or 2 for most Nucleos
# @param cs_pin The pin which is connected to the driver chips' SPI
#     chip select (or 'slave select') inputs, in a pyb.Pin object
# @param stby_rst_pin The pin which is connected to the STBY/RST inputs, in a pyb.Pin object
## The CPU pin connected to the Chip Select (AKA 'Slave Select') pin
#     of both the L6470 drivers.
self.cs_pin = cs_pin

## The CPU pin connected to the STBY/RST pin of the 6470's.
self.stby_rst_pin = stby_rst_pin

## The SPI object, configured with parameters that work for L6470's.
# pyb.SPI (spi_number, mode=pyb.SPI.MASTER,
#    baudrate=2000000, polarity=1, phase=1,
#    bits=8, firstbit=pyb.SPI.MSB)
self.spi = pyb.SPI (spi_object, mode=pyb.SPI.MASTER,
               baudrate=2000000, polarity=1, phase=1,
               bits=8, firstbit=pyb.SPI.MSB)

# Make sure the CS and STBY/RST pins are configured correctly
self.cs_pin.init (pyb.Pin.OUT_PP, pull=pyb.Pin.PULL_NONE)
self.cs_pin.high ()
self.stby_rst_pin.init (pyb.Pin.OUT_OD, pull=pyb.Pin.PULL_NONE)
# Reset the L6470's

stby_rst_pin.low ()
time.sleep (0.01)
stby_rst_pin.high ()

...  

# Set the registers which need to be modified for the motor to go
# This value affects how hard the motor is being pushed
K_VAL = 80
self._set_par_1b ('KVAL_HOLD', K_VAL)
self._set_par_1b ('KVAL_RUN', K_VAL)
self._set_par_1b ('KVAL_ACC', K_VAL)
self._set_par_1b ('KVAL_DEC', K_VAL)

# Speed at which we transition from slow to fast V_B compensation
INT_SPEED = 1032 #3141
self._set_par_2b ('INT_SPEED', INT_SPEED)

# Acceleration and deceleration back EMF compensation slopes
ST_SLP = 25
self._set_par_1b ('ST_SLP', ST_SLP)
self._set_par_1b ('FN_SLP_ACC', ST_SLP)
self._set_par_1b ('FN_SLP_DEC', ST_SLP)

# Set the maximum speed at which motor will run
MAX_SPEED = 1
self._set_par_2b ('MAX_SPEED', MAX_SPEED)

# Set the maximum acceleration and deceleration of motor
ACCEL = 5
DECEL = 12
self._set_par_2b ('ACC', ACCEL)
self._set_par_2b ('DEC', DECEL)

# Set the number of Microsteps to use
SYNC_EN = 0x00
SYNC_SEL = 0x10
STEP_SEL = 8
self._set_MicroSteps (SYNC_EN, SYNC_SEL, STEP_SEL)

# Set the Stall Threshold
STALL_TH = 64
self._setStallThreshold(STALL_TH)

# Set motors in high impedance mode
self.SoftHiZ(1)
self.SoftHiZ(2)
self.GetStatus(1,verbose = 0)
self.GetStatus(2,verbose = 0)

...

def _setStallThreshold(self, value):
    ## Sets the threshold back emf value for the board to stall at.
    # @param value is an integer input from 1 to 64 which corresponds to a
    # current of x*31.25mA where x is the input from 1 to 16
    self._set_par_1b('STALL_TH', value)
def _set_MicroSteps(self, SYNC_Enable, SYNC_Select, num_STEP):
    ## Set the number of Microsteps to use, the SYNC output frequency, and the
    ## SYNC ENABLE bit.
    ## @param SYNC_Enable A 1-bit integer which determines the behavior of the
    ## BUSY/SYNC output.
    ## @param SYNC_Select A 3-bit integer which determines SYNC output frequency
    ## @param num_STEP the integer number of microsteps, numbers which are
    ## powers of 2 up to 128 are acceptable.
    for stepval in range(0,8):
        #convert num_STEPS to 3-bit power of 2
        if num_STEP == 1:
            break
        num_STEP = num_STEP >> 1
        if SYNC_Enable == 1:
            SYNC_EN_Mask = 0x80
        else:
            SYNC_EN_Mask = 0x00
        self._set_par_1b('STEP_MODE', SYNC_EN_Mask|stepval|SYNC_Select)
    
    def _read_bytes(self, num_bytes):
        ## Read a set of arguments which have been sent by the L6470's in
        ## response to a read-register command which has already been
        ## transmitted to the L6470's. The arguments are shifted into the
        ## integers supplied as parameters to this function.
        ## @param num_bytes The number of bytes to be read from each L6470
        data_1 = 0
        data_2 = 0

        # Each byte which comes in is put into the integer as the least
        # significant byte so far and will be shifted left to make room for
        # the next byte
        for index in range(num_bytes):
            self.cs_pin.low ()
            data_1 <<= 8
            data_1 |= (self.spi.recv (1))[0]
            data_2 <<= 8
            data_2 |= (self.spi.recv (1))[0]
            self.cs_pin.high ()
            #print("in read bytes motor 1 is " + str(bin(data_1)))
            #print("in read bytes motor 2 is " + str(bin(data_2)))
        return ([data_1, data_2])

    def _get_params(self, command_byte, recv_bytes):
        ## Send one byte to each L6470 as a command and receive two bytes
        ## from each driver in response.
        ## @param command_byte The byte which is sent to both L6470's
        ## @param recv_bytes The number of bytes to receive: 1, 2, or 3
        # Send the command byte, probably a read-something command
        self._sndb(command_byte, command_byte)

        # Receive the bytes from the driver chips
        [data_1, data_2] = self._read_bytes (recv_bytes)
return([[data_1, data_2]])

def _set_par_2b (self, reg_name, num):
    ## Set a parameter to both L6740 drivers, in a register which needs
    # two bytes of data.
    # @param reg_name (string) The register to be set
    # @param num The two-byte number to be put in that register
    reg = self.REGISTER_DICT[reg_name][0]
    self._sndbs (reg, reg)
    highb = (num >> 8) & 0x03
    self._sndbs (highb, highb)
    lowb = num & 0xFF
    self._sndbs (lowb, lowb)
    # utime.sleep(1)

def _set_par_1b (self, reg_name, num):
    ## Set a parameter to both L6740 drivers, in a register which needs
    # one byte of data.
    # @param reg_name (string) The register to be set
    # @param num The one-byte number to be put in that register
    reg = self.REGISTER_DICT[reg_name][0]
    self._sndbs (reg, reg)
    self._sndbs (num, num)
    # utime.sleep(1)

def _sndbs (self, byte_1, byte_2):
    ## Send one command byte to each L6470. No response is expected.
    # @param byte_1 The byte sent first; it goes to the second chip
    # @param byte_2 The byte sent second, to go to the first chip
    self.cs_pin.low ()
    self.spi.send (byte_1)
    self.spi.send (byte_2)
    self.cs_pin.high ()

def _cmd_1b (self, motor, command):
    ## Send a command byte only to one motor. The other motor is sent a NOP
    # command (all zeros).
    # @param motor The number, 1 or 2, of the motor to receive the command
    if motor == 1:
        self._sndbs (command, 0x00)
    elif motor == 2:
        self._sndbs (0x00, command)
    else:
        raise ValueError ('Invalid L6470 motor number; must be 1 or 2')

def _cmd_3b (self, motor, command, data):
    ## Send a command byte plus three bytes of associated data to one of
    # the motors.
    # @param motor The motor to receive the command, either 1 or 2
    # @param command The one-byte command to be sent to one motor
    # @param data The data to be sent after the command, in one 32-bit
    # integer
    # Break the integer containing the data into three bytes
byte_2 = (data >> 16) & 0x0F
byte_1 = (data >> 8) & 0xFF
byte_0 = data & 0xFF

# Send commands to one motor, NOP (zero) bytes to the other
if motor == 1:
    self._sndbs(command, 0x00)
    self._sndbs(byte_2, 0x00)
    self._sndbs(byte_1, 0x00)
    self._sndbs(byte_0, 0x00)
elif motor == 2:
    self._sndbs(0x00, command)
    self._sndbs(0x00, byte_2)
    self._sndbs(0x00, byte_1)
    self._sndbs(0x00, byte_0)
else:
    raise ValueError('Invalid L6470 motor number; must be 1 or 2')

# ----------------- L6470 Built-in Functions-----------------

def Run (self, motor, steps_per_sec):
    ## Tell the motor to run at the given speed. The speed may be
    # positive, causing the motor to run in direction 0, or negative,
    # causing the motor to run in direction 1.
    # @param motor Which motor to move, either 1 or 2
    # @param steps_per_sec The number of steps per second at which to
    #   go, up to the maximum allowable set in @c MAX_SPEED

    # Figure out the direction from the sign of steps_per_sec
    if steps_per_sec < 0:
        direc = 0
        steps_per_sec = -steps_per_sec
    else:
        direc = 1

    # Convert to speed register value: multiply by ~(250ns)(2^28)
    steps_per_sec *= 67.108864
    steps_per_sec = int (steps_per_sec)

    # Have the _cmd_3b() method write the command to a driver chip
    self._cmd_3b(motor, self.COMMAND_DICT['Run'] | direc,
                 steps_per_sec & 0x0000FFFF)

# def StepClock(self, direc = None):
#     '''Needs Writing'''

# ----------------- L6470 Built-in Functions-----------------

def Move (self, motor, steps, direc = None):
    ## Tell motor driver @c motor to move @c num_steps in the direction
    # @c direction.
# @param motor Which motor to move, either 1 or 2
# @param steps How many steps to move, in a 20 bit number
# @param direc The direction in which to move, either 0 for one
t# way or nonzero for the other; if unspecified, the sign of the
t# number of steps will be used, positive meaning direction 0

# Figure out the intended direction
if direc == None:
    if steps <= 0:
        direc = 1
        steps = -steps
    else:
        direc = 0
else:
    if direc != 0:
        direc = 1

# Call the _cmd_3b() method to do most of the work
self._cmd_3b(motor, self.COMMAND_DICT['Move'] | direc, steps & 0x003FFFFF)

```python
'''---
----------------------------------------------------'''
```

```python
def GoTo(self, motor, pos_steps):
    ## This command asks the specified motor to move to the specified
    # position.
    # @param motor Which motor to move, either 1 or 2
    # @param pos_steps The position to which to move, in absolute steps

    # Call the _cmd_3b() method to do most of the work
    self._cmd_3b(motor, self.COMMAND_DICT['GoTo'], pos_steps & 0x003FFFFF)

    ```python
    '''---
    ----------------------------------------------------'''
```

```python
# def GoTo_DIR(self, motor, pos_steps, direc):
#   `''Needs Writing
#   # Not needed for our code - Whittaker'''

    ```python
    '''---
    ----------------------------------------------------'''
```

```python
def GoUntil(self, motor, steps_per_sec, direc = None):
    `''Goes until a switch has been hit.
    @param motor is the value 1 or 2 of the motor on the board
    being ordered to move.
    @param steps_per_sec is the speed at which you want the
    motor to move in steps per second.
    @param direc indicates the direction you want. This is
    superceded by the direction indicated by the
    param steps_per_sec'''

    # Figure out the direction from the sign of steps_per_sec
    if steps_per_sec < 0:
        direc = 0
        steps_per_sec = -steps_per_sec
    else:
        direc = 1

    ```python
    '''---
    ----------------------------------------------------'''```
# Convert to speed register value: multiply by ~\((250\text{ns})\cdot 2^{28}\)
steps_per_sec *= 67.108864
steps_per_sec = int(steps_per_sec)

# Have the _cmd_3b() method write the command to a driver chip
self._cmd_3b (motor, self.COMMAND_DICT['GoUntil'] | direc, steps_per_sec & 0x000FFFFF)

def ReleaseSW(self, motor, direc):
    '''The ReleaseSW command produces a motion at minimum speed imposing a forward
(DIR = '1') or reverse (DIR = '0') rotation. When SW is released (opened), the ABS_POS
register is reset (ACT = '0') or the ABS_POS register value is copied into the MARK register
(ACT = '1'); the system then performs a HardStop command.''

    self._cmd_1b(motor, self.COMMAND_DICT['ReleaseSW'] | direc)

# def GoHome(self, motor):
#     '''Same as Goto(0), not needed'''

# def GoMark(self, motor):
#     '''The GoMark command keeps the BUSY flag low until the MARK position is reached. This
# command can be given only when the previous motion command has been completed
# (BUSY flag released).'''

def ResetPos (self, motor):
    ## This command sets the given motor driver's absolute position register
    # to zero.
    # @param motor The motor whose position is to be zeroed, either 1 or 2
    self._cmd_1b (motor, self.COMMAND_DICT['ResetPos'])

def ResetDevice(self, motor):
    self._cmd_1b (motor, self.COMMAND_DICT['ResetDevice'])

# def SoftStop (self, motor):
#     ## Tell a motor driver to decelerate its motor and stop wherever it ends
#     # up after the deceleration.
#     # @param motor Which motor is to halt, 1 or 2
#     self._cmd_1b (motor, self.COMMAND_DICT['SoftStop'])

# def HardStop (self, motor):
Tell the specified motor to stop immediately, not even doing the usual smooth deceleration. This command should only be used when the compost is really hitting the fan because it asks for nearly infinite acceleration of the motor, and this will probably cause the motor to miss some steps and have an inaccurate position count.

@param motor Which motor is to halt, 1 or 2

self._cmd_1b (motor, self.COMMAND_DICT['HardStop'])

---

```python
def SoftHiZ (self, motor):
    ## Tell the specified motor to decelerate smoothly from its motion, then put the power bridges in high-impedance mode, turning off power to the motor.
    # @param motor Which motor is to be turned off, 1 or 2
    self._cmd_1b (motor, self.COMMAND_DICT['SoftHiZ'])
```

---

```python
def HardHiZ (self, motor):
    ## Tell the specified motor to stop and go into high-impedance mode (no current is applied to the motor coils) immediately, not even doing the usual smooth deceleration. This command should only be used when the compost is really hitting the fan because the motor is put into a freewheeling mode with no control of position except for the small holding torque from the magnets in a PM hybrid stepper, and this will probably cause the motor to miss some steps and have an inaccurate position count.
    # @param motor Which motor is to be turned off, 1 or 2
    self._cmd_1b (motor, self.COMMAND_DICT['HardHiZ'])
```

---

```python
def GetStatus (self, motor, verbose=1):
    status = self._get_params(self.COMMAND_DICT['GetStatus'], 2)
    if verbose:
        self.Print_Status(motor, status)
    return status
```

---

## Secondary Functions

```python
# def getPositions (self, motor):
#    ## Get the positions stored in the drivers for the selected motor. Each driver stores its motor's position in a 22-bit register. If only one position is needed, it's efficient to get both because the drivers are daisy-chained on the SPI bus, so we have to send two commands and read a bunch of bytes of data anyway.
#    # @return The current positions of the selected motor
#    # Read (command 0x20) register 0x01, the current position
#    [data_1, data_2] = self._get_params (self.COMMAND_DICT['GetParam']|self.REGISTER_DICT['ABS_POS'], 2)
#    print("motor 1 is " + str(bin(data_1)))
#    print("motor 2 is " + str(bin(data_2)))
```
# Sign-extend the signed absolute position numbers to 32 bits
# ''' We Don't need to sign extend the number
# if data_1 & 0x00400000:
#   data_1 |= 0xFF800000
# else:
#   data_1 &= 0x001FFFFF
# if data_2 & 0x00400000:
#   data_2 |= 0xFF800000
# else:
#   data_2 &= 0x001FFFFF
#...
# if motor == 1:
#   data = data_1
# else:
#   data = data_2
# return (self.GetTwosComplement(data, 22))

def GetTwosComplement(data, length):
    if (data & (1 << (length - 1))) != 0:
        data = data - (1 << length)
        return (data)
    return (data)

def isStalled(self, motor):
    status = self.GetStatus(1, verbose = 0)
    if ((status[motor-1]&(1<<13) == 0) and (status[motor-1]&(1<<14) == 0)):
        # if status[2-1] & (1<<6) == True and status[2-1] & (1<<5) == True:
        return True
    else:
        return False

def isBusy(self, motor):
    status = self.GetStatus(1, verbose = 0)
    if (status[motor-1] & (1<<1)) == 0:
        return(True)
    else:
        return(False)

def isHome(self, motor):
    status = self.GetStatus(1, verbose = 0)
    if status[motor-1] & (1<<2):
        return(True)
    else:
        return(False)

def Print_Status(self, motor, status):
    ''' Formatted printing of status codes for the driver.

    @arg @c motor (int): the motor which the status is representing.
    @arg @c status (int): the code returned by a GetStatus call.
    '''
    # check error flags
    print ('Driver ', str(motor), ' Status: ') #, bin(status))
    print(status)
    for bit_addr in range(7,15):
        print(’ Flag ’, self.STATUS_DICT[bit_addr][0], ': ', end='')
# we shift a 1 to the bit address, then shift the result down again
if ((status[motor-1] & 1<<bit_addr)>>bit_addr)==self.STATUS_DICT[bit_addr][1]:
    print("ok")
else:
    print("Alert!")

# check SCK_MOD
if status[motor-1] & (1<<15):
    print('  Step-clock mode is on.')
else:
    print("  Step-clock mode is off.")

# check MOT_STATUS
if status[motor-1] & (1<<6):
    if status[motor-1] & (1<<5):
        print("  Motor is at constant speed.")
    else:
        print("  Motor is decelerating.")
else:
    if status[motor-1] & (1<<5):
        print("  Motor is accelerating.")
    else:
        print("  Motor is stopped.")

# check DIR
if status[motor-1] & (1<<4):
    print("  Motor direction is set to forward.")
else:
    print("  Motor direction is set to reverse.")

# check BUSY
if not (status[motor-1] & (1<<1)):
    print("  Motor is busy with a movement command.")
else:
    print("  Motor is ready to recieve movement commands.")

# check HiZ
if status[motor-1] & 1:
    print("  Bridges are in high-impedance mode (disabled).")
else:
    print("  Bridges are in low-impedance mode (active).")

# check SW_EVEN flag
if status[motor-1] & (1<<3):
    print("  External switch has been clicked since last check.")
else:
    print("  External switch has no activity to report.")
# check SW
if status[motor-1] & (1<<2):
    print("  External switch is closed (grounded).")
else:
    print("  External switch is open.")

#setLoSpdOpt(boolean enable);
#configSyncPin(byte pinFunc, byte syncSteps);
#configStepMode(byte stepMode);
Appendix P6: main.py

# -*- coding: utf-8 -*-
##
# @file main.py
# @author Robert Tam
# This program was written for an ME Senior Project.
# -*- coding: utf-8 -*-

# importing modules for buffers and functions
print("Program initializing")
import utime
import Gantry
import Probe
import BeamActuator
import RailAct
import setup
from setup import ErrInit,
    ErrCalibration,
    ErrLeveling,
    ErrBoltCsv,
    ErrCalCsv,
    ErrGantry,
    ErrProbe,
    ErrRailActL,
    ErrRailActR,
    ErrBeamAct,
    ErrFileCheck,
    ErrSong,
    XBeam,
    zero_flags,
    FileCheck,
    DCMotor,
    Solenoid,
    Mode,
    RedLED,
    GreenLED,
    YellowLED,
    Buzzer,
    Go,
    Board1,
    Board2

def Lights_Sound_Off():
    """Turns all LEDs and the buzzer off
    Function has no input parameters or returned values""
    RedLED.low()
    YellowLED.low()
    GreenLED.low()
    Buzzer('off')
    print("All Lights and Sound have been turned off")

def Lights_Sound_Action():
    """This function turns on various LEDs and controls the buzzer depending on
the error flags which have been raised. Function has no input parameters or
returned values""
    Lights_Sound_Off()  # Turn lights off
    print("Lights_Sound_Action() has been called. Let the show begin.")
    import utime
switch = 0  # Switch is boolean variable that switches each run of the LEDs and buzzer so the system knows to alternate
Green = 0  # Variable that says the green LED is to be used
Yellow = 0  # Variable that says the yellow LED is to be used
Red = 0  # Variable that says the red LED is to be used
Blink = 0  # Variable that says LEDs need to blink
Stop = 0  # Variable that counts number of Go's pressed to exit function

if ErrGantry.get() == 1:
    # If the Gantry error flag is raised, tell system to turn on green and yellow LEDs, no blinking
    Green = 1
    Yellow = 1
    print("   Gantry Error detected")
elif ErrProbe.get() == 1:
    # If Probe flag raised, yellow and red LED, no blinking
    Yellow = 1
    Red = 1
    print("   Probe Error detected")
elif ErrBeamAct.get() == 1:
    # If beam actuator flag raised, green and red LED, no blinking
    Green = 1
    Red = 1
    print("   Beam Actuator Error detected")
elif ErrSong.get() == 1:
    # If Song error flag raised, all LEDs on, blinking
    Green = 1
    Yellow = 1
    Red = 1
    Blink = 1
    print("   Piano Switch Board Combination Error detected")
elif ErrRailActR.get()==1 or ErrRailActL.get() == 1:
    # If Either rail actuator flags are raised
    Green = 1
    Yellow = 1
    Red = 0
    Blink = 1
    print("   One or Both of the Rail Actuators Error detected")
elif ErrBoltCsv.get() == 1 or ErrCalCsv.get() == 1:
    Green = 1
    Yellow = 1
    Red = 1
    Blink = 0
    print("   Missing file")
else:
    # No error, Set Green LED, turn noise on 1 sec
    print("   No Error detected, doing the green light and 1 second beep")
    GreenLED.high()
    Buzzer('on')
    utime.sleep(1)
    Buzzer('off')
    return

while True:
    Start = utime.ticks_ms()
# Check Switch for what to do
if switch == 0:
    # Check Buzzer for if it should run or not. Should not run if Stop
    # is not 0 meaning user hit go at some point.
    if Stop == 0:
        Buzzer('On')
        print("Sound On")
    else:
        print("Sound Off")
        Buzzer('Off')

    # Check what LEDs should be on
    print('LEDs on')
    if Green == 1:
        GreenLED.high()
    if Yellow == 1:
        YellowLED.high()
    if Red == 1:
        RedLED.high()
else:
    # Buzzer should be off
    print("Sound Off")
    Buzzer('Off')

    # Check what LEDs should be blinking. If so, then the LEDs now
    # need to be off.
    if Blink == 1:
        print("LEDs off")
        if Green == 1:
            GreenLED.low()
        if Yellow == 1:
            YellowLED.low()
        if Red == 1:
            RedLED.low()

    print("Go pressed: "+str(Stop))
    print("Go() reads "+str(Go())+"\r\n")

Current = utime.ticks_ms()
# Wait in the below while loop until the difference in time from the
# begginning of the while loop to present >= 500ms or user hits go
while True:
    if Go() == 1:
        #
        print("Go pressed")
        if Stop == 0:
            Stop = 1
            # wait .25 sec for button to be released
            utime.sleep_ms(250)
        elif Stop == 1:
            Stop = 2
            break
        elif (Current - Start) >= 500:
            break
    Current = utime.ticks_ms()

# If user has hit go twice during the above while loop, exit function.
# i.e. hit it once, than hit it again when the function loops back to
# the above nested loop.
if Stop == 2:
    Lights_Sound_Off()
    YellowLED.high()
    print("Exiting Lights_Sound_Action()")
    return()

# Toggle switch so the system toggles sound and LEDs
if switch == 0:
    switch = 1
elif switch == 1:
    switch = 0

def ErrorHandler():
    """This function checks flags, prints errors, sets lights, and makes noise accordingly."
    """
    # print statements for debugging purposes
    from setup import ErrSleep, ErrAssembly
    print("Beginning Error Handling")
    print("ErrInit = " + str(ErrInit))
    print("ErrSleep = " + str(ErrSleep))
    print("ErrCalibration = " + str(ErrCalibration))
    print("ErrLeveling = " + str(ErrLeveling))
    print("ErrAssembly = " + str(ErrAssembly))
    print("ErrProbe = " + str(ErrProbe))
    print("ErrGantry = " + str(ErrGantry))
    print("ErrBeamAct = " + str(ErrBeamAct))
    print("ErrRailActR = " + str(ErrRailActR))
    print("ErrRailActL = " + str(ErrRailActL))
    print(""")

    # There was no issues with all of the named error flags, check all of the other error flags
    if ErrSong.get() == 0 and \
        ErrBoltCsv.get() == 0 and \
        ErrCalCsv.get() == 0:
        f = open("Error Report.txt","w")

        if ErrProbe.get() == 1:
            # print("Error with probe, writing error")
            f.write("Probe was unable to take valid measurements")
        elif ErrGantry.get() == 1:
            # print("Gantry was unable to move to the destination")
            f.write("Gantry was unable to move to the destination")
        elif ErrBeamAct.get() == 1:
            # print("Error with beam actuator, writing error")
            f.write("Beam Actuator was unable to move to the destination")
        elif ErrRailActR.get() == 1 or ErrRailActL.get() == 1:
            if ErrRailActR.get() == 1 and ErrRailActL.get() == 1:
                x = "Both rail actuators"
            elif ErrRailActR.get() == 1:
                x = "The right rail actuator"
            elif ErrRailActL.get() == 1:
                x = "The left rail actuator"
            print("error with the " + x)
f.write("x+"Has attempted run past the maximum stroke length "+
"allowed")

f.close()

# After finishing up writing to the Error Report text file, now run the
#  function responsible for turning on and off the LEDs and sound.
Lights_Sound_Action()

#  print("Exiting Error Handler")
# Now Home the system
Home("All")

def Home(*arg):
    '''Function Homes parts as indicated by the *arg which is a tuple of
    strings of whatever the user inputs. For reference, google Python optional
    arguments.
    Inputs:
    *agr: can be any to all of the following
    Probe:   Homes the probe
    RailActR: Homes Right rail actuator
    RailActL: Homes Left rail actuator
    Screwdrivers:   Homes the screwdrivers
    Gantry:   Homes the Gantry
    Bact:    Homes the Beam Actuator
    RailAct:   Homes both rail actuators
    All:     Homes everything
    Outputs:
    None'''

    #  print("Beginning to Home system")

    # Set the stall thresholds of the stepper drivers to the maximum
    #  amount (which they should be already)
    Board1._setStallThreshold(16)
    Board2._setStallThreshold(16)

    # First Home Probe if Probe or all is listed in *arg
    if ('Probe' in arg) or ('All' in arg):
        Probe.Home()

    # Now Home Right and Left rail actuator
    #  print("Homming one or both rail actuators")
    # if ('RailActR' in arg) or ('All' in arg) or ('RailAct' in arg):
    #     RailAct.Home(1)

    # if ('RailActL' in arg) or ('All' in arg) or ('RailAct' in arg):
    #     RailAct.Home(2)

    # Home screwdriver DC Motors and Solenoids
    if ('Screwdrivers' in arg) or ('All' in arg):
        # Turn off both DC motors and solenoids
        #  print("Turning off all DC motors and Solenoids for screwdrivers")
        DCMotor(1,1)
        Solenoid(1,1)

    # Home Gantry
    if ('Gantry' in arg) or ('All' in arg):
Gantry.Home()

# Home Beam Actuator
if ('Bact' in arg) or ('All' in arg):
    BeamActuator.Home()

def Calibration_Mode():
    '''This function directs the XBAS machine to calibrate a machine csv file to a specific X-Beam indicated by length in the file name. The file to be edited is indicated by the user input in a piano switch board.
    Function Inputs:
        None
    Function Outputs:
        None
    ...

    # print("Beginning Calibration Mode")
    
    # Variable switch is used to identify if this is the first time running
    # through this program or not.
    switch = 0

    # Variable block is a hardcoded user input that determines the behavior
    # of the program
    # Value of 1 indicates that the user is using a short block that they
    # manually have to move for calibration, thus the machine waits
    # after moving the gantry for the user to hit go.
    # Value of 2 indicates the user is using a long block for calibration
    # that they don't have to move, so the machine will not wait for
    # user input and just go.
    block = 1

    # 1st Layer
    while True:
        # Set Error Flag fir this mode
        ErrCalibration.put(1)

        # 2nd Layer. System will stay in this loop unless an error occurs or
        # the system finishes calibration.
        while True:
            # Pre-Calibration Mode

            # Turn Yellow LED on indicating the system is initializing the
            # calibration mode. Runs once at the beginning of the function
            Lights_Sound_Off()
            YellowLED.high()

            # If first time running through this section of code
            if switch == 0:
                # First Check Files are present
                Output = FileCheck()
                if type(Output)==str:
                    # If file is missing, handle the error than break back into
                    # 1st layer.
                    break
# Home Machine
Output = Home("All")

# Error Check
if Output == "Error Occurred":
    break

if switch == 0:
    # print("Waiting in pre-Calibration Stage")
    # The calibration mode is ready to go, turn green light on and
    # wait for go.
    Lights_Sound_Off()
    GreenLED.high()
    while True:
        if Go() == 1:
            # User hit go
            Lights_Sound_Off()
            YellowLED.high()  # Turn Yellow LED on Indicating the
            # system is working now
            break  # Exit the 3rd layer and resume in
            # 2nd layer
        # print("Go selected, continuing function")
        elif Mode() != 1:
            # User has selected a different mode before hitting go,
            # exit the function
            # print("Another Mode selected, exiting function")
            return()

# User has hit go, import files. Will error if the file is missing
# and user must hit go twice (once to turn sound off, once to
# resume). After error, return to the pre-calibration stage by
# breaking from the 2nd loop back to the 1st loop.
Output = Import.Song()
if Output == "Error Occurred":
    # Error occurred, handle the error, then break out of the 2nd
    # layer to the 1st layer
    break

# Song imported, import calibration. If there is an issue, go to
# error and wait for user input upon which it breaks back into
# the 1st loop.
Output = Import.Calibration()
if Output == "Error Occurred":
    break

# Now we've imported the Calibration file w/o error, save the value
# of Output to a variable for later use. Also take values out of
# the list output of ImportCalibration and save specific values
# to named variables for ease of use. Note, Offset adjusts the
# distance NearSide and FarSide so they are from the end of the
# X-Beam
    # print("Transcribing calibration data over to local variables")
CalibrationData = Output
NearSide = Output(0)
FarSide = Output(1)
# Move Gantry to the near side
print("Moving Gantry to the near side for calibration purposes")
Output = Gantry.Move(NearSide, probe = False)

# Check output for if an error occured. If no error occured, the
# function should have returned a number indicating the
# measurement.
if Output == "Error Occurred":
    # An error did occur, break back to 1st layer
    print("Error occurred moving gantry, do error handling")
    break

# Gantry is in position, check the "block" variable for if the
# machine needs to wait for user input or not
if block == 1:
    # print("Waiting for user input to do measurement")
    Lights_Sound_Off()
    GreenLED.high()
    while True:
        # Sit in a while loop until the user hits Go()
        if Go() == 1:
            # print("Go pressed, taking measurement")
            Lights_Sound_Off()
            YellowLED.high()
            break

# XBAS now needs to take measurement
Output = Probe()

# Check Output for error from probe()
if Output == "Error Occurred":
    # print("Probe measurement error, do error handling")
    # Error Occurred, exit 2nd layer to 1st layer for error handling.
    break

# Store the data of the near side to the calibration data
CalibrationData[2] = Output
print("Substituting probe measurement into CalibrationData, "+
      "value "+str(Output))

# Calue of NearSide has been stored, now for FarSide
print("Moving Gantry to the Farside for Calibration Purpose")
Output = Gantry.Move(FarSide, probe = False)

# Check output for if an error occured. If no error occured, the
# function should have returned a number indicating the
# measurement.
if Output == "Error Occurred":
    # print("Gantry moving error, do error handling")
    # An error did occur, handle error and break back to 1st layer
    ErrorHandler()
    break

# Gantry is in position, check the "block" variable for if the
# machine needs to wait for user input or not
if block == 1:
    # print("Waiting for user input to do measurement")
Lights_Sound_Off()
GreenLED.high()

while True:
    # Sit in a while loop until the user hits Go()
    if Go()==1:
        # print("Go pressed, taking measurement")
        Lights_Sound_Off()
        YellowLED.high()
        break

    # XBAS now needs to take measurement
    Output = Probe()

    # Check Output for error from probe()
    if Output == "Error Occurred":
        # Error Occurred, exit 2nd layer to 1st layer for error handling
        # print("Probe error occurred, handle error")
        break

    # Store the data of the far side to the calibration data
    CalibrationData[3] = Output

    # Calculate the difference between the level of the two points for
    # the calibration constant

    # At this point, there should have been no errors or such, so
    # finish up by storing information into the Calibration file for
    # the X-Beam Length being calibrated and signal the user that the
    # machine is done.
    Input = XBeam
    FileName = "Calibration"+str(Input)+".csv"
    # print("Writing CalibrationData to "+FileName)
    f = open(FileName,'w')
    String = ''
    for item in CalibrationData:
        String = String + str(item) + ','
    String = String.rstrip(', ')
    # print("writing line: "+String)
    f.write(String)

    # Exit Function
    # print("Calibration Done, exiting function")
    return ('Done')

# End of 2nd Layer Loop

# End of 1st Layer Loop
ErrorHandler()

def Leveling_Mode():
    '''This function is part 1 of the assembly mode which is leveling the
    X-Beam relative to the datum surface.
    Function Inputs:
None
Function Outputs:
  None
...

# print("Beginning Leveling half of the Assembly Mode")

# Variable switch indicates if this is the first time the code is
# running through the code. This is for the purpose of skipping
# the pre-stage every run through after the first, ie after
# error handling.
switch = 0

# While Loop 1st Layer
while True:

  # Set error flag for this mode
  ErrLeveling.put(1)

# While Loop 2nd Layer
while True:

  # Pre Assembly Mode
  
  # Turn Yellow LED on indicating the system is initializing the
  # mode. Only does this once for the functions first
  # run
  
  Lights_Sound_Off()
  YellowLED.high()

  # Is this the first time running through this section of code?
  if switch == 0:
    # First Check Files are present
    Output = FileCheck()
    if type(Output)==str:
      # If file is missing, break back into 1st layer where the
      # error handle function is (at the bottom)
      break

  # Home function, home everything
  Output = Home("All")

  # Check output
  if Output == "Error Occured":
    break

  if switch == 0:
    # Turn Green LED on indicating ready for user input
    Lights_Sound_Off()
    GreenLED.high()

    # Wait for user to hit go or to switch system modes to a
    # different mode
    while True:
      #
      # print("Waiting in pre-Assembly Stage")
      if Go() == 1:
        # Go was pressed, exit the 3rd Layer while loop and
# resume the 2nd layer loop via a break command
#
# print("User hit Go, continue leveling and assembly")
break

elif Mode() != 3:
# Else if the 3 position switch is not set to 3 which
#   is the assembly mode, then exit this function via
#   return

# print("User selected different mode, exit function")
return()

# Set switch to 1 so that on future runs, it skips the while loop
#   for waiting for the user to hit go
switch = 1

# Turn Yellow Light on
Lights_Sound_Off()
YellowLED.high()

# User has hit Go, read the piano board via ImportSong()
Output = Import.Song()

# Check Output for if an error occurred
if Output == "Error Occurred":
#   Error occurred, break 2nd layer loop to return to 1st layer
#   and handle error
#   print("Error in ImportSong, handle error")
break

# ImportSong() had no issues, continue by importing Calibration
#   Data
Output = Import.Calibration()

# Check Output of ImportCalibration for errors
if Output == "Error Occurred":
#   Error occurred, break 2nd layer loop to return to 1st layer
#   and handle error
#   print("Error in ImportCalibration, handle error")
break

# Take data from Import Calibration and Store it for future use
#   print("Transcribe Calibration data to local variables")
CalibrationData = Output
NearSide = CalibrationData(0)
FarSide = CalibrationData(1)
CalConst = CalibrationData(4)

# Import BoltPatternXXX.csv file for the X-Beam just to check
#   that it is both present and valid.
Output = Import.BoltPattern()

# Check the output for an error
if type(Output) == str:
#   If error, break out of 2nd layer into 1st layer for
#   error handling
#   print("Error in ImportBoltPattern, handle error")
break
# Move the Gantry to the far side and take the first measurement
print("Moving Gantry to far side and take measurement")
Output = Gantry.Move(FarSide, probe = True)

# Check function output
if Output == 'Error Occurred':
    # Error occurred, break loop
    print("Error in moving Gantry or probe, handle error")
    break

# Store output
FarLevel = Output

# Move the Gantry to the near side and take the first measurement
print("Moving Gantry to near side and take measurement")
Output = Gantry.Move(NearSide, probe = True)

# Check function output
if Output == 'Error Occurred':
    # Error occurred, break loop
    print("Error in moving Gantry or probe, handle error")
    break

# Store output modified by calibration constant
NearLevel = Output - CalConst
print("Calculate nearside w/cal constant")
print(" FarLevel "+str(FarLevel))
print(" NearLevel "+str(NearLevel))

# Level X-Beam by actuating X-Beam Actuator to raise the X-Beam
to the level needed.
# x1 = distance from line constraint to x-beam actuator point constraint
# x2 = distance from where probe takes near side measurement to line constraint
# Fraction indicating how much the point being leveled changes as
# the X-Beam actuator levels the beam
x = x2/x1

# NOTE: If time, fix this, it seems wrong

# Actuate X-Beam leveling actuator and measure
print("Actuate X-Beam actuator")
Output = BeamActuator.Move((NearLevel - FarLevel)/x,
                           probe = True)

# Check output
if Output == 'Error Occurred':
    # Error occurred, break loop
    print("Error w/beam actuator, handle error")
    break

# Store output
NearLevel = Output - CalConst

# Check, retry, and repeat
Error = 0
Tolerance = 100
while True:
    if NearLevel - FarLevel >= -Tolerance or
       NearLevel - FarLevel <= Tolerance:
        # Within of range, continue while loop
        break
    # Actuate X-Beam leveling actuator and measure
    print("X-Beam not leveled, re-attempt leveling with actuator")
    Output = BeamActuator.Move((NearLevel - FarLevel)/x,
                               Probe = True)

    # Check output
    if Output == 'Error Occurred':
        # Error occurred, break loop
        Error = 1
        break

    # if there was an error in the above while loop, its supposed to
    # exit to the 1st loop. THis is the second break that helps
    # achieve that.
    if Error == 1:
        # print("Error w/beam actuator, handle error")
        break

    # At this point, the beam should be leveled with no errors. Home,
    # zero flags and exit function.
    return ('Done')

# End of the 2nd Layer
ErrorHandler()
# End of the First Layer

def TorqueDown(Input):
    from setup import ErrAssembly
    '''Function Torques down the side indicated by the input
    Function inputs:
    Input is a character "L" or "R", indicating that either the
    left or right side needs to be torqued down.
    Function outputs:
    None'''

    # print("Beginning to Toque Down")
    # Rail Actuators constants
    DistancePerStep = 0.0079375 # mm linear travel per step

    # Position above rails in mm away from the motor. Does not have
    # to be very accurate (+/- 1mm)
    PositionAboveRails = 10 # mm is the distance we determined too
                            # small for human fingers
    # Covert PositionAboveRails from distance in mm to steps for the
    # drivers
    PositionAboveGranite = PositionAboveRails / DistancePerStep

    # Calculating position above granite from the motor just as with
    # PositionAboveRails
    PositionAboveGranite = 10
PositionAboveGranite = PositionAboveGranite / DistancePerStep

# 1st Layer while loop
while True:
    # Set torque for the approach to the rails at a very low stall torque
    print("setting stall threshold of rail actuators low")
    Board2._setStallThreshold(4)

    # Turn on actuators and move twoards a position just above the rails
    print("move rail actuator to point above rails")
    if Input == "L":
        # Turn on left actuator
        print(" left actuator on")
        Board2.Goto(1,PositionAboveRails)
    elif Input == "R":
        # Turn on right actuator
        print(" right actuator on")
        Board2.Goto(2,PositionAboveRails)

# 2nd Layer while Loop
error = 0
while True:
    # Check for if the rail actuator has stalled or if the rail
    # actuator has reached destination.
    if Board2.isStalled(1) == True and Input == "L":
        # Left actuator stalled, call error and break out of while
        # loop for handling it
        print(" left actuator stalled")
        ErrRailActL = 1
        error = 1
        break
    elif Board2.isStalled(2) == True and Input == "R":
        # Right actuator stalled, call error and break out of while
        # loop for handling it
        print(" right actuator stalled")
        ErrRailActR = 1
        error = 1
        break
    elif Board2.isBusy(2) == 1:
        # Check for completion (Done flag false), exit with no
        # error called
        print(" actuator in position above rails")
        error = 0
        break

    # End of the 2nd Layer while loop, back in the 1st layer

# Error Check, If an error was declared earlier, then skip
# the following code responsible for fully pressing down the
# rails and bolting down the screws
if error == 0:
    # Set torque high
    print("setting torque high")
    Board2._setStallThreshold(16)
# Set actuators on depending on input
#  print("moving rail actuator to point above granite")
if Input == "L":
    # Turns left actuator on
    print("    left actuator on")
    Board2.Goto(1,PositionAboveGranite)
elif Input == "R":
    # Turns right actuator on
    print("    right actuator on")
    Board2.Goto(2,PositionAboveGranite)

# 2nd layer while loop checking destination and stall flags as before.
while True:
    if Board2.isBusy(2) == 1:
        # Check for completion, call error if actuator moved to destination successfully
        if Input == "L":
            # left actuator reached destination, error.
            print("    left actuator reached destination, error.")
            ErrRailActL.put(1)
        if Input == "R":
            # right actuator reached destination, error.
            print("    right actuator reached destination, error.")
            ErrRailActR.put(1)
        break
    if Board2.isStalled(1) == True or Board2.isStalled(2) == True:
        # Left or right actuator stalled, move on to next stage (torque down)
        # print("Actuator stalled, begin torque down section")
        # Truning On solenoids based on side being done
        if Input == "L":
            # Left Side
            print("    Left Solenoid On")
            Solenoid('l','on')
        if Input == "R":
            # Right Side
            print("    Right Solenoid On")
            Solenoid('r','off')

    # Import Time so system can wait for stuff to finish
    # Wait for solenoids to extend
    utime.sleep(100)

    # Turn On motors depepending on side being done
    if Input == "L":
        # Left Side
        print("    Left DC motor On")
        DCMotor("l","On")
    if Input == "R":
        # Right Side
        print("    Right DC motor On")
        DCMotor("R","On")

    # Sleep for X amount of time so the DC motor torques down the bolt.
utime.sleep(5000)

# Turn screwdriver motors and solenoids off
    #
    # Turn screwdriver motors and solenoids off
    #
    # Turn screwdriver motors and solenoids off
print("all motors and solenoids off")
DCMotor(1,1)
Solenoid(1,1)

# Sleep till solenoids retract
utime.sleep(100)

# home the rail actuators depending on side being done
if Input == "L":
    # Left Side
    #
    #  print("Homing Left Actuator")
    Home("RailActL")
if Input == "R":
    # Right Side
    #
    #  print("Homing Right Actuator")
    Home("RailActR")

# Program done, return
return()

# end 2nd Layer, back in 1st layer
# Home the rail actuators in preparation for error handling.
Home("RailAct")

# Error Handling and set mode error back
ErrorHandler()
ErrAssembly.put(1)

# End of 1st Layer, resume at beginning

def Assembly_Mode():
    from setup import ErrAssembly
    '''
    Function that runs after the beam is leveled to force rails into
    position and torque them down.
    Function Inputs:
    None
    Function Outputs:
    None
    ...
    '''

    # print("Beginning Assembly half of the assembly mode")

    # Set Error Flag for mode
    ErrAssembly.put(1)

    # Beginning of the while loop for importing the data stored in
    # BoltPatternXXX.csv
    while True:
        #
        #  print("ImportBoltPattern function called")
        Output = Import.BoltPattern()
        if type(Output) == str:
            # Error Occurred
            #
            #  print("Error importing Bolt Pattern csv file.")
            ErrorHandler()
            ErrAssembly = 1

    # End of the while loop for importing the data stored in
    # BoltPatternXXX.csv
else:
    # No Error Occured
    break
    # End of while loop

# Store Output into variables for later use. Also assign value to offset # to adjust the Bolt
print("Transcribing bolt pattern data to two separate lists")
BoltSide = Output(0)
BoltData = Output(1)

# For loop to run through the values of BoltSide and BoltData
for counter in range(0,len(BoltData)):
    # Nested while loop to handle moving the gantry to the target point
    while True:
        # Run Gantry to position
        #
        # Output = Gantry.Move(BoltData(counter))
        #
        # Check output of the function for error
        if Output == "Error Occured":
            # Error Occured
            # print("error ocured moving ganty, handling error")
            ErrorHandler()
            ErrAssembly.put(1)
        else:
            # No Error Occured
            # print("Gantry successfully moved, "+str(counter)+"/"+
            # str(len(BoltData))+" moves completed")
            break
    # End of while loop, go back to for loop and repeat entire # process of moving gantry.
    # The gantry should be in position, press rails down and bolt them
    TorqueDown(BoltSide(counter))
    #
    # End of for loop, go back to beginning unless done.
    # Zero Flags, indicate the system is done via Lights_Sound_Action(), and # return
    zero_flags()
    Lights_Sound_Action()
return

def Sleep_Mode(Input):
    '''This is the sleeping mode for the machine. It homes the machine and goes to sleep.
    Function Inputs:
    The amount of time spend checking the Mode()
    Function Outputs:
    Time spent in the sleep mode after if finishes homing with no issues.
    '''
    from setup import ErrSleep
    # Turn all lights and sounds off, then turn on yellow LED
    Lights_Sound_Off()
YellowLED.high()

# Set the sleep mode error flag buffer to 1 so that if an issue occurs, the flag is already set.
ErrSleep.put(1)

# Turn all lights and sounds off, then turn on green LED
Lights_Sound_Off()
GreenLED.high()

# Pre-Sleep mode: Wait in while loop for one of two exit conditions. 
# 1) User hits go
# 2) User changes the system mode to something that isn't the sleep mode
print("Sleep mode pre-stage")
Timer = 0  # Creating a timer variable
Start = utime.ticks_ms()  # Creating a starting reference time
while True:
  if Timer > 60000:  # 1 min * 60s/min * 1000ms/s
    print("Timed Out, going to sleep")
    break
  if Go() == 1:
    print("Go pressed, going to sleep")
    break
  elif Mode() != 2:
    print("Another mode selected, exiting function")
    return(1000)
  # increment Timer
  print("Time in Sleep pre-stage: "+str(Timer)+" ms")
  Current = utime.ticks_ms()
  Timer = Current - Start + Timer
  Start = Current

  # User must have hit go to have reached this point of the program. Turn off lights, turn yellow LED, then enter the new while loop that will try to home the system. Function will continue until the machine has been homed.
  # Amendment, if the system spend 500ms or more in the while loop checking the Mode() function, then the system homes. Otherwise, if it spent less than that, it doesn't run the Home() function.
  if Input >= 500:
    Lights_Sound_Off()
    print("check")
    YellowLED.high()
    while True:
      # print("Time spend in main checking Mode() >500ms")
      # If the system enters sleep mode, Home
      Output = Home("All")

      # Check Output for Error
      if type(Output) == str:
        # If error, do the whole error handeler and when user hits go
        # 2nd time, repeat (ie home again)
        print("Error Homing, handle error")
        ErrorHandler()
        ErrSleep = 1
      elif type(Output) != str:
def Move(Input):
    '''Moves the probe up, down, or not at all
    Function Inputs:
        Input can be 1 of 3 things
        1) "up" or "Up" moves the probe up
        2) "down" or "Down" moves the probe down
        3) anything else stops the probe.
    Function Outputs:
        None'''
    if Input == "up" or Input == "Up":
        RaisePin.high()
        LowerPin.low()
    elif Input == "down" or Input == "Down":
        RaisePin.low()
        LowerPin.high()
    else:
        RaisePin.high()
        LowerPin.high()

def Probe(Limit = False, UpperLimit = 0, LowerLimit = 0):
    '''Lower Probe, Take measurement, Check measurement, Raise the probe if no error. If not, throw an error.
    Function Inputs:
        Limit is True or False, defaulting to False. If the Limit is False, the function does not check teh value taken against the upper and lower limits. If True, it does compare.
    Function Outputs: Returns the value of the measurement taken. Otherwise returns message "Error Occured" if the reading is outside the given limits.'''
    print("Home the Probe")
    Home()
    print("Taking measurement with probe")
    RefRead = ProbeEncoder.read()
    Move("down")
    start = utime.ticks_ms()
    current = utime.ticks_ms()
    tolerance = 1
    while True:
        # wait for readings from probe to change (delta) by the tolerance amount.
        CurrentRead = ProbeEncoder.read()
        delta = CurrentRead - RefRead
        # many debugging statements. ctr + 1 to comment or uncomment sections of code FYI
        # print(" Current reading = "+str(CurrentRead))
        # print(" Previous reading = "+str(RefRead))
        RefRead = CurrentRead
        # print(" delta = "+str(delta))
        # print(" tolerance= "+str(tolerance))
# print(" time passed in ms = "+str(current - start))
# print(" exit cond 1: "+str(delta)+" >= "+str(tolerance)+
# " is "+str(delta <= tolerance))
# print(" exit cond 2: "+str(delta)+" <= "+
# str(-tolerance)+" is "+str(delta >= -tolerance))
# print(" exit cond 3: "+str(current - start)+
# " > 500 is "+str((current - start > 500)))
if delta <= tolerance and delta >= -tolerance and 
(current - start > 500):
    print(" Probe met surface")
    break
# print("
utime.sleep_ms(100)
current = utime.ticks_ms()

# Check Reading is within the limits if the Limit flag is true
Reading = ProbeEncoder.read()
print(" Probe read: "+str(Reading))
if Limit == True:
    if Reading > UpperLimit or Reading < LowerLimit:
        print(" probe reading was above upper limit of readings")
        ErrProbe.put(1)
    print(" Homing probe")
    Home()

print(" ErrProbe = "+str(ErrProbe.get())))
# If there was or was not an error
if ErrProbe.get() == 0:
    print(" No error, exit")
    # No error, return
    return(Reading)
else:
    # Error, return error message
    print(" Error Error Error Error")
    return("Error Occurred")

def Home():
    '''Function homes the probe by checking the reference tick for the
    probe. The function also has a timer if the reference tick doesn't
    work
    Function Inputs:
    None
    Function Outputs:
    None'''
    if ProbeReference.value() != 1:
        # False, Retract Probe until it is at the reference tick
        print(" Probe is not at reference tick, raise probe")
        Move("up")

        # While Loop to check the probe for when its reaches reference
        # tick
        start = utime.ticks_ms()
        while True:
            current = utime.ticks_ms()
            if ProbeReference.value == 1 or (current - start) > 2000:
Move("Stop")
print("Probe is at reference tick, Zero out encoder")
ute.sleep_ms(100)
ProbeEncoder.zero()
ProbeEncoder.position = 0
ute.sleep_ms(500)
print("Probe homed at " + str(ProbeEncoder.position))
break

def read():
    '''Function returns the read value from the Probe’s Encoder
    Function Inputs:
    None
    Function Outputs:
    None'''
    return(ProbeEncoder.read())

# Encoder Pins and Object for the Probe
# C6 is encoder ch1
# C7 is encoder ch2
# ProbeEncoder is a Quadrature Encoder Object
import encoder as enc
import pyb
import ute

tim8 = pyb.Timer (8,freq=1000)
ProbeEncoder = enc.Quad_Encoder(pinC6,pinC7,tim8)

'''Probe Pins'''
# H1 is the reference tick
ProbeReference = pyb.Pin(pyb.Pin.cpu.H1, mode = pyb.Pin.IN,
    pull = pyb.Pin.PULL_DOWN)

# B9 high sends the probe up if B8 is Low
    pull = pyb.Pin.PULL_DOWN)
RaisePin.high()

# B8 high sends the probe down if B9 is Low
    pull = pyb.Pin.PULL_DOWN)
LowerPin.high()

# Grab the ErrProbe error flag from setup
from setup import ErrProbe
def checkSide(Side):
    '''Function determines the rail actuator being called
    Function Inputs:
        Side indicates which rail actuator is being operated. It takes values of Right, right, r, R, and 1 for the right rail actuator. It takes values of Left, left, l, L, and 2 for the left rail actuator.
        Both, both, b, B, or 3 indicates both rail actuators and is only for the Home command
    Function Outputs:
        Function outputs either a 1 or 2 corresponding to the right and left actuator respectively
    ...
    if Side == 'right' or \
        Side == 'Right' or \
        Side == 'r' or \
        Side == 'R' or \
        Side == 1:
        return(1)
    elif Side == 'left' or \
        Side == 'Left' or \
        Side == 'l' or \
        Side == 'L' or \
        Side == 2:
        return(2)

def Move(Side, Destination, stall=90):
    '''Moves the selected rail actuator to the given destination
    Function Inputs:
        Side indicates which rail actuator is being operated. The value is put into the function checkSide to determine the actuator being called.
        Destination is the destination desired in steps.
        stall is the stall threshold of both the rail actuators. defaults to 65 which we found to be weak enough not to hurt a person, but enough to move without stall. Set to higher stalls for specific commands
    Function Outputs:
        If there was stall, returns "Stalled"
        If move was completed, returns "Completed move command"
    Note: This function was written very last minute and is untested ...
    
    # setting stall threshold
    Board2._setStallThreshold(stall)
    
    # determine side
    motor = checkSide(Side)
    
    # get status on motor being run (clears errors)
    Board2.GetStatus(motor)
    
    # run motor. This gets the motor going for the GoUntil
    # command Later since we noticed the GoUntil command
# sometimes doesn't work until after a Run command.
# this was a last minute addition to help cover the
# aforementioned bug and is not in the flow chart
Board2.Run(motor,200)

# start timer
start = utime.ticks_ms()

# while loop that waits for stall or a time out
while True:
    current = utime.ticks_ms()
    if Board2.isStalled(motor):
        print('stalled')
        Board2.HardHiZ(motor)
    if current-start > 1000:
        utime.sleep(5)
        Board2.HardHiZ(motor)
        Home(motor)
        break

# move motor to destination
Board2.GoTo(motor,Destination)

# check rail actuators for completion or stall
while True:
    if Board2.isBusy(motor) == False and switch == 1:
        # gantry at switch, continue.
        print(" Board isn't busy anymore")
        Board2.GetStatus(motor)
        Board2.HardHiZ(motor)
        Board2._setStallThreshold(127)
        utime.sleep(1)
        return("Completed move command")
    elif Board2.isStalled(motor) == True:
        print(" Board stalled: True")
        # motor stalled, return an error and set flag
        Board2.HardHiZ(motor)
        Board2._setStallThreshold(127)
        utime.sleep(1)
        print(" rail actuator Stalled during move command")
        return("Stall occured")

def Home(Side):
    '''Function homes one or both of the rail actuators depending on the input
    Function Inputs:
    Side indicates which actuator (if not both) is being homed
    Function Outputs:
    returns "Completed move command" when done
    Note: this function was written last minute and is untested
    ...
    print('homing rail actuator')
    motor = checkSide(Side)
    if motor == 1 or motor == 2:
        Board2.GoUntil(motor,-100)
    else:
Board2.GoUntil(1, -100)
Board2.GoUntil(2, -100)

if Board2.isHome(motor) == False:
    print(' not home')
while True:
    if Board2.isBusy(motor) == False:
        # gantry at switch, continue.
        print(" Board isn't busy anymore")
        Board2.HardHiZ(motor)
        Board2.GetStatus(motor, verbose = 0)
    else:
        print(' already Bhome')

while True:
    if Board2.isBusy(motor) == False or Board2.isHome(motor) == False:
        Board2.HardHiZ(motor)
        Board2.GetStatus(motor, verbose = 1)
        utime.sleep(5)
        return("Completed move command")
    else:
        print(' already Bhome')

from setup import Board2
import utime
def zero_flags():
    '''Function sets all error flag buffers to false aka 0 or its equivalent.
    No input parameters or returned values. Does require for the task_share.py
    file to be present and the main section of this program to have been run
    so that the buffers exist.''

    ErrInit.put(0)
    ErrSleep.put(0)
    ErrLeveling.put(0)
    ErrAssembly.put(0)
    ErrBoltCsv.put(0)
    ErrCalCsv.put(0)
    ErrGantry.put(0)
    ErrProbe.put(0)
    ErrRailActL.put(0)
    ErrRailActR.put(0)
    ErrBeamAct.put(0)
    ErrFileCheck.put(0)
    ErrSong.put(0)
    XBeam.put(0)
    print("Flags Zeroed")

def FileCheck():
    '''Function checks for files in the same directory and writes to the error
    report, listing the files from FileList that are missing. After that, if
    the Switch has been set to 1 (meaning an file was missing), it sets the
    ErrFileCheck to 1 and runs the ErrorHandler() function. This function
    also completely takes care of its own error report. Takes no parameters.
    @return Function returns a string if an error occurred, an integer if not.
    '''

    print("Checking Files")
    Switch = 0  # Indicates if an error has occurred. Used so that the error
    # message indicating that the error occurred during the
    # initialization phase is written only once at the
    # beginning of the report.
    import os  # Import os for the listdir() function

    # List of strings of file names to check for in the system directory
    FileList = ["main.py",
                "boot.py",
                "16470nucleo.py",
                "encoder.py",
                "task_share.py",
                "BeamActuator.py",
                "Gantry.py",
                "Probe.py",
                "Import.py",
                "setup.py"]

    # Retrieve the system directory information as a list of strings
    files = os.listdir()
# print("files present")
# print(files)
# print("")
# Open the Error Report text file in preparation of writing errors to
f = open("Error Report.txt", 'w')
for file in FileList:
    print(" file being checked: "+str(file))
    if file not in files:
        # If the file being checked is not on the pyboard
        if Switch == 0:
            f.write('There was an error during the system initializiation'+'\'"r\r\n")
            Switch = 1
        f.write("The file "+file+" is missing\r\n")
        print(" The file "+file+" is missing")
        # If special action is required for a file, add an if statement
        # here which checks for said file to write that specific
        # recommended action here.
        f.write("  Recommended action: Copy "+file+" from a backup onto"+'" the pyboard\r\n")
        # Written Error report should look like...
        # There was an error during the initializiation phase
        # The file main.py is missing
        # Recommended action: Copy main.py from a backup onto the pyboard

# Finished writing to Error Report, close the text file
f.close()

# Copied from main.Lights_Sound_Action(). This section of error handling
# is if task_share.py is missing in which case main.Lights_Sound_Action()
# and main.ErrorHandler() cannot work due to undefined error flag buffers.
# Thus FileCheck has to take care of itself
if Switch != 0:
    # If there was an error, set flag
    print("File Check Done, error occured, please press (deliberatly) "+"\n    "the go once to turn the noise off")
    # Perform special error handling unique to this function which must be
    # done here for the system to work
    RedLED.high()
    YellowLED.high()
    GreenLED.high()
    switch = 0
    go = 0
    Time = 0
    while True:
        start = utime.ticks_ms()
        if switch == 0 and go == 0:
            Buzzer('on')
        elif switch == 1 and go == 0:
            Buzzer('off')
        else:
            Buzzer('off')
        if Go() == 1 and go == 0:
            go = 1
Buzzer('off')
print("Go hit first time, sleep for 0.25 seconds.")
ute.sleep_ms(250)
print("Buzzer Off, please press the go button once more\n" to resume")
elif Go() == 1 and go == 1:
    break
if go == 0:
current = ute.ticks_ms()
Time = Time + (current-start)
start = current
if Time >= 1000 and go == 0:
    if switch == 0:
        switch = 1
    else:
        switch = 0
    Time = 0
    return("Error Occured")
else:
    print("File Check Done, no errors")
    return([files,FileList])

def paramatarize():
    
    Function parameterizes the l6470nucleo.Dual6470 objects
    Function Inputs:
    None
    Function Outputs:
    None
    ...

    print(' parameterizing')
    # Set the registers which need to be modified for the motor to go
    # This value affects how hard the motor is being pushed
    print(' parameterizing board 1')
    K_VAL = 65
    Board1._set_par_1b ('KVAL_HOLD', K_VAL)
    Board1._set_par_1b ('KVAL_RUN', K_VAL)
    Board1._set_par_1b ('KVAL_ACC', K_VAL)
    Board1._set_par_1b ('KVAL_DEC', K_VAL)
    # Speed at which we transition from slow to fast V_B compensation
    INT_SPEED = 1032 #3141
    Board1._set_par_2b ('INT_SPEED', INT_SPEED)
    # Acceleration and deceleration back EMF compensation slopes
    ST_SLP = 25
    Board1._set_par_1b ('ST_SLP', ST_SLP)
    Board1._set_par_1b ('FN_SLP_ACC', ST_SLP)
    Board1._set_par_1b ('FN_SLP_DEC', ST_SLP)
    # Set the maximum speed at which motor will run
    MAX_SPEED = 30
    Board1._set_par_2b ('MAX_SPEED', MAX_SPEED)
    # Set the minimum speed at which motor will run
    MIN_SPEED = 15
    Board1._set_par_2b ('MIN_SPEED', MIN_SPEED)
    # Set the maximum acceleration and deceleration of motor
    ACCEL = 1
DECEL = 20
Board1._set_par_2b ('ACC', ACCEL)
Board1._set_par_2b ('DEC', DECEL)

# Set the number of Microsteps to use
SYNC_EN = 0x00
SYNC_SEL = 0x10
STEP_SEL = 8
Board1._set_MicroSteps (SYNC_EN, SYNC_SEL, STEP_SEL)

# Set the Stall Threshold
STALL_TH = 127
Board1._setStallThreshold(STALL_TH)

print(' parameterizing board 2')
# Set the registers which need to be modified for the motor to go
# This value affects how hard the motor is being pushed
K_VAL = 60
Board2._set_par_1b ('KVAL_HOLD', K_VAL)
Board2._set_par_1b ('KVAL_RUN', K_VAL)
Board2._set_par_1b ('KVAL_ACC', K_VAL)
Board2._set_par_1b ('KVAL_DEC', K_VAL)

# Speed at which we transition from slow to fast V_B compensation
INT_SPEED = 1032 #3141
Board2._set_par_2b ('INT_SPEED', INT_SPEED)

# Acceleration and deceleration back EMF compensation slopes
ST_SLP = 25
Board2._set_par_1b ('ST_SLP', ST_SLP)
Board2._set_par_1b ('FN_SLP_ACC', ST_SLP)
Board2._set_par_1b ('FN_SLP_DEC', ST_SLP)

# Set the maximum speed at which motor will run
MAX_SPEED = 20
Board2._set_par_2b ('MAX_SPEED', MAX_SPEED)

# Set the maximum acceleration and deceleration of motor
ACCEL = 12
DECEL = 12
Board2._set_par_2b ('ACC', ACCEL)
Board2._set_par_2b ('DEC', DECEL)

# Set the number of Microsteps to use
SYNC_EN = 0x00
SYNC_SEL = 0x10
STEP_SEL = 8
Board2._set_MicroSteps (SYNC_EN, SYNC_SEL, STEP_SEL)

# Set the Stall Threshold
STALL_TH = 127
Board2._setStallThreshold(STALL_TH)

def DCMotor(Side, Dir):
    '''Function for running the DC motors
    Function Inputs:
    Side can be any of the following and indicates which DC motor
    is being called upon:
    1) "Left", "left", "l", and "L" calls on the left motor
    2) "Right", "right", "r", and "R" calls on the right motor
    3) Anything else turns off both motors regardless of the
value of Dir

Dir Indicates if the DC motors selected by Side is turned on or off. Takes the following inputs:
1) "On" or "on" will turn the motor selected on
2) "Off" or "off" will turn the selected motor off

Function Outputs:
None

```
if Side == "Left" or Side == "left" or Side == "l" or Side == "L":
    if Dir == "on" or Dir == "On"
        DCMotorLeftPin.low()
    elif Dir == "off" or Dir == "Off"
        DCMotorLeftPin.high()
elif Side == "Right" or Side == "right" or Side == "r" or Side == "R":
    if Dir == "on" or Dir == "On"
        DCMotorRightPin.low()
    elif Dir == "off" or Dir == "Off"
        DCMotorRightPin.high()
else:
    DCMotorLeftPin.high()
    DCMotorRightPin.high()
```

def Solenoid(Side, Dir):
    '''Function for running the solenoids

    Function Inputs:
    Side can be any of the following and indicates which solenoid is being called upon.
    1) "Left", "left", "l", and "L" calls on the left solenoid
    2) "Right", "right", "r", and "R" calls on the right solenoid
    3) Anything else turns off both solenoids regardless of the value of Dir

    Dir Indicates if the Solenoid selected by Side is turned on or off. Takes the following inputs:
    1) "On" or "on" will turn the solenoid selected on
    2) "Off" or "off" will turn the selected solenoid off

    Function Outputs:
    None

    ```
    if Side == "Left" or Side == "left" or Side == "l" or Side == "L":
        if Dir == "on" or Dir == "On"
            SolenoidLeftPin.low()
        elif Dir == "off" or Dir == "Off"
            SolenoidLeftPin.high()
    elif Side == "Right" or Side == "right" or Side == "r" or Side == "R":
        if Dir == "on" or Dir == "On"
            SolenoidRightPin.low()
        elif Dir == "off" or Dir == "Off"
            SolenoidRightPin.high()
    else:
        SolenoidLeftPin.high()
        SolenoidRightPin.high()
    ```

def Buzzer(Input, duty = 50):
    '''Function runs the buzzer, turning it on or off depending on the inputs.

    Function Inputs:
    Input can be one of the following.
    1) "On", or "on" indicate that the buzzer should turn on
2) Anything else turns the buzzer off
duty defaults to 100 and represents the duty cycle of the pwm wave controlling the buzzer. The function restricts the values of duty to be between 0 and 100 percent

```python
if Input == "On" or Input == "on":
    if duty > 100:
        duty = 100
    if duty < 0:
        duty = 0
    BuzzerChannel.pulse_width_percent(duty)
else:
    BuzzerChannel.pulse_width_percent(0)
```

def callback(line):
    '''This is a function which runs during interrupts. This should occur when the emergency stop button is pressed down. It waits until the emergency stop has been disengaged and initiates a soft restart'''
    print("Emergency Stop pressed...")
    RedLED.high()
    while True:
        if Stop_Pin.value() == 0:
            print("... and released")
            RedLED.low()
            break

import pyb
pyb.hard_reset()

def Mode():
    '''Function which is used to read and return the selection of the three position switch.

@input  Inputs are the read values of pin ThreeSwitch()
@return Returns either a 1, 2, or 3 corresponding with the selection of the three position switch which are differentiated by different resistor values hardcoded below.'''

    # Get the reading from the three position switch analog pin
    value = ThreeSwitch()

    # Pins by Voltage in voltage
    V3 = 4030
    V2 = 1950
    V1 = 1300
    tolerance = 200
    # print("First Position value in ticks: "+str(V1))
    # print("Second Position value in ticks: "+str(V2))
    # print("Third Position value in ticks: "+str(V3))
    # print("Tolerance of "+str(tolerance))
    # print("Three Pos Switch read value in ticks: "+str(value))

    # Check if the first position has been selected
    if (value <= (V1 + tolerance)) and (value >= (V1 - tolerance)):
        return("Selected Mode 1")
    return(1)

    # Check if the second position has been selected
elif (value <= (V2 + tolerance)) and (value >= (V2 - tolerance)):
    # print("Selected Mode 2")
    return(2)

# Check if the third position has been selected
elif value <= V3 + tolerance and value >= V3 - tolerance:
    # print("Selected Mode 3")
    return(3)
else:
    # print("No Mode Selected")
    return(0)

import pyb
import utime
start = utime.ticks_ms()

# Pin Definition
print("Creating pins")

# Notable stepper driver pins

# LED Pin Definition
print("    creating LED pins")
    pull = pyb.Pin.PULL_DOWN)
    pull = pyb.Pin.PULL_DOWN)
    pull = pyb.Pin.PULL_DOWN)

YellowLED.high()
RedLED.low()
GreenLED.low()

# Piez Buzzer
timBuzzer = pyb.Timer(5,freq = 2730)
BuzzerChannel = timBuzzer.channel(2, pyb.Timer.PWM, pin = BuzzerPin)
BuzzerChannel.pulse_width_percent(0)

# Solenoid and DC Motor Pin call outs and functions for controlling said pins.
print("    Creating solenoid pins")
SolenoidLeftPin = pyb.Pin (pyb.Pin.cpu.D2, mode = pyb.Pin.OUT_PP,
    pull = pyb.Pin.PULL_DOWN)
SolenoidLeftPin.high()

    pull = pyb.Pin.PULL_DOWN)
SolenoidRightPin.high()

print("    Creating DC motor pins")
    pull = pyb.Pin.PULL_DOWN)
DCMotorLeftPin.high()
    pull = pyb.Pin.PULL_DOWN)

DCMotorRightPin.high()

# Creating the Go button function call. Go() should give 0 or 1 depending on
# the pin input. Basically, equating Go_Pin.value() to Go() so I do less
# typing.
print("    Creating go pin")
Go = Go_Pin.value

'''Three Position Switch Pin'''
print("    Creating 3 pos switch pin")
adc = pyb.ADC(ThreeSwitch_Pin)
ThreeSwitch = adc.read

# Check Files
Output = FileCheck()

# Check Output of function FileCheck() for an error in the shape of a string
if type(Output)==str:
    # If there was an error, handle it
    print("error with files, error handled by special exception error error
          handling section of FileCheck()")

    # If there was an error and the user hit the go twice, do a soft reset,
    # restarting the program from the beginning.
    print("SOFT RESET!!!!")
    import sys
    sys.exit()

# If there was no error, create all items
print("Creating class objects")

'''Stepper Driver pin and object creations'''
import 16470nucleo  # Import file
SCK= pyb.Pin(pyb.Pin.cpu.B5)  # stby_rst_pin
ncs1= pyb.Pin(pyb.Pin.cpu.A10)  # cs_pin for board 1
ncs2= pyb.Pin(pyb.Pin.cpu.A4)  # cs_pin for board 2
Board1 = 16470nucleo.Dual16470(1,ncs1,SCK)  # Controls the Gantry (2) and
    # Beam Actuator (1)
Board2 = 16470nucleo.Dual16470(1,ncs2,SCK)  # Controls Left (1) and Right
    # (2) rail actuators
    # Beam Actuator (1)

paramatarize()
Board1.HardHiZ(1)
Board1.GetStatus(1,verbose = 0)
Board1.HardHiZ(2)
Board1.GetStatus(2,verbose = 0)
Board2.HardHiZ(1)
Board2.GetStatus(1,verbose = 0)
Board2.HardHiZ(2)
Board2.GetStatus(2,verbose = 0)

print("    creating interrupt for emergency stop")
The same as the Go pin, but greating a short function call for the emergency stop button. This pin is defined last as to prevent memory issues found importing task_share.


'''External Interrupt Pin'''

import micropython

micropython.alloc_emergency_exception_buf(100)

extint = pyb.ExtInt(Stop_Pin, pyb.ExtInt.IRQ_FALLING, pyb.Pin.PULL_DOWN, callback)

import task_share

# Variable Buffer Creation

ErrInit = task_share.Share ('i', thread_protect = False,
        name = "Initialization Error Flag")

ErrSleep = task_share.Share ('i', thread_protect = False,
        name = "Sleep Error Flag")

ErrCalibration = task_share.Share ('i', thread_protect = False,
        name = "Sleep Error Flag")

ErrLeveling = task_share.Share ('i', thread_protect = False,
        name = "Sleep Error Flag")

ErrAssembly = task_share.Share ('i', thread_protect = False,
        name = "Sleep Error Flag")

ErrBoltCsv = task_share.Share ('i', thread_protect = False,
        name = "BoltPattern.csv Error Flag")

ErrCalCsv = task_share.Share ('i', thread_protect = False,
        name = "Calibration.csv Error Flag")

ErrGantry = task_share.Share ('i', thread_protect = False,
        name = "Gantry Error Flag")

ErrProbe = task_share.Share ('i', thread_protect = False,
        name = "Probe Error Flag")

ErrRailActL = task_share.Share ('i', thread_protect = False,
        name = "Left Rail Actuator Error Flag")

ErrRailActR = task_share.Share ('i', thread_protect = False,
        name = "Right Rail Actuator Error Flag")

ErrBeamAct = task_share.Share ('i', thread_protect = False,
        name = "X-Beam Actuator Error Flag")

ErrSong = task_share.Share ('i', thread_protect = False,
        name = "Incorrect Piano Board Input Flag")

ErrFileCheck = task_share.Share ('i', thread_protect = False,
        name = "File Check Error Flag")

# This buffer contains the value of the X-Beam Length indicated by the combination of switches on the piano switch board.

XBeam = task_share.Share ('i', thread_protect = False,
        name = "X-Beam Length")

current = utime.ticks_ms()

print('importing setup.py took '+str((current - start)/1000)+' seconds')
Appendix P10: task_share.py

# -*- coding: utf-8 -*-
#
## @file task_share.py
## This file contains classes which allow tasks to share data without the risk
## of data corruption by interrupts.
#
## @copyright This program is copyright (c) JR Ridgely and released under the
## GNU Public License, version 3.0.

import array
import gc
import pyb
import micropython

## This is a system-wide list of all the queues and shared variables. It is
## used to create diagnostic printouts.
share_list = []

class Share:
    """ This class implements a shared data item which can be protected
    against data corruption by pre-emptive multithreading. Multithreading
    which can corrupt shared data includes the use of ordinary interrupts as
    well as the use of a Real-Time Operating System (RTOS). """

    ## A counter used to give serial numbers to shares for diagnostic use.
    ser_num = 0

    def __init__(self, type_code, thread_protect = True, name = None):
        """ Allocate memory in which the shared data will be buffered. The
        data type code is given as for the Python 'array' type, which
        can be any of
        * b (signed char), B (unsigned char)
        * h (signed short), H (unsigned short)
        * i (signed int), I (unsigned int)
        * l (signed long), L (unsigned long)
        * q (signed long long), Q (unsigned long long)
        * f (float), or d (double-precision float)
        @param type_code The type of data items which the share can hold
        @param thread_protect True if mutual exclusion protection is used
        @param name A short name for the share, default @c ShareN where @c N
        is a serial number for the share """

        self._buffer = array.array (type_code, [0])
        self._thread_protect = thread_protect

        self._name = str (name) if name != None \
                      else 'Share' + str (Share.ser_num)

        # Add this share to the global share and queue list
        share_list.append (self)

@micropython.native
def put (self, data, in_ISR = False):
    """ Write an item of data into the share. Any old data is overwritten.
    This code disables interrupts during the writing so as to prevent
data corrupting by an interrupt service routine which might access
the same data.
@param data The data to be put into this share
@param in_ISR Set this to True if calling from within an ISR """

# Disable interrupts before writing the data
if self._thread_protect and not in_ISR:
    irq_state = pyb.disable_irq()

self._buffer[0] = data

# Re-enable interrupts
if self._thread_protect and not in_ISR:
    pyb.enable_irq(irq_state)

@micropython.native
def get(self, in_ISR=False):
    """ Read an item of data from the share. Interrupts are disabled as
the data is read so as to prevent data corruption by changes in
the data as it is being read.
@param in_ISR Set this to True if calling from within an ISR """

# Disable interrupts before reading the data
if self._thread_protect and not in_ISR:
    irq_state = pyb.disable_irq()

    to_return = self._buffer[0]

# Re-enable interrupts
if self._thread_protect and not in_ISR:
    pyb.enable_irq(irq_state)

return (to_return)

def __repr__(self):
    """ This method puts diagnostic information about the share into a
string. """

    return ('{:12s} Share'.format(self._name))
Appendix Q: User Manual

X-Beam Alignment System

User Manual

MICRO-VU
Precision Measurement Equipment Since 1959

Cal Poly SLO Senior Project 2017
Written By: Joseph Falcao, Whittaker Hamill, and Robert Tam
Project Advisor: Ian Davison
Table of Contents

1. User Interface Panel Buttons  
   1.1. Three Position Rotary Switch  
   1.2. Green Button or Go  
   1.3. Red Button or Emergency Stop  
   1.4. Piano Switchboard  

2. Nucleo Board Program Files  
   2.1. CalibrationXXX.csv  
   2.2. BoltPatternXXX.csv  
   2.3. ErrorReport.txt  
   2.4. main.py  
   2.5. setup.py  
   2.6. BeamActuator.py  
   2.7. Gantry.py  
   2.8. Import.py  
   2.9. encoder.py  
   2.10. Probe.py  
   2.11. RailAct.py  
   2.11. l6470nucleo.py  
   2.12. task_share.py  
   2.13. boot.py  
   2.14. pybcdc.inf  

3. XBAS Operation  
   3.1. Calibration  
   3.2. Preparing the X-Beam for Assembly  
   3.3. Loading  
   3.5. Assembly  

4. Error Handling  
   4.1. Handling an Error  
   4.2. LED Patterns, Errors, and Solutions  
      4.2.1. Missing Files  
      4.2.2. Gantry
4.2.3. Probe 250
4.2.4. Beam Actuator 250
4.2.5. Invalid Switchboard Input 251
4.2.6. Rail Actuator 251

5. Adding X-Beam 251
   5.1. Creating the BoltPatternXXX.csv file 251
   5.2. Creating the CalibrationXXX.csv file 252
   5.3. Adding the ID code to Import.py 253
   5.4. Optional csv file Debugging 253
   5.5. Optional Documenting 253

6. Machine Service 255
1. User Interface Panel Buttons
This section addresses the User Interface Panel’s buttons of the XBAS.

1.1. Three Position Rotary Switch
The three position switch serves the purpose of letting the user select one of three modes for the XBAS to be in. The three states are:

1. Calibration Mode: The XBAS enters a mode designed to calibrate the variance of height of the rails the gantry runs along out of the measurements used to level the X-Beam in the Assembly mode. In order to run the Calibration mode, see Calibration under XBAS Operation.

2. Sleep Mode: In this mode, the system resets by zeroing all the motors and actuators before going to sleep; waiting for the user to turn it to either the Calibration or Assembly mode. When the user does select either Calibration or Assembly, the system zeroes itself again.

3. Assembly Mode: The XBAS will proceed to assemble the X-Beam which consists of three steps:
   a. Leveling the X-Beam
   b. Straightening the rails
   c. Torqueing down bolts
Figure 1. User Interface Panel Buttons from which the user will control the operations of the XBAS. The XBAS should be positioned so that the user can easily access these buttons.

1.2. Green Button or Go
The green button starts the process selected by the 3-position switch. For example, if the user has selected the Assembly mode, pressing Go will start the Assembly process. Be aware that if the green LED is not lit, then the XBAS is not ready to perform the selected mode and will ignore inputs from pushing the green button until it is done. This is indicated by the green LED turning on.

1.3. Red Button or Emergency Stop
This is the Emergency Stop Button. When pressed, it stops all activity of the XBAS. When released, the machine will restart, losing all progress.

The Emergency Stop follows the standard convention for usage. Press down the button to engage. Turn counterclockwise to disengage.
1.4. Piano Switchboard
The switchboard allows the selection of various options for the XBAS. For example, one combinations of switches could indicate the user is calibrating the machine for a 500mm X-Beam whereas another combination means the machine is being calibrated to a 600mm X-Beam.

![Switchboard Image]

Figure 2. Above to the right is an image of the piano switchboard located in the bottom right area of the electronics. On the left is a drawing of the switch board showing which switch is which and what position correlates to on or off.

The 5 switches can be toggled to various positions, different combinations correlating to a number from 1 to 32. This is done by converting the switches into a binary number where each switch represents a bit and adding 1. For example, if switches 2 and 4 are on and the rest are off, that correlates to the binary number 10 (I.E. 01010 = 10). Add one to that, and the switchboard reads as option 11. This information is used by the Import.Song(), Import.Calibration(), and Import.BoltPattern() functions to identify the documents necessary for the XBAS to work with the X-Beam selected. For further details of how to edit Import.Song() in order to edit the combinations of the switchboard, refer to section 5.3. For further details of how the program works, please refer to the program.

2. Nucleo Board Program Files
This section lists out the files on the pyboard and what they are for.

2.1. CalibrationXXX.csv
Contains distances locating points where the XBAS will calibrate as well as the resulting calibration data from the system. There should be a CalibrationXXX.csv file for every X-Beam the system has stored where XXX is a unique ID code for said X-Beam. This file is used in the Calibration and leveling part of the Assembly mode. The system cannot run either modes if the CalibrationXXX.csv
file for the X-Beam selected by the piano switchboard is not present. See Section 1.4. Piano Switchboard for how to select the X-Beam or see 5.2. for directions to creating the CalibrationXXX.csv file.

2.2. BoltPatternXXX.csv
Contains bolting pattern of how far to travel to the next bolt and which side to bolt indicated by L, R, or B for left, right, and both respectively. This file is used in the Assembly mode to press down the rails using the actuators and bolting down the screws. There is a BoltPatternXXX.csv file for every X-Beam the system has stored where XXX is a unique ID code for said X-Beam. The leveling and assembly functions of the Assembly mode will not be able to finish if the file for the X-Beam being assembled is not present. Details about creating the BoltPatternXXX.csv file can be found in Section 5.1.

2.3. ErrorReport.txt
This text file contains a report detailing the last error to occur. It should also include possible solutions to the error that occurred. Note, this file will update; however if the user is connected to the pyboard via micro-usb, the file may appear to not update. This is fixed by ejecting the pyboard, disconnecting, and reconnecting the pyboard. If this file is not present, it will be created by the program if an error does occur.

2.4. main.py
This file contains the main program as well as a number of functions used for error handling, homing, and the functions for the Calibration, Sleep, Leveling, and Assembly modes.

2.5. setup.py
This file contains the pin and object definitions that the system uses as well as a number of functions such as FileCheck() which are imported and used in a lot of the other program files.

2.6. BeamActuator.py
This file contains the functions used to run and operate the Beam Actuator. It contains three functions as well as code that import objects needed for said functions from the setup.py.

2.7. Gantry.py
This file contains the functions used to run and operate the Gantry. It contains three functions as well as code that import objects needed for said functions from the setup.py.
2.8. Import.py
This file contains the functions used to import the BoltPattern and Calibration csv files and to read the piano switch board to determine the X-Beam being edited.

2.9. encoder.py
This file contains a class called QuadEncoder which is used to manage quadrature encoders. This is used to read measurements from the Heidenhain probe.

2.10. Probe.py
This file contains the functions used to run and operate the Heidenhain probe. It contains four functions as well as code that import objects needed for said functions from the setup.py and uses the QuadEncoder class from encoder.py to keep track of position.

2.11. RailAct.py
This file contains the functions used to run and operate the rail actuators. It contains four functions as well as code that import objects needed for said functions from the setup.py.

2.11. l6470nucleo.py
This file contains the functions used to run and operate the stepper drivers for all of the stepper motors. It contains a class with many class members used in the stepper motor operations.

2.12. task_share.py
This file contains a class called Share which is used to create buffer like variables which can be used to share information between functions while protecting said information from corruption from interrupts.

2.13. boot.py
This script file is a python file used by pyboards for initialization. This file is critical for the system's operation.

2.14. pybcdc.inf
This system initialization file is used when connecting to the terminal for controlling the pyboard. It is not necessary to the function of the board; however, the user may not be able to access the pyboard.
3. XBAS Operation

Below are the steps and detailed descriptions for operating the XBAS. Please refer to the User Interface Panel for details of the individual buttons, switches, and indicators.

3.1. Calibration

In order to calibrate the XBAS, the user will require 1 gage block. After obtaining said block, the first step for the user is to set the XBAS to the calibration mode by setting the three position rotary switch to Calibrate.

After the user has set the machine to calibrate, it will do some initialization. This is indicated by the Yellow LED. Wait for the Yellow LED to turn off and the green LED to turn on. Once the green LED is on, press the Go button. It may take a moment, but the gantry will run to the first calibration point. Again, wait for the green LED. Once you have the green LED, hit go. The probe will lower and take a measurement. After completing this measurement, the gantry will move to the next position. Again, wait for the green LED before hitting go. After the user hits go, the probe will take the second and final reading and the machine will home the system. The light will turn green indicating it is ready to calibrate again if you hit go. Otherwise, turn the 3-position switch back to sleep and hit go.

If you wish to view the calibration data, please view section 4.3.4 for details.

3.2. Preparing the X-Beam for Assembly

The X-Beam and rails must be cleaned and inspected for no significant scratches, dents, or extreme deformation. The rails are cleaned with Lithium-based grease and the X-beam is cleaned with Lithium-based grease as well.

3.3. Loading

The user loads the XBAS by taking the X-Beam prepared as in Section 3.2. and placing one end on the line constraint. Then push the X-Beam slowly into the machine until it reaches the bolt atop the hardstop. If necessary, help the X-Beam up onto the hardstop. DO NOT HIT the beam actuator, the Metro Probe, or the rail actuators.

After the X-Beam has been placed, attach the rails to the X-Beam with the bolts in the holes, but not torqued down.
3.5. Assembly
After the X-Beam has been loaded and the rails loosely attached, select the Assembly mode option on the 3-position switch if you have not already done so and wait for the green LED to turn on if it hasn’t already. After the user has the green LED, they can press the green button to start the assembly. The XBAS will assemble the machine and will turn the green LED light back on, signalling the XBAS has completed its assembly of the X-Beam.

4. Error Handling
When an error occurs, the XBAS will buzz at 1 second intervals and display an LED pattern specific to the error. This section talks about how to handle errors, what each LED pattern means, and give possible solutions to fixing those errors.

4.1. Handling an Error
When an error occurs, hit the go button to turn off the sound. Take your time to figure out what the error was and address it. When you are ready, hit go again and the machine will run.

4.2. LED Patterns, Errors, and Solutions
Below in table 1 is a list of the possible errors that can occur as well as LED patterns that are associated to that error. Possible solutions are proposed in the subsections of this section below the table.

<table>
<thead>
<tr>
<th>Error</th>
<th>Green LED</th>
<th>Yellow LED</th>
<th>Red LED</th>
<th>Relevant Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing File</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>4.2.1.</td>
</tr>
<tr>
<td>No Bolt Pattern File</td>
<td>Blinking</td>
<td>Blinking</td>
<td>Blinking</td>
<td>4.2.1.</td>
</tr>
<tr>
<td>Gantry Actuator Error</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>4.2.2.</td>
</tr>
<tr>
<td>Rail Actuator Error</td>
<td>Blinking</td>
<td>Blinking</td>
<td>Off</td>
<td>4.2.6.</td>
</tr>
<tr>
<td>Beam Actuator Error</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>4.2.4.</td>
</tr>
<tr>
<td>Probe Error</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>4.2.3.</td>
</tr>
</tbody>
</table>
4.2.1. Missing Files
This error occurs if the FileCheck() function detects that a system critical file is missing. The program will write what files are missing to the ErrorReport.txt file.

In order to access this file, the user must connect to the pyboard via the microusb cord. Note, if the user is already connected when the ErrorReport.txt file is generated, the user may have to unplug the cord reconnect. If the user has to do this, please eject the pyboard from the computer as one would eject a USB drive. Once the user has connected to the pyboard, the ErrorReport.txt file can be accessed like how one would access a USB drive.

After seeing the ErrorReport.txt file, reupload the missing files from a backup drive.

The files that should be on the pyboard are:
1. BeamActuator.py
2. boot.py
3. encoder.py
4. Gantry.py
5. Import.py
6. l6470nucleo.py
7. main.py
8. Probe.py
9. RailAct.py
10. setup.py
11. task_share.py

Note, if the issue persists, there are two possible issues not mentioned above. First, it is possible that the FileCheck() function is checking for files that it should not be checking for. There are two fixes for this, both are listed below.

One, replace all files on the pyboard with back up files.

Two, access main.py using a python script editor and find the function FileCheck(). There is a variable called FileList which is a list of strings of all the files that the system checks for. Make sure that all the files it is looking for is on the pyboard.

The second issue is that the missing file may be the Calibration or BoltPatern csv file for
the X-Beam selected by the piano switchboard. See section 5 for creating these two files.

4.2.2. Gantry
This error occurs if the gantry motor has been stalled. Clear the machine of obstructions and run the machine again.

If the problem persists, then check the actuator, leadscrew, and shaft coupling for the gantry. Make sure they all function, are not broken, and are properly installed.

If all of the above systems are fine, check the back emf levels for the gantry’s actuator, if this is set too low, then the actuator may stall out at a much smaller load than it is capable at handling.

4.2.3. Probe
This error occurs if the probe detects is measurements are invalid. This can be caused by one of three things. First, this can be caused by an obstruction which can be solved by ensuring that the probe has a clear path down to the point of measurement and resuming the program.

The second possible problem is similar to the first which is that the probe has attempted to move past its maximum allowed position. This error should only occur if the system is not loaded correctly in which case the error is solved by loading the machine. If the probe is obviously triggering this error well before it should, check the Probe() function in the code and locate the variables ReadingLwrLimit and ReadingUpprLimit. Check that these values are correct.

The third problem has the same solution as the second issue. If the probe is getting readings, but not accepting them, then it’s possible that the range of acceptable values in the program is not wide enough and may have to be changed.

4.2.4. Beam Actuator
If this error occurs, then the beam actuator stalled out. Possible issues is that the beam actuator has been loaded past its maximum rating and cannot lift the X-Beam. Try to run the system once more, making sure no objects or extra weight is being exerted on the
actuator. If the problem persists, then it is likely that there is an issue with the actuator itself. If the actuator is fine, check the back emf levels for the beam actuator, if this is set too low, then the actuator may stall out at a much smaller load than it is capable at handling.

4.2.5. Invalid Switchboard Input
This error occurs when the piano switchboard has an invalid switch combination. First, check the combination. If the combination is correct, check the file Import.py. In the Song() function, there is a section of code of if and else if statements. Check that the switch combination that you have selected has a corresponding variable Length value. Note that these values must be integers.

4.2.6. Rail Actuator
This error occurs if either of the rail actuators stall out outside of a range of permissible values. Possible errors include an obstruction or software limits being incorrect. The former is solved by ensuring the rail actuators have a clear path to the rails they are pressing down on and that said rails are present. The ladder is solved by checking and editing the range of acceptable values defined by the variables floor and ceiling.

5. Adding X-Beam
This section contains step by step directions for creating CalibrationXXX.csv and BoltPatternXXX.csv files, adding those files’ ID code (XXX) to the Import.Song() function, and suggested method for debugging the two csv files, and directions to editing and updating the flowcharts if the user so chooses to.

5.1. Creating the BoltPatternXXX.csv file
1. Open Excel
2. Open the Engineering drawing for the X-Beam you are writing file this for
3. In Column B, write out the distances of the bolts you want bolted on the first pass for the RIGHT SIDE
   a. i.e. write out the distance from the near edge of the X-Beam to the bolt to be torqued in the order you want. For example, torque down every other bolt including the 4th to last bolt. Then add the third to last and the last bolt.
4. In Column A, write R next to all the values you just wrote out
5. On the next row in column B after all the stuff you wrote, repeat step 3 for the LEFT SIDE
6. In Column A, write L next to all the values you just wrote out
7. Highlight all the data you wrote
8. In the toolbar, select Data
   a. At the top 4 tabs over from Home
9. Select Sort, and Sort by column B from smallest to largest.
10. Now in column C and D, repeat steps 3 to 6 (with column C instead of A and D instead of B) except now you are recording the positions of all the bolts that haven’t been called out in steps 3 to 6
11. Sort these data points (Sort by column D) from largest to smallest
12. Copy and paste these below the data in column A and B, then delete the source data in C and D
13. Insert two lines above the data (right click on row number to add rows)
14. Copy all data in A and B, then right click on cell A1 and say paste special, Transpose.
15. Delete the source data (ie everything that you didn’t paste)
16. Save File as BoltPatternXXX.csv where XXX is the identifying number of this X-Beam. Recommend using the X-Beam Length. XXX must be the same as CalibrationXXX.csv for this X-Beam

---

**Figure 3. Image of top view of X-Beam with labels of which side is right and left with regards to the linear encoder.**

---

**5.2. Creating the CalibrationXXX.csv file**

Write out the data from cells in A1 to A2 as identified below.

A2) The distance to the far side of the X-Beam. Indicated by variable FarSide.

Delete all other data. Note, A3-A5 are used by the program, so if you’re editing the calibration csv file, don’t change these or you will have to run the calibration mode.

Save File as CalibrationXXX.csv where XXX is the identifying number of this X-Beam. Recommend using the X-Beam Length. XXX must be the same as BoltPatternXXX.csv for this X-Beam.

5.3. Adding the ID code to Import.py
1. Open the program Import.py, preferably using a python editor such as spyder3.
2. Scroll down to all of the “elif Number == ...” statements in the function Import.
3. Choose a Number and replace the ErrSong.put(1) with Length = XXX where XXX is a number.
   o Recommended that XXX is the length of the X-Beam
   o If there are no numbers left, you can add numbers up to a total of 32 options.
4. When done, save and close the program.

5.4. Optional csv file Debugging
If you want to check or debug either of the csv files, connect a computer to the pyboard using the micro-usb cable and a software like putty. Run the Import.BoltPattern() or Import.Calibration() functions through the command window. If there is an error, they should print an error indicating the error, which row, which spot, and what is in the spot that the function identified as an incorrect input.

5.5. Optional Documenting
It is recommended that you change the visio program flow chart/state diagram to reflect your changes as it is easier to read than the program for some. Requires Microsoft Visio 2016 or later versions.

1. Open the state diagram with Visio and navigate to the Import.Song() page using the tabs at the bottom of the window.
   a. If you don’t see the tab, there should be an AllΔ tab. Click on that and a drop up menu should appear with all the pages in the document should appear.
2. On the right side, there are many boxes indicating ErrSong.put(1). Change the one you changed in the program by checking the diamond blocks for the Number associated to that ID.
## 6. Machine Service

Table 2. Summary of Maintenance Schedule for Parts

<table>
<thead>
<tr>
<th>Part</th>
<th>Manufacturer</th>
<th>How Often Replace/Calibrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rails and Bearings</td>
<td>IKO</td>
<td>6 months</td>
</tr>
<tr>
<td>Rail Actuator Extensions</td>
<td>Misumi USA</td>
<td>6 months</td>
</tr>
<tr>
<td>Torque Limiters</td>
<td>Fix it Sticks</td>
<td>6 months</td>
</tr>
<tr>
<td>Metro M-60 Probe</td>
<td>Heidenhain</td>
<td>6 months</td>
</tr>
<tr>
<td>Gantry Actuator Leadscrew</td>
<td>Misumi USA</td>
<td>6 months</td>
</tr>
<tr>
<td>Gantry Actuator Nut</td>
<td>Misumi USA</td>
<td>6 months</td>
</tr>
<tr>
<td>Gantry Actuator Support</td>
<td>Misumi USA</td>
<td>12 months</td>
</tr>
<tr>
<td>Hardstop</td>
<td>Made in House</td>
<td>12 months</td>
</tr>
<tr>
<td>LEDs</td>
<td>Digi-Key</td>
<td>Undetermined</td>
</tr>
<tr>
<td>Position Switch</td>
<td>Digi-Key</td>
<td>Undetermined</td>
</tr>
<tr>
<td>Green Button</td>
<td>Digi-Key</td>
<td>Undetermined</td>
</tr>
<tr>
<td>Red Button</td>
<td>Digi-Key</td>
<td>Undetermined</td>
</tr>
<tr>
<td>Soft Point Constraint (Set Screw)</td>
<td>Misumi USA</td>
<td>6 months</td>
</tr>
</tbody>
</table>
Appendix R: Reference Documents