Indiana University-Purdue University Fort Wayne
Department of Civil and Mechanical Engineering
And
Department of Electrical and Computer Engineering
ENGR 410 – ENGR 411
Capstone Senior Design Project
Report #2

Project Title: Rolling Drum

Sponsor: Custom Engineered Wheels

Team Members:
Robert Crouch (EE)
Brandon Kloer (ME)
Marie Langford (EE)
Scott McNerney (ME)
Erin Sheehan (ME)
Aaron Steffen (EE)

Faculty Advisors:
Dr. Elizabeth Thompson (EE)
Dr. Nashwan Younis (ME)

Date: April 24, 2017
Acknowledgements

The team is tremendously grateful for CEW’s sponsorship and support throughout the duration of this project. The team would like to extend special thanks to Tony Chalk, Jeff Peters, Dave Wagoner, Carl Saldivar, and Jeremy Howard.

Dr. Thompson and Dr. Younis provided helpful advice and guidance during the completion of this project. The team would like to thank both advisors for their commitment.

Additional thanks go out to Jason Moyer for lending his expertise and spare time to assisting us throughout the entire project and Devin Allen for his assistance.

The team would lastly like to thank the IPFW Engineering departments for their support and the opportunity to use their labs and equipment.
Abstract

Custom Engineered Wheels produces non-pneumatic wheels for use within a wide variety of industries. These wheels require testing to ensure their durability and reliability. The wheels are pressed against a cylindrical spinning drum causing them to rotate. To simulate a wheel’s environment, an operator can install cleats onto the drum. The cleat will then strike the wheel on each revolution of the drum. The operator can run the test for a specified amount of time to gather information on each wheel. The current testing apparatus has minimal automated features for gathering data, produces loud noise during operation, and lacks adequate safety features for the operators.

The goal of the project this semester is to build the rolling drum wheel testing apparatus based on the previous semester's design and test the device to ensure it fulfills the requirements set forth by the sponsor CEW. The system must be built in such a way that the components are secure and operate properly without causing damage to any system. The mechanical half of the team is responsible for the physical build of the machine, the attachment of the necessary components, and ensuring that the device can operate correctly and safely. The goal of the electrical half of the multidisciplinary team is to ensure that the rolling drum wheel tester is able to be controlled by the user through a Human Machine Interface (HMI) and that the automated features are installed properly so as to ensure functionality. Following the construction of the prototype, the team is to test the ability of the machine to fulfill the goals set forth in the testing parameters memo.
Section I: Building Process
1.1: Redesign of Project

1.1.1: Reasoning

Following the first semester Capstone Senior Design Project presentation, a couple of concerns were raised by the sponsor. The first concern was that building the entire frame out of 8020 T-Slotted aluminum tubing may result in a frame that is not as stable as desired. In order to increase stability and reduce vibration, CEW requested that the senior design team redesign the frame in order to have a more stable welded steel base. It was requested that the safety cage still be constructed out of t-slotted aluminum framing that is mounted to the steel base. The other major concern the sponsor mentioned is that with the vertical orientation of the drum, debris from a failed wheel may fall into other wheels under test and thus damage those wheels. CEW proposed that in order to resolve this design concern, the drum orientation be changed to horizontal with a space below the wheels for the debris to fall without damaging any working components. The mechanical team agreed to comply with these requests; and over the winter break, the rolling drum wheel tester was redesigned with a horizontal drum configuration and the base of the frame made of welded steel. Another design modification that was made at the sponsor’s request was reducing the number of pneumatic cylinders from four to three. This change was made for two reasons, the primary reason being the footprint of the frame limited the amount of space available for cylinders. The space was further limited by the fact that the sponsor also suggested the use of piston guide rails to prevent unnecessary force on the piston rods and to stabilize the wheels during testing. The second reason for reducing the number of pneumatic cylinders was that the team was already over budget and reducing the number of cylinders to three reduced the costs significantly. The motor selection was changed to a 1.5 horsepower, 1200 rpm, 460 volt three-phase motor in order to comply with CEW’s power supply.

1.1.2: Completed Redesign

The redesign of the rolling drum wheel tester resulted in a design with a frame constructed of 2” x 2” x ¼” thick hot rolled square steel tubing MIG welded together. The number of cylinders was reduced to three, and the cylinder pistons were supported on both sides by guide rails. The safety cage was constructed of 25 series T-Slotted aluminum framing from 8020 and 6 mm thick polycarbonate panels. The initial design and completed redesign are shown in Figure 1.
**Figure 1:** Solid model of the rolling drum system original design (left) and the final redesign (right).
1.2: Mechanical Building Process

1.2.1: Frame

The bottom half of the rolling drum device originally began as separate pieces of 2” x 2” x ¼” thick hot rolled steel tubing and ¼” A36 plates that were welded together by Buhrt Engineering. Buhrt Engineering allowed for a few members of the mechanical team to observe and assist in the welding process. Using the prints provided by the mechanical team, the welder MIG welded the frame together. A few notches where the A36 steel plate and tubing met were required to ensure that the pieces fit correctly and were able to be welded together properly.

Once the frame was delivered to IPFW from Buhrt Engineering the mechanical team started working on frame assembly. The first step in getting the frame ready was to level the frame relative to the ground so that when the drum was placed, it could be leveled relative to the frame. The next step of the process was to drill the holes for mounting the components. Holes were required for mounting the cylinders, the guide rails for the fixture arms, for the sheet metal skin, and for mounting the motor. Figure 2 shows a team member measuring and marking the locations of holes for mounting the fixture arm guide rails. While looking at the assemblies for the rolling drum device, the team measured for hole location twice on the frame and then center punched the mark for the holes. The holes were all measured relative to a common reference line along the back side of the frame. Once all of the marks were punched, the team then verified that the components would line up with the center punch marks before drilling. After each mark was verified, the holes were drilled and then checked again. The final step of the frame process was painting the entire frame black as requested by CEW. Before painting of the frame began, however, the entire frame was washed down using acetone to remove excess oil and debris in order to ensure the paint would adhere properly to the frame.
1.2.2: Pneumatic Cylinders

The pistons used in the rolling drum design were Festo standard cylinders with an 80 mm bore and a 320 mm stroke. The pistons were mounted using Festo cylinder foot mounts which were bolted to the ¼” A36 steel plate using four M12 locking nuts and bolts. The cylinders were mounted approximately one inch from the back edge of the frame in order to position the cylinders such that wheels as large as 27” in diameter will have room to be mounted. This cylinder positioning also allows for wheels as small as 4” in diameter to be mounted in such a way that they are able to make contact with the drum given the extension of the pistons. Festo rod couplers were used at the end of the pistons in order to compensate for any potential misalignment between the cylinders and the fixture arms as they extend along the guide rails. Finally, Festo one way control valves were used to attach the pneumatic tubing. This allows for the user to manually control the speed of the piston extension and thus prevent potential damage to the wheels or other components.

1.2.3: Fixture Arms

The designed fixture arms were sent out to be manufactured by Diversified Metals in Fort Wayne. The mechanical team created DXF files for each arm so that Diversified Metals could water jet cut each of the fixture arm components. These pieces were then TIG welded together and delivered to the IPFW campus. Once the fixture arms were received, they were put in place on the guide rail carriages in order to verify that the measurements were correct and the arms would mount properly to the carriages. Two out of the three arms’ holes aligned perfectly with the bolts on the guide rail carriages, but the third arm had a fork that was bent slightly and thus did not allow proper alignment. This was corrected by applying pressure to both sides of the fork until the sides bent into the proper parallel alignment. The M20 holes in the center of the fixture
arm fork were tapped by hand in order to thread them onto the piston rod couplers. The fixture arms were then bolted onto the guide rail carriages using M6 bolts and onto the ends of the couplers that were attached to the end of the cylinders. Figure 3 shows the fixture arms bolted to the coupler and guide rail carriages. The next step was to cut the axles for mounting the wheels to the correct length of seven inches. Wheels were then placed into the arms and it was verified that the axles and the fixture arms were compatible with the various sized wheels that were provided by CEW.

![Figure 3: The fixture arms mounted to the guide rail carriages as well as the piston rod coupler.](image)

### 1.2.4: Drum

The primary component of the rolling drum is a 33” long piece of 10” diameter structural pipe. The end caps of the drum are fabricated from 1” thick A36 steel plate which was machined down to 10” diameter disks at IPFW by Jason Moyer. The center axle of the drum is a 1.5” 1045 turned, ground, and polished steel rod which is welded to holes in the center of the drum end caps. The TIG welding of the drum was completed by Diversified Metals in Fort Wayne, and the
welded drum was turned in order to ensure that the outside diameter was consistent and smooth. An 8M pitch, 90 tooth QD bushed HTD pulley was mounted onto the center axle of the drum using a QD bushing and ⅜” key stock. The drum was mounted to the frame using pillow block bearings. The drum was lifted into the frame using an engine hoist and the bearings were bolted to the frame using M12 bolts and flanged locking nuts. The drum mounted into the frame is shown in Figure 4.

![Figure 4: The drum mounted into the frame before being balanced.](image)

After mounting the drum it was evident that it was not balanced. There was a heavy side to the drum which was most likely a result of a weld in the structural pipe. In order to balance the drum and thus reduce excessive vibration, weights were added on the light side of the drum and material was drilled out of the end caps on the heavy side. Jason Moyer machined six weights which were bolted to the end cap. In order to bolt the weights onto the end cap, the mechanical team first drilled and tapped 6M bolt holes. Figure 5 shows weights taped to the end of the drum in order to test the drum balance.
Figure 5: Weights taped to the side of the drum to determine how much weight would need to be added to the rolling drum in order to obtain balance.

The weights alone did not balance the drum so holes ½” in diameter and ½” in depth were drilled into the 1” thick end caps on the heavy side of the rolling drum. This material removal made it possible to balance the drum and thus reduce vibration. Figure 6 shows how the holes were drilled into the end of the drum.
Figure 6: Holes being drilled into the end caps of the drum in order to help balance the drum.

Finally, testing of wheels with cleats was one of the project goals. In order to do so, holes were drilled and tapped into the surface of the drum; and one cleat was bolted onto the rolling drum.

1.2.5: Safety Cage

The T-Slotted aluminum and the polycarbonate panels for the safety cage arrived on pallets and the mechanical team started the safety cage building process. Anchor fasteners were used to secure the counterbored aluminum framing. The safety cage was built independent of the steel frame with the intention of attaching the safety cage to the frame once all assembly on the frame was completed. The mechanical team started the cage assembly by first arranging all of the aluminum components on a large open floor. The T-Slotted aluminum was assembled one side at a time using locktite on the bolts of the anchors to ensure that they would not become loose during machine operation. Figure 7 shows pieces of the frame being connected using the counterbore anchors.
The doors of the frame were completed first and then the sides containing the doors were assembled. The doors to the safety cage did not initially fit due to the fact that the tolerances of the T-Slotted aluminum were too tight to allow clearance above and below the doors. The doors were then taken apart and a quarter inch of the aluminum was removed to reduce the height of the door using a bandsaw. The polycarbonate panels of the doors were also cut down a quarter of an inch to allow for proper fitting. The safety hinges and locks were then installed to ensure the doors remained secured in place. Once the cage assembly was completed it was lifted onto the frame to verify that it was sized correctly to mount to the frame. Figure 8 shows the safety cage resting on top of the frame. It was confirmed that the safety cage dimensions matched with the frame; and holes were marked for the brackets, which would attach the safety cage to the frame. Following the drilling of holes for the brackets, the cage was removed to allow the electrical team to mount their components on the rolling drum device. The use of an arm to mount the HMI to the rolling drum device was investigated, but it was later decided that the better option would be to place the HMI in the polycarbonate panel next to one of the doors. The necessary polycarbonate panel was then taken out of the cage and a hole for the HMI was measured and cut out by the mechanical team. The hole for the HMI was cut by drilling holes at the four corners of the rectangular hole and connecting them with a jigsaw. The manual pressure regulators were also mounted on the same panel as the HMI, with holes drilled for bolts, power cables, and air lines. Finally, the safety enclosure was placed on top of the frame and bolted on utilizing mending brackets.
Figure 8: The safety cage mounted on top of the steel frame in order to check for appropriate dimensions and fit.
1.2.6: Motor Drive and Pulleys

In order to mount the motor, the 30 tooth HTD QD bushed pulley was first attached to the motor shaft. The motor was then aligned onto the mounting plate underneath the rolling drum and aligned such that it was parallel with the drum itself. The HTD belt was then placed on the pulley mounted to the drum as well as the pulley mounted to the motor shaft. The motor was then aligned so that the motor shaft was parallel to the drum shaft and the belt was aligned and vertically leveled as shown in Figure 9.

![Figure 9: Leveling of the HTD belt in order to ensure the motor is mounted in the correct position under the rolling drum.](image)

Once in position, the mounting holes for the motor were marked and drilled. The motor was mounted in place using M10 bolts and lock nuts. In order to ensure proper belt tension during operation, a tensioner was designed by the engineering team and constructed using a ⅜” hot rolled steel bar cut to length and a 60 mm long idler roller. This belt tensioner was welded to the steel frame and adjusted to ensure proper placement and tension on the HTD belt. The tensioner can be adjusted in the future to ensure that the belt will always run appropriately.
1.3: Electrical Building

1.3.1: Human Machine Interface (HMI)

One of the main goals established for the project by CEW was to automate the testing process. Therefore, the system was designed to utilize a Human Machine Interface (HMI). The HMI is a device which allows the operator to set up and observe tests through a touchscreen display. On this project, the HMI performs the majority of the logic and processing for the testing automation.

A Kadet G307K200 HMI made by Red Lion was selected for this project because CEW currently utilizes this HMI on other equipment currently on their production floor. CEW will have to maintain the software in the future, and their engineers and technicians are already familiar with this product.

The Kadet model HMI features a 7” color backlit touchscreen, making it easy for an operator to see and interact with the device. It has the capability to perform serial communication using RS-232 and RS-485 with its two onboard serial ports. It also has an SD card slot to allow new software to be loaded into the system without being tethered via serial cable to a computer.

The Kadet is programmed using Crimson, which is a proprietary Red Lion software suite. Crimson is designed to allow users to program an HMI without an extensive software background, and utilizes a significant amount of drag-and-drop functionality, as well as features such as setting properties from dialog boxes. However, to achieve more complex functionality on the HMI, Crimson also allows C code to be added. All C code created for this project is included in attached Appendix A.

To create the software for the HMI, Crimson’s drag-and-drop features were used to sketch out the initial layout of the pages of the user interface. Figure 10 shows the layout of the main menu page. After the layout was created by dragging boxes onto the screen, functionality was added to tie the buttons to associated pages when pressed. Within the box properties, it is also possible to dynamically change colors of the boxes and to enable or disable the pushbutton functionality. In this page, the “Stop Test” button is gray and deactivated because there is currently no test running. When a test is running, the “Stop Test” button becomes green, and the “Start Test” button is set to gray and is deactivated, so that no new test can be started until the current test finishes or is cancelled.
A new test is started by selecting the “New Test” button. When a new test is started, the user has the option of starting a test with one, two, or all three of the cylinders on the machine. This allows the user the ability to test a single wheel at a time, or to run multiple tests simultaneously. Figure 11 shows how a user would enable or disable a cylinder when starting a test. In this screen capture, the second wheel has been disabled and will currently not be used in the test. To change this, the user can select the “Enable” button, which will set the second wheel to be active, and will cause a “Disable” button to display. This allows the user to toggle the cylinders from activated to deactivated.
After a wheel has been either enabled or disabled as desired, the “Next” button will take the user to the next page. If the wheel has been enabled, the next page will allow the user to enter the diameter of the wheel under test on that cylinder, as well as the desired force to be applied. The requirements for the project specified that the range of diameters of wheels to be tested on this machine were 4-27 inches, so the data entry feature limits the inputs to be within that range. The data entry keypad shown in Figure 12 is activated by pushing the blue button displaying the current set value for wheel diameter or applied force.

Next, the user can enter the number of cleats mounted on the drum. The user may enter a value from 0-10, which should correspond to the number of cleats which have been physically mounted onto the drum for that particular wheel. This data entry screen is shown in Figure 13. It is possible to mount different numbers or types of cleats on the three different wheels; therefore, the number of cleats is entered separately for each wheel. Tests can be run with or without cleats, so the program will allow the user to enter ‘0’ as the number of cleats, and will allow up to 10 cleats to be entered. The HMI will save the last entered test parameters, and therefore if the new test is utilizing some of the same values as the previous test, it will not be necessary to re-enter these values. The user may simply click “Next” to move to the next page and will not need to bring up the numeric keypad.
Once the number of cleats has been set, the final parameter to set for the wheel is the total test time. This is set in hours, minutes, and seconds. Each is set separately by selecting the blue data box; this brings up the numeric keypad to allow the user to input the desired value. The ‘Sec’ and ‘Min’ boxes will allow values from 0-59, while the ‘Hr’ box will allow the user to enter a value up to a maximum of 999 hours. This extreme upper range is to allow the option of running the wheels until they fail.
These pages are repeated for each of the enabled wheels in order to set all the necessary parameters. Finally, the overall testing speed must be set. This is shown below in Figure 15.
The last step before beginning the test is to review the settings which have been chosen. By displaying the chosen settings before beginning a test, the operator can determine whether an error has been made before starting the test. This page to review the settings can be seen below in Figure 16. In this figure, it can be seen that only cylinders 1 and 3 have wheels under test. When the user has reviewed the settings, the test can be activated by pressing the “Start Test” button. If the operator determines that he does not wish to start a test, he may adjust the test settings by going back the appropriate page using the “Back” buttons, or may return back to the main menu by selecting the “Home” button.

![Review Test Settings](image)

**Figure 16: Review Test Settings**

When the “Start Test” button is selected, the HMI performs processing to set up and activate the test. Proper register values for some of the PLC outputs are calculated and passed to the PLC. The HMI sets the timers for each cylinder, and resets the variables tracking the necessary data to be collected. Finally, the HMI sends a flag to the PLC to start the test, which signals to the PLC to activate its outputs. The HMI will wait to receive a flag from the PLC indicating that all active wheels have reached the drum and the drum has begun spinning before starting the timers on the PLC.

As each cylinder completes its set testing time, the HMI sends a flag to the PLC to retract the wheel under test. When the last wheel is finished (or is retracted due to failure or user
cancellation), the HMI will then send a stop test flag to the PLC to stop the motor and reset the PLC outputs.

After the “Start Test” button is selected, the user may choose to go back to the main menu via the “Home” button. After 5 seconds, the screen will time-out to the Test Status page. This page may also be accessed from the main menu via the “Current Test Status” button. For each wheel, this page shows settings such as the wheel diameter, applied force, and total test time. These settings are shown in the top group of displayed data for each wheel. The second group of data shows the data which is being collected for each wheel in real time. From the set speed of the drum and the total elapsed time, the number of revolutions that the drum has completed can be calculated. Then, the number of cleat hits is calculated by multiplying the revolutions of the drum by the number of cleats mounted on the drum for that wheel. The distance traveled for the wheel is calculated by multiplying the speed of the wheel by the total elapsed test time. Finally, the revolution count of the wheel is determined by dividing the distance the wheel has traveled by its circumference. These values, along with the elapsed run time and wheel status, are updated once per second. The page is shown below in Figure 17. The processing to calculate the data and monitor the testing time was done by adding backend C code to the software.

![Figure 17: Test Status Page](image)

The “Status” displayed at the bottom of the data for each wheel will display the current status of each cylinder. Currently, the status for both of the wheels under test is “RUNNING.” When the
test finishes, the status will change to “COMPLETED.” If the wheel encounters a failure condition such as an overstroke (“OVERSTROKE”), or fails to extend to the drum at the start of the test (“DID NOT EXTEND”), the status will display the error. If the user cancels the test, the status will display “CANCELLED.” Once all three cylinders have completed their tests, the green box displaying “Status: Test Running” will turn red and will display “Status: Test Completed.” An example of a screen after a test finishes running is shown below in Figure 18.

![Status Screen with Completed Test](image)

**Figure 18:** Status Screen with Completed Test

If the user wishes to cancel the test for one or more wheels, he may do so from the “Stop Test” button on the main menu. This will bring up the page shown in Figure 19. A single cylinder may be cancelled by selecting the respective cylinder button. If desired, all active cylinders may be cancelled by selecting the “STOP ALL” button. To ensure that the operator does not accidentally cancel a test, the cancel cylinder buttons require a confirmation as a layer of protection. In case of an emergency, the operator can instantly shut off the cylinders by pressing the emergency stop button located on the outside of the machine.
The HMI software stores the past five sets of completed test data. This data can be accessed from the main menu via the “Stored Test Data” which is shown in Figure 20. The test data is listed by test date, starting with the most recent. When a new test completes, the data in Test 5 is discarded, each set of test data is moved down one slot, and the new test data is stored in Test 1. The timestamp marks the time at which the test was begun.

The data displayed in this section is stored in the Kadet’s non-volatile memory. This means that the data will be safely stored even through a power cycle.
Once the desired test data is selected, it will bring up the corresponding test data in a window as shown in Figure 21. As seen in the test data, the first wheel ran successfully for 30 seconds and completed just over 20 revolutions. It also successfully encountered 98 cleat hits without failure. No wheel was tested on cylinder 2. After 22 seconds of testing, wheel 3 was cancelled. Wheel 3 did not have any cleats mounted on the drum, and so encountered 0 cleat hits during its 22 second test period. However, it did travel 0.0125 miles over its short test duration.
In case any issues are encountered, the operator has the ability to access I/O status debug pages, which display the input, output, and flag statuses from the PLC. These pages are accessed from the “PLC I/O & Flags Status” button on the main menu. This enables the user via the HMI to determine how the PLC is functioning and what may be going wrong. Figure 22 shows the page for the PLC flags; the pages for the PLC inputs and outputs are very similar.
Finally, from the main menu page, the HMI will time out to a machine status page as shown in Figure 23. This page will display a “ticker” across the top showing whether there is a currently running test, and what if any alarms are present in the system. The screen capture below shows the highly unusual case that upon starting the test, all three wheels failed to extend to the drum. When this is the case, the orange alarm shown below will appear on the screen. If the wheel begins to deform and extends to the overstroke position sensor, an overstroke alarm will trigger, and a red box stating “WHEEL X: OVERSTROKE” will appear in the same place as the orange “WHEEL X: FAILED TO EXTEND” warning is currently shown. Touching anywhere on this screen will take the user to the main menu.
The alarms are tracked by monitoring alarm flags on the PLC. When the HMI observes an alarm flag trigger on the PLC, it will set an alarm on the HMI to display the warning above, and will call the program function to retract the cylinder which encountered the error. This process allows for the test to automatically detect and retract in the case of a wheel failure.
1.3.2: Programmable Logic Controller (PLC)

The PLC, in coordination with the HMI, provides the automation for wheel testing. The PLC used for this project was the CPX-FEC-1-IE model. The Festo controller utilizes the Festo Software Toolset (FST) to program and it is important to select the correct controller model in the project settings when creating a new project within FST. It provides a statement-list or ladder logic approach. Statement-list was chosen to program the PLC due to the familiarity of CEW’s employees with statement-list style syntax. To provide precise control and a wide variety of test settings in the rolling drum machine, the I/O needs to be configured first. The I/O configuration provides the correct addresses for each input and output to be synchronized with the physical PLC setup. The PLC is equipped with (2) 8 digital input modules, (2) 2 analog output modules, (1) 4 digital output module, and (2) 4 pneumatic output modules. Each module needs to be selected from the catalog provided within the FST. If the module is not correct, the Festo Maintenance Tool (FMT) will diagnose which module is not correctly identified. Most model numbers are found directly on the modules themselves and can be determined by process of elimination if needed. This is essential to successfully download a project to the PLC. Shown below in Figure 24 is the final I/O configuration for the rolling drum machine.

![Figure 24: Final I/O configuration for the rolling drum machine](image)

From the I/O configuration it is easy to see the addresses for each the input words (IW column) and the output words (OW column). These addresses provide the starting address for each bit.
However, each module has different capabilities and bit registers. To decipher which capabilities are present within the module, it is possible to bring up the module settings which show all selected modules and how many bits are available in each to the programmer. The modules used each contain 4 sockets, but not all are necessarily accessible. For example, the analog output module is 2x16 output bits, and only uses two of the four sockets available. The digital input module, on the other hand, has 8x1 input bits available providing two inputs per socket on the module. Shown below in Figures 25 and 26 is an example of each of the referred modules.

![Figure 25: Analog output module](image)

![Figure 26: Digital input module](image)

In order to control the inputs and outputs of the system, a reference page, or allocation list, was established. This list created a location to view all current inputs and outputs (operands). Also, each operand was given an assignment. For instance, the digital input operand I0.1 that consists of the emergency shutoff to de-energize the system had to be interpreted by the PLC. All of the position sensors (9 total with 3 on each cylinder) had to be defined such that the wheel testing could be automated. The PLC would read the digital input from each sensor with the status “RETRACTED”, “AT DRUM”, or “OVERSTROKE”. Along with the input and output operands were the flags. Flags were designated for use with communication between the PLC and HMI. These bits can be coordinated between the HMI and PLC to interpret various scenarios. An example would be F55.0 ("Stop/Retract CYL 1" flag), that is set only when the user selects to retract cylinder 1. Timers from the PLC were used to signify a failure to reach the drum within a time limit and de-energize the respective individual cylinder. Registers were also used to store pre-scaled values calculated by the HMI to control analog outputs. During the progress of the project, operands were added and omitted in order to increase automation and ease of use. The final allocation list including all operands as well as all of the PLC code is in Appendix B.

Programming the digital portion of the PLC was simply a matter of setting inputs and outputs by their respective individual bits. The individual bits were represented by their plug in location to the socket within the physical module. An example would be the operands I0.0 ("Reset Pushbutton") and I0.1 ("E-stop"). These specific inputs had to be correctly wired and plugged
into socket 1 of the left most digital input module in order to provide consistent addressing for remaining inputs on the module. Each input and output was interpreted based on the reference shown on the allocation list and ultimately configured in a manner to increase automation during wheel testing. Providing analog outputs to the three pressure regulators and the VFD required a slightly different tactic. To determine how to output an analog signal ranging from 0 V to 10 V, CEW referred the team to Allied Automation, Inc. A representative was able to provide the register format for each of the two outputs on the analog module, shown below in Figure 27.

![Figure 27: Register format for the analog output 16-bit](image)

It was determined that bits D0-D11 provided the scaling for the analog output. It was also determined that the scaling values for the output would be a value of 0 to represent 0 V and a value of $2^{12}-1 = 4095$ representing the max output voltage at 10 V. The actual scaling for the force and speed was done via the HMI and then stored into a register within the PLC to use at the appropriate time. The analog output allowed for the system to utilize all three proportional pressure regulators to control individual cylinders, as well as control the motor using the VFD.

The final project consists of a total of 7 PLC programs with separate functions. The first of which is the start up program, P0. The start up program consists of initializing all operands and programs to 0, setting the system pressure via the soft-start valve using a reset button, awaiting a start test select from the user, and a while loop structure for an emergency stop or stop test from the user. Each case is handled such that the operands and programs will be re-initialized and then proceed amongst the same conditions as before.

The cylinder sequence programs (P1-P3) control each individual pneumatic cylinder. Each program has a flag to check whether the cylinder selected on the HMI (i.e. F55.0 “Stop/Retract CYL 1”). When set, the position sensor on the cylinder designated as “RETRACTED” is checked to make sure the cylinder is retracted completely to its home position. When these conditions are met the pneumatic valve output is set, the analog output to the respective proportional pressure regulator is designated by user input, and a timer is started in the PLC. In
the event the timer expires and the cylinder has not reached the “AT DRUM” position, the
cylinder is retracted and requires a re-select. Otherwise, the programs each enter into a loop
which checks for an input on the “OVERSTROKE” sensor (wheel failure) as well as a flag check
for the user to retract the cylinder.

Program 4 controls loading a scaled value from the HMI (0-4095) to the VFD analog output via
Register 4. The program then loops with a check for a stop test select by the user or the event
that the cylinders are all retracted. The analog register load program (P5) is used to constantly
update each register with the user input for force by each cylinder and/or speed of the drum. The
error handling program (P63) is used solely for an emergency stop situation. Upon an emergency
stop, the error word and the startup program are reset and then set to begin from the very start of
the cycle.

Downloading to the PLC required an RS-232 connection cable. The project also had to be built
in order to create an object file for the download. Within the controller settings, the project is
chosen to download source files which allows for upload of current project to the PLC. An
online tool within the FST was used to diagnose bugs in the code quickly. It allows the
programmer to check where the code is currently being evaluated as well as check the status of
all used operands in the code. The PLC would also display an error status LED indicator to show
an error within the downloading process. To test recently downloaded code, the PLC needed a
hard reset.

1.3.3: Wiring

The connection diagram of all electrical components is in Appendix C. CEW has a three-phase
480 VAC wall supply in their lab to power the current drum and requires the same supply for the
replacement apparatus. The three-phase power is used to run the motor through the VFD as well
as to power a 24 VDC power supply. This connection is shown in Figure 28 with terminals L1,
L2, and L3 representing the inputs to the VFD and terminals T1, T2, and T3 as the output to the
motor.
Figure 28: Connection diagram of 460 VAC power

Figure 29 below shows the connections to the 24 VDC supply to power the HMI and the PLC. There are three normally closed switches on the high voltage line as a safety precaution. The first switch is a pushbutton emergency stop, and it is followed by a safety switch on each door of the cage. With the current setup, pushing the emergency stop button or opening either door will cut power to all of the components, which will in turn release all of the air pressure from the lines to immediately stop the test. An RS-232 cable is used to communicate between the HMI and the PLC.

Figure 29: Connection diagram of 24 VDC power

The connection of the inputs and outputs to the PLC is shown below in Figure 30. Input 0.0 contains a normally open “Reset” push button. This button was requested by CEW in order to
turn on the soft-start valve and supply air to the pneumatics. This request was made in order to ensure that air will only be applied to the machine when intentionally applied. The emergency stop circuit is also connected to I0.1 in order to be monitored by the PLC. I0.5 through I1.5 are all connected to Festo CPS-AP-F magnetic position sensors. The pinouts of the input module and the magnetic position sensors are shown in Figure 31 below. For example, the sensors connected I1.0 and I1.1 both have Pin 1 connected to Pin 1 and Pin 3 connected to Pin 3 on the input module, but Pin 4 on I1.0 is connected to Pin 4 and Pin 4 on I1.1 is connected to Pin 2 on the module as there are two inputs per socket on the CPX-8DE module.

![Connection diagram of I/O](image)

**Figure 30:** Connection diagram of I/O

![Pinout of digital input and magnetic position sensors](image)

**Figure 31:** Pinout of digital input and magnetic position sensors

The analog output modules are used to control the VFD and the pressure regulators. The pinout of the analog module is shown in Figure 32 and that of the VFD in Figure 33. It was determined
that Pins 2 and 4 from the module were to be connected to Pins 13 and 14 on the VFD in order to control it with the 0-10 V signal. The pressure regulators were then connected to the remaining analog outputs with pins 1 to 1, 2 to 4, 3 to 3, and 4 to 2 in order to match to polarities of the 0-10 V. This can be seen on the output and regulator pinouts in Figures 32 and 34, respectively.

![Figure 32: Analog output pinout](image)

![Figure 33: VFD pinout](image)

![Figure 34: Pressure regulator pinout](image)

The digital output module is used to control the soft-start valve through O0.0 with the Pins 1 and 2 from the valve connecting to 4 and 3 on the output module. This pinout is because Pins 3 and 4
on the output module are 0 V and 24 V, as is shown below in Figure 35. O0.1 is then connected
to the coil of a normally open relay in order to close a connection between Pins 2 and 11 on the
VFD to begin spinning the motor in the forward direction as is shown in the pinout in Figure 33.
Pin 11 on the VFD is also used to supply power to the brake and stop the motor quickly at the
end of a test or upon an emergency stop.

Figure 35: Digital output pinout
Section II: Testing and Evaluation
2.1: Testing Parameters

Parameters:

1. Must be capable of variable speed ranging from 0-6 mph in at least 0.5 mph increments -- The drum will be rotated at various speeds and the speed will be recorded. The speed will be tested in 0.5 mph increments and verified using a hand held tachometer. The test will be successful when the drum reaches 6 mph and when the speed can be controlled in 0.5 mph increments.

2. Automatic shut-off or retraction of fixture arm when introduced to wheel failure -- Wheels will be tested until failure to ensure that they will retract in the case of failure. Intentionally weakened wheels may be used to expedite the process and ensure proper response to failure.

3. Automatic shut off if cage is open -- The wheel tester will be run at various speeds and the safety cage will be opened. If the device properly powers down, then the automatic shut-off is working properly. This test will be run multiple times to determine the time required to dissipate stored energy from various components including the rolling drum and the pneumatic cylinders.

4. Must be laterally stable -- The device will be run at various speeds ranging from the lowest speed, 0.5 mph to the highest speed, 6 mph under various cleat conditions. Lateral movement will be recorded and measured to determine maximum lateral movement. Maximum lateral movement should be under one eighth of an inch.

5. Must allow for the attachment of different wheels ranging from 4” to 27” -- Wheels of various sizes and types ranging from 4” to 27” will be mounted on the fixture and tested to ensure that they attach to the fixture arm securely and that they make full contact with the rolling drum during the testing process.

6. Must allow up to 400 lbs of force to each wheel -- The device will use pneumatic cylinders to generate force on the attached wheels. This will be tested at various forces up to 400 lbs in order to verify that a range of forces can be used on independent wheel fixtures. The force will be verified using hand calculations.

7. Data collection of speed, run time, revolutions, and number of cleat hits -- Several short (5 min or less) tests will be run and data will be collected. Data that is collected will be verified through hand calculations.

8. Programmable start and end times -- The device will be programmed to start and end at various times for multiple trials and the proper function and timing will be verified.
2.2: Test Procedure

The first testing parameter to be verified was the ability for the drum’s capability of various speeds ranging from 0-6 mph in 0.5 mph increments. In order to test this the drum was to be tested at various speeds in those 0.5 mph increments and the speed was then verified through the use of a handheld tachometer. The testing began with calibration at the 10 mph setting in order to ensure accuracy of results. The drum speed was controlled through the HMI and various speeds beginning with 10 mph and decreasing in the .5 mph increments were tested.

The second test involved shut off and retraction of the fixture arm when presented with wheel failure. In order to test this function the fixture arms were allowed to extend beyond a “failure point”, which is the point to which the fixture arm would extend in the case of a wheel failure. A wheel was intentionally damaged by an employee of CEW in order to ensure failure within a reasonable testing window. The wheel was then mounted on the fixture arm and tested at various forces, speeds, and with or without cleats in order to determine if the system would detect failure and retract the arm.

The next test was to determine if the automatic shutoff switches functioned properly when the safety cage was opened during operation. In order to test this parameter, the machine was operated under normal conditions with the drum rotating and the fixture arms extended. For the test to be determined successful, the machine was required to power down and exhaust pressure to the cylinders. For additional safety verification, the time for kinetic energy in the drum to dissipate was also recorded.

Lateral stability of the wheels during machine operation was a design objective and as such required testing to verify stability. In order to determine the lateral stability of the wheels, the rolling drum machine was operated under normal conditions with cleats and without. The operation was recorded from an angle perpendicular to the contact between the wheel and the drum and lateral movement was analyzed using tracker software. This test is determined to be successful if lateral movement is less than ⅛ inch in either direction.

The size of wheels that can be mounted on to the fixture arms and properly tested was to be determined. The system must be capable of testing wheels anywhere from 4” to 27” based on the original requirements set forth by CEW. In order to test this, wheels of various sizes were mounted onto the fixture arms from the smallest available to the largest. This test is passed if the wheels make contact with the rolling drum when the machine is in use without binding.

Another requirement of the rolling drum wheel tester was that it be able to deliver forces of up to 400 lbs on wheels independently. Force on the wheels was applied utilizing pneumatic cylinders.
In order to determine what force was applied, simple calculations could be performed. In order to safely test the force that is capable of being applied to the wheels, pressure readings were recorded utilizing both the digital pressure regulators and an analog pressure gage. These readings were compared against each other and against the input from the HMI, and the force applied at this pressure was calculated. These forces were then compared against each other to verify the accuracy and range of forces. Success in this case is determined by being able to deliver a range of different forces from 0-400 lbs to all three cylinders simultaneously.

The rolling drum wheel tester records data on the speed, run time, revolutions, and number of cleat hits of the wheels under test. The HMI simply records the commanded output speed, which was verified above. In order to verify that the HMI accurately recorded and tracked run time, a test was started on the HMI, and at the same time, a stopwatch was started. The HMI time and stopwatch time matched perfectly. CEW requested that revolutions and cleat hits be calculated based on existing measured values, rather than installing separate sensors to track these values. Therefore, the revolutions and cleat hits are calculated from the total test time and the motor speed, both of which have been tested and verified. Therefore, since these values are calculated from previously validated values, these calculated values can be considered to be validated as well.

The final test that was performed was to verify that the rolling drum was capable of programmable start and end times. In order to verify that this function worked properly, several tests were run with various start and end times and were timed manually. If the machine started and ended at the correct times the test was determined to be successful.
2.3: Testing Results

The drum speed test results were intended to determine that the drum could be operated from 0.5-6 mph in 0.5 mph increments. The sponsor asked that the drum be able to operate at up to 10 mph if possible, so the testing began at 10 mph and was tested at each 0.5 mph increment until it reached 0.5 mph. Rotations per minute were recorded using an Advent optical tachometer manufactured by Compact Instruments Limited. The rotations per minute were also calculated using the circumference of the drum at the same speeds for comparison. These speeds are shown in Figure 36.

![Figure 36: Comparison between the calculated speed and rpm of the drum and the speed of the drum that was recorded using the digital tachometer.](image)

The resolution of the digital tachometer only provided readings of full rotations per minute and thus created uncertainty in our results. At speeds 1.5 mph and lower, the digital tachometer was unable to give a reading. Thus the rpm for 0.5, 1 and 1.5 mph were taken manually which also resulted in uncertainty and a larger percent error in our calculated versus experimental results. The results and percent error can be seen in Table 1. These results were within an acceptable range of error as prescribed by CEW.

<table>
<thead>
<tr>
<th>Table 1: Drum Speed Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculated Speed (MPH)</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2.5</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3.5</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>4.5</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>5.5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6.5</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>7.5</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8.5</td>
</tr>
</tbody>
</table>
The test to ensure minimal lateral movement of the fixture arms and wheels was conducted next. Video footage was recorded and analyzed using Tracker video analysis software in order to measure lateral movement. A screenshot of the video analysis using tracker is shown in Figure 37.

![Screenshot of the analysis performed on the lateral movement of a wheel making contact with the rolling drum. Movement was automatically tracked and the data was saved in a table.](image)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>8.993</td>
<td>284.232</td>
<td>284</td>
<td>-0.081544613</td>
</tr>
<tr>
<td>9.5</td>
<td>9.488</td>
<td>300.022</td>
<td>300</td>
<td>-0.007476077</td>
</tr>
<tr>
<td>10</td>
<td>10.006</td>
<td>315.813</td>
<td>316</td>
<td>0.059185606</td>
</tr>
</tbody>
</table>

Figure 37: Screenshot of the analysis performed on the lateral movement of a wheel making contact with the rolling drum. Movement was automatically tracked and the data was saved in a table.

Results of the video analysis indicated that lateral movement of the fixture arms themselves was negligible to the point of being imperceptible when analysing the recordings of the tests. When tests were performed on the lateral movement of a wheel that was mounted properly using Tracker, the maximum lateral movement was 0.1059 inches which is below the requisite maximum lateral movement of 0.125 inches. Figure 38 shows a graph of the position of the edge of the wheel with respect to time. Data tables which were created using the Tracker software which indicate change in position over time are included in the Appendix D.
The next test that was performed was to verify that the device would purge air pressure and cut power in the case of the safety cage door opening during operation. There is a door on either side of the rolling drum wheel tester and each is equipped with a safety shut off door hinge. In order to test if the machine would shut off and purge air when a door opens, several trials were run. The machine was powered up, the drum was spun, and pressure was applied through the cylinders. Each door was opened under these conditions at least ten times and in each case the safety shut off hinges activated. This caused the air to automatically purge from the system and the motor to cease rotation. The rolling drum dissipated energy in under 5 seconds when being rotated at 10 miles per hour and less for lower speeds. This test was deemed successful and under no trial did the machine fail to shut down when the doors were opened.

Testing was performed to ensure that a wide range of wheel styles and sizes could be mounted and tested. Wheels from 6 inches in diameter to 25 inches in diameter were mounted and tested with and without cleats mounted to the drum. All wheels tested were able to be mounted to the fixture arms and made proper contact with the drum during testing. For the purposes of this project, axle inserts accommodating wheels with center axles of ¾ inches and ½ inches were used, though inserts could be machined to allow mounting of a wider range of wheels.

Another requirement of the design is that it be able to apply various forces up to 400 lbs on each of the wheels being tested. It was deemed too dangerous by our sponsor to test force directly and would require disabling the safety protocols in order to do so. Instead, to verify that the rolling drum wheel tester is capable of this, a series of tests was run and the pressure was verified using...
both a digital pressure regulator and an analog pressure gage. Different levels of force were input into the HMI and the resulting pressure reading from the digital regulator and the analog gage were recorded.

![Digital pressure regulators and the analog pressure gage](image)

**Figure 39:** Digital pressure regulators and the analog pressure gage which were used to determine the force applied.

These pressures were compared to each other as well as the calculated pressure to determine that the appropriate force was being applied. A table of the pressures recorded and the force applied are shown in Table 2.

**Table 2:** Force and Pressure Data
<table>
<thead>
<tr>
<th>Force (lbs)</th>
<th>Set Pressure (psi)</th>
<th>Digital Reading (psi)</th>
<th>Analog Reading (psi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>3.209</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>50</td>
<td>6.418</td>
<td>6.5</td>
<td>7</td>
</tr>
<tr>
<td>75</td>
<td>9.626</td>
<td>9.6</td>
<td>10</td>
</tr>
<tr>
<td>100</td>
<td>12.835</td>
<td>12.7</td>
<td>14</td>
</tr>
<tr>
<td>125</td>
<td>16.044</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>150</td>
<td>19.253</td>
<td>19.4</td>
<td>20</td>
</tr>
<tr>
<td>175</td>
<td>22.461</td>
<td>22.6</td>
<td>22</td>
</tr>
<tr>
<td>200</td>
<td>25.670</td>
<td>25.8</td>
<td>27</td>
</tr>
<tr>
<td>250</td>
<td>32.088</td>
<td>32.3</td>
<td>33</td>
</tr>
<tr>
<td>300</td>
<td>38.505</td>
<td>38.8</td>
<td>39</td>
</tr>
<tr>
<td>350</td>
<td>44.923</td>
<td>45.2</td>
<td>46</td>
</tr>
<tr>
<td>400</td>
<td>51.340</td>
<td>51.7</td>
<td>52</td>
</tr>
</tbody>
</table>

The resulting pressure data is what was expected with the exception being the reading from the analog gage. The difference can be explained by the lack of resolution in the analog gage which gives an approximated pressure reading. The pressure and force results are shown in graphical form in Figure 39.
**Figure 40:** Pressure readings given by the digital pressure regulators, the analog reading, and the pressure set in the HMI as well as the resulting calculated force.
Section III: Cost and Budget
3.1: Budget

The initial budget for this project was set at $10,000 and the engineering team was able to propose a budget that was under $10,000 based on the premise that the assembly and fabrication would not factor into the cost. Upon further review at the end of the semester, Custom Engineered Wheels determined that the budget was far too restrictive for the IPFW engineering team to be able to deliver a design which incorporated all of the desired features. With this in mind the CEW contact, Tony Chalk, informed the team that the budget was being increased to $14,136.90 to meet the latest cost estimate but with the understanding that there would be additional expenses not included in this total. This extension in the budget was due primarily to the fact that the initial budget of $10,000 was a rough, flexible estimate with little consideration of what the actual budget should be. The budget increase ended up being necessary due to the fact that labor and fabrication costs quickly added to the final cost. Welding of the steel frame and rolling drum alone cost a total of $3,127 which is approximately one third of the entire initial budget. Welding and fabrication of the fixture arms was another $795 and $332 was added to the budget due to labor and tooling for IPFW. These additions as well as others quickly drove the final total to $16,442.53 which though above budget, the sponsor reported that they were highly satisfied with the final cost. Figure 41 shows how the budget was portioned out to the different assemblies. Whereas Figure 42 shows how each disciplinary used the budget. The final budget is shown in Appendix E.

![Pie chart showing the cost of assembly](image)

**Figure 41:** Assembly cost.
Figure 42: Amount of total budget used by each disciplinary.
Conclusion

The purpose of this senior design project was to build a rolling drum wheel testing system for Custom Engineered Wheels. The system designed was required to test 4 to 6 wheels simultaneously in varying wheel sizes from 4” to 27” in diameter. A drum was to be constructed of inelastic material with an outer diameter of 250mm ±25 mm. The system needed to be able to apply loads of up to 400 lbs on a drum capable of rotating at speeds up to 6 mph in 0.5 mph increments.

The rolling drum project was completed on time and is capable of meeting the design requirements. The final build was completed over the initial budget but was within the adjusted budget set by CEW. The frame was built from 2” x 2” square hot rolled steel tubing which provided a heavy stable base capable of withstanding the forces and vibration produced during testing. The rolling drum is highly automated and is capable of simultaneously testing three wheels of diameters ranging from 4” to 27”. The use of pneumatic cylinders allowed for the application of force up to 400 lbs and also reduced noise levels during testing.

Test start and end times are automated and data is collected automatically. The system is capable of detecting wheel failure and stopping the test of the failed wheel. It will also shut off and automatically release all of the air pressure if the safety cage is opened while it is running or if the emergency stop button is pressed.
Appendix A: HMI Code
void StartTest(void)

// This program is called from the button listener of the Start Test button

// -------------------------------
// Set HMI Variables and flags:
// -------------------------------

// ----- Cleats Mounted -----
CylinderValues.hmi.Cleats.Cleats_set_1 = CylinderValues.hmi.Cleats.Cleats_temp_1;

// ----- Set Cleat Hits to Zero -----
TestStatus.CleatHits_1 = 0;
TestStatus.CleatHits_2 = 0;
TestStatus.CleatHits_3 = 0;

// ----- Wheel Sizes ----- 
CylinderValues.hmi.Wheel_1_Size_set = CylinderValues.hmi.Wheel_1_Size_temp;
CylinderValues.hmi.Wheel_2_Size_set = CylinderValues.hmi.Wheel_2_Size_temp;
CylinderValues.hmi.Wheel_3_Size_set = CylinderValues.hmi.Wheel_3_Size_temp;

// ----- Cylinder Forces ----- 
CylinderValues.hmi.Wheel_1_force_set = CylinderValues.hmi.Wheel_1_force_temp;
CylinderValues.hmi.Wheel_2_force_set = CylinderValues.hmi.Wheel_2_force_temp;
CylinderValues.hmi.Wheel_3_force_set = CylinderValues.hmi.Wheel_3_force_temp;

// ----- Calculate RPM of each wheel ----- 
// RPM = [ speed(mi/hr) * (1 hr / 60 min) * (5280 ft / mi) * (12 in / ft) ] / [wheel circumference(in / revolution)]
// Circumference = pi * diameter
TestStatus.rpm_1 = ((float)Speed.TestSpeed * (1/(float)60) * (5280) * (12)) / (Pi() * CylinderValues.hmi.Wheel_1_Size_set);
TestStatus.rpm_2 = ((float)Speed.TestSpeed * (1/(float)60) * (5280) * (12)) / (Pi() * CylinderValues.hmi.Wheel_2_Size_set);
TestStatus.rpm_3 = ((float)Speed.TestSpeed * (1/(float)60) * (5280) * (12)) / (Pi() * CylinderValues.hmi.Wheel_3_Size_set);

// ----- Set Revolutions / Distance Traveled to Zero ----- 
TestStatus.Revolutions_1 = 0;
TestStatus.Revolutions_2 = 0;
TestStatus.Revolutions_3 = 0;
TestStatus.Distance_1 = 0;
TestStatus.Distance_2 = 0;
TestStatus.Distance_3 = 0;
StartTest

// ----- Timers -----

// Calculate total run seconds for each wheel

// Set each of the "set" time strings to the temp values

// Set elapsed hr/min/sec to zero
Timing.set.Cylinder_1.elapsed_hrs_1 = 0;
Timing.set.Cylinder_1.elapsed_mins_1 = 0;
Timing.set.Cylinder_1.elapsed_secs_1 = 0;

Timing.set.Cylinder_2.elapsed_hrs_2 = 0;
Timing.set.Cylinder_2.elapsed_mins_2 = 0;
Timing.set.Cylinder_2.elapsed_secs_2 = 0;

Timing.set.Cylinder_3.elapsed_hrs_3 = 0;
Timing.set.Cylinder_3.elapsed_mins_3 = 0;
Timing.set.Cylinder_3.elapsed_secs_3 = 0;

Timing.set.Cylinder_1.tot_elapsed_secs = 0;
Timing.set.Cylinder_2.tot_elapsed_secs = 0;
Timing.set.Cylinder_3.tot_elapsed_secs = 0;

// Set "Test Finished" flags. If wheel is used, set to 0. If not used, set to 1 (cylinder is finished)
CylinderFlags.Cyl_1_TestEnded = !CylinderFlags.Wheel1_Used;
CylinderFlags.Cyl_2_TestEnded = !CylinderFlags.Wheel2_Used;
CylinderFlags.Cyl_3_TestEnded = !CylinderFlags.Wheel3_Used;

// Set the elapsed time strings
Timing.updateElapsedTimeStrings();

// ----- Test Start/Stop Flags -----
TestStatus.start_test=1;
TestStatus.end_test=0;

TestStatus.Test_Status = "Status: Running";
if (CylinderFlags.Wheel1_Used) TestStatus.Status_Cylinder_1 = "RUNNING";
else TestStatus.Status_Cylinder_1 = "NOT USED";
if (CylinderFlags.Wheel2_Used) TestStatus.Status_Cylinder_2 = "RUNNING";
else TestStatus.Status_Cylinder_2 = "NOT USED";
if (CylinderFlags.Wheel3_Used) TestStatus.Status_Cylinder_3 = "RUNNING";
else TestStatus.Status_Cylinder_3 = "NOT USED";


// -------------------------------
// Set flags on PLC:
// -------------------------------

TestStatus.plc_start_test = 1;
TestStatus.plc_end_test = 0;

CylinderFlags.plc_cylinder_1_select = CylinderFlags.Wheel1_Used;
CylinderFlags.PLC_STOP_CYL_1 = 0;

CylinderFlags.plc_cylinder_2_select = CylinderFlags.Wheel2_Used;
CylinderFlags.PLC_STOP_CYL_2 = 0;

CylinderFlags.plc_cylinder_3_select = CylinderFlags.Wheel3_Used;
CylinderFlags.PLC_STOP_CYL_3 = 0;

// Calculates register scaled values for force output and sets PLC registers
CylinderValues.plc.plc_force_scaled_1 = UtilityMethods.RoundInt(CylinderValues.hmi.Wheel_1_force_set * CylinderValues.plc.FORCE_TO_PSI_SCALE);
CylinderValues.plc.plc_force_scaled_2 = UtilityMethods.RoundInt(CylinderValues.hmi.Wheel_2_force_set * CylinderValues.plc.FORCE_TO_PSI_SCALE);
CylinderValues.plc.plc_force_scaled_3 = UtilityMethods.RoundInt(CylinderValues.hmi.Wheel_3_force_set * CylinderValues.plc.FORCE_TO_PSI_SCALE);

// Set scaled VFD output register value
Speed.plc_vfd_scaled = UtilityMethods.RoundInt(Speed.TestSpeed*Speed.VFD_SPEED_TO_REGISTER_SCALE);
cancelCylinder

void cancelCylinder(int CylinderNumber, cstring CylinderStatusMessage)

// CylinderNumber corresponds to wheel 1,2,3. Sets HMI and PLC flags low, and sets the stop cylinder flag on PLC high.

// This program is called from cylinder alarm listeners, from the timer (when it expires), and
// from the "cancel cylinder" buttons on the HMI.

if((CylinderNumber == 1) && !CylinderFlags.Cyl_1_TestEnded)
{
    CylinderFlags.Cyl_1_TestEnded = 1;
    CylinderFlags.plc_cylinder_1_select = 0;
    CylinderFlags.PLC_STOP_CYL_1 = 1;
    TestStatus.Status_Cylinder_1 = CylinderStatusMessage;
}
else if ((CylinderNumber == 2) && !CylinderFlags.Cyl_2_TestEnded)
{
    CylinderFlags.Cyl_2_TestEnded = 1;
    CylinderFlags.plc_cylinder_2_select = 0;
    CylinderFlags.PLC_STOP_CYL_2 = 1;
    TestStatus.Status_Cylinder_2 = CylinderStatusMessage;
}
else if ((CylinderNumber == 3) && !CylinderFlags.Cyl_3_TestEnded)
{
    CylinderFlags.Cyl_3_TestEnded = 1;
    CylinderFlags.plc_cylinder_3_select = 0;
    CylinderFlags.PLC_STOP_CYL_3 = 1;
    TestStatus.Status_Cylinder_3 = CylinderStatusMessage;
}
EndTest

void EndTest()

// Called when all cylinders are retracted or when the user chooses to end entire test

// -------------------------------
// Set HMI Variables and flags:
// -------------------------------

TestStatus.start_test = 0;
TestStatus.end_test = 1;
TestStatus.Test_Status = "Status: Test Finished";

CylinderFlags.Cyl_1_TestEnded = 1;
CylinderFlags.Cyl_2_TestEnded = 1;
CylinderFlags.Cyl_3_TestEnded = 1;

// -------------------------------
// Set flags on PLC:
// -------------------------------

TestStatus.plc_start_test = 0;
TestStatus.plc_end_test = 1;

// -------------------------------
//     -----  Save Data  -----  
// -------------------------------

saveData();
void Timing.counter(void)

// This program is called once each second from the system OnTick listener

if (TestStatus.start_test && PLC_OUTPUTS.VFD_ENABLE)
    if (TestStatus.start_test)
        {
            // Update elapsed time hr/min/secs and tickers of each cylinder
            Timing.updateElapsedTime();
            // Update calculations of revolutions / cleat hits / etc.
            Timing.updateTestData();
            // Decrement each test timer
            if (CylinderFlags.Wheel1_Used && !CylinderFlags.Cyl_1_TestEnded)
                {
                    Timing.set.Cylinder_1.test_timer_1 =
                    Timing.set.Cylinder_1.test_timer_1 - 1;
                    Timing.set.Cylinder_1.tot_elapsed_secs =
                    Timing.set.Cylinder_1.tot_elapsed_secs + 1;
                    if (Timing.set.Cylinder_1.test_timer_1 == 0)
                        {
                            // Cancel cylinder 1
                            cancelCylinder(1, "COMPLETED");
                        }
                }
            if (CylinderFlags.Wheel2_Used && !CylinderFlags.Cyl_2_TestEnded)
                {
                    Timing.set.Cylinder_2.test_timer_2 =
                    Timing.set.Cylinder_2.test_timer_2 - 1;
                    Timing.set.Cylinder_2.tot_elapsed_secs =
                    Timing.set.Cylinder_2.tot_elapsed_secs + 1;
                    if (Timing.set.Cylinder_2.test_timer_2 == 0)
                        {
                            // Cancel cylinder 2
                            cancelCylinder(2, "COMPLETED");
                        }
                }
            if (CylinderFlags.Wheel3_Used && !CylinderFlags.Cyl_3_TestEnded)
                {
                    Timing.set.Cylinder_3.test_timer_3 - 1;
                    Timing.set.Cylinder_3.tot_elapsed_secs =
                    Page 1
counter
Timing.set.Cylinder_3.tot_elapsed_secs + 1;

    if (Timing.set.Cylinder_3.test_timer_3 == 0)
    {
        // Cancel cylinder 3
        cancelCylinder(3, "COMPLETED");
    }
}

// If all cylinders have finished, then end test
if (CylinderFlags.Cyl_1_TestEnded && CylinderFlags.Cyl_2_TestEnded &&
    CylinderFlags.Cyl_3_TestEnded)
{
    EndTest();
}

} else
{

}
saveData

void saveData(void)

//Discard 5th test, load tests 1-4 into 2-5, load new data into test 1

// ---------------------------------------- LOAD TEST 4 TO TEST 5 ----------------------------------------

SavedTestData.Test_5.Total_Runtime_Wheel_1 = SavedTestData.Test_4.Total_Runtime_Wheel_1;
SavedTestData.Test_5.Total_Runtime_Wheel_2 = SavedTestData.Test_4.Total_Runtime_Wheel_2;
SavedTestData.Test_5.Total_Runtime_Wheel_3 = SavedTestData.Test_4.Total_Runtime_Wheel_3;

SavedTestData.Test_5.Set_Time_Wheel_1 = SavedTestData.Test_4.Set_Time_Wheel_1;
SavedTestData.Test_5.Set_Time_Wheel_2 = SavedTestData.Test_4.Set_Time_Wheel_2;
SavedTestData.Test_5.Set_Time_Wheel_3 = SavedTestData.Test_4.Set_Time_Wheel_3;

SavedTestData.Test_5.Test_Timestamp = SavedTestData.Test_4.Test_Timestamp;
SavedTestData.Test_5.Speed = SavedTestData.Test_4.Speed;

SavedTestData.Test_5.Force_Cyl_1 = SavedTestData.Test_4.Force_Cyl_1;
SavedTestData.Test_5.Force_Cyl_2 = SavedTestData.Test_4.Force_Cyl_2;
SavedTestData.Test_5.Force_Cyl_3 = SavedTestData.Test_4.Force_Cyl_3;

SavedTestData.Test_5.Diameter_Wheel_1 = SavedTestData.Test_4.Diameter_Wheel_1;
SavedTestData.Test_5.Diameter_Wheel_2 = SavedTestData.Test_4.Diameter_Wheel_2;

SavedTestData.Test_5.Status_Wheel_1 = SavedTestData.Test_4.Status_Wheel_1;
SavedTestData.Test_5.Status_Wheel_2 = SavedTestData.Test_4.Status_Wheel_2;

SavedTestData.Test_5.Wheel1_Used = SavedTestData.Test_4.Wheel1_Used;
SavedTestData.Test_5.Wheel2_Used = SavedTestData.Test_4.Wheel2_Used;
SavedTestData.Test_5.Wheel3_Used = SavedTestData.Test_4.Wheel3_Used;

SavedTestData.Test_5.rpm_1 = SavedTestData.Test_4.rpm_1;
SavedTestData.Test_5.rpm_2 = SavedTestData.Test_4.rpm_2;
SavedTestData.Test_5.rpm_3 = SavedTestData.Test_4.rpm_3;

SavedTestData.Test_5.Distance_1 = SavedTestData.Test_4.Distance_1;
SavedTestData.Test_5.Distance_2 = SavedTestData.Test_4.Distance_2;
SavedTestData.Test_5.Distance_3 = SavedTestData.Test_4.Distance_3;

SavedTestData.Test_5.Revolutions_1 = SavedTestData.Test_4.Revolutions_1;
SavedTestData.Test_5.Revolutions_2 = SavedTestData.Test_4.Revolutions_2;
saveData
SavedTestData.Test_5.Revolutions_3 = SavedTestData.Test_4.Revolutions_3;
SavedTestData.Test_5.CleatHits_1 = SavedTestData.Test_4.CleatHits_1;
SavedTestData.Test_5.CleatHits_2 = SavedTestData.Test_4.CleatHits_2;
SavedTestData.Test_5.Cleats_set_1 = SavedTestData.Test_4.Cleats_set_1;
SavedTestData.Test_5.Cleats_set_2 = SavedTestData.Test_4.Cleats_set_2;

// ------------------------------------------------------------------------------
// ------------------------ LOAD TEST 3 TO TEST 4 -------------------------------
// ------------------------------------------------------------------------------
SavedTestData.Test_4.Total_Runtime_Wheel_1 = 
SavedTestData.Test_3.Total_Runtime_Wheel_1;
SavedTestData.Test_4.Total_Runtime_Wheel_2 = 
SavedTestData.Test_3.Total_Runtime_Wheel_2;
SavedTestData.Test_4.Total_Runtime_Wheel_3 = 
SavedTestData.Test_3.Total_Runtime_Wheel_3;
SavedTestData.Test_4.Set_Time_Wheel_1 = SavedTestData.Test_3.Set_Time_Wheel_1;
SavedTestData.Test_4.Set_Time_Wheel_2 = SavedTestData.Test_3.Set_Time_Wheel_2;
SavedTestData.Test_4.Test_Timestamp = SavedTestData.Test_3.Test_Timestamp;
SavedTestData.Test_4.Speed = SavedTestData.Test_3.Speed;
SavedTestData.Test_4.Force_Cyl_1 = SavedTestData.Test_3.Force_Cyl_1;
SavedTestData.Test_4.Force_Cyl_2 = SavedTestData.Test_3.Force_Cyl_2;
SavedTestData.Test_4.Diameter_Wheel_1 = SavedTestData.Test_3.Diameter_Wheel_1;
SavedTestData.Test_4.Diameter_Wheel_2 = SavedTestData.Test_3.Diameter_Wheel_2;
SavedTestData.Test_4.Status_Wheel_1 = SavedTestData.Test_3.Status_Wheel_1;
SavedTestData.Test_4.Wheel1_Used = SavedTestData.Test_3.Wheel1_Used;
SavedTestData.Test_4.Wheel2_Used = SavedTestData.Test_3.Wheel2_Used;
SavedTestData.Test_4.Wheel3_Used = SavedTestData.Test_3.Wheel3_Used;
SavedTestData.Test_4.rpm_1 = SavedTestData.Test_3.rpm_1;
SavedTestData.Test_4.rpm_2 = SavedTestData.Test_3.rpm_2;
SavedTestData.Test_4.rpm_3 = SavedTestData.Test_3.rpm_3;
saveData
SavedTestData.Test_4.Distance_1 = SavedTestData.Test_3.Distance_1;
SavedTestData.Test_4.Distance_2 = SavedTestData.Test_3.Distance_2;
SavedTestData.Test_4.Distance_3 = SavedTestData.Test_3.Distance_3;

SavedTestData.Test_4.Revolutions_1 = SavedTestData.Test_3.Revolutions_1;
SavedTestData.Test_4.Revolutions_2 = SavedTestData.Test_3.Revolutions_2;
SavedTestData.Test_4.Revolutions_3 = SavedTestData.Test_3.Revolutions_3;

SavedTestData.Test_4.CleatHits_1 = SavedTestData.Test_3.CleatHits_1;
SavedTestData.Test_4.CleatHits_2 = SavedTestData.Test_3.CleatHits_2;

SavedTestData.Test_4.Cleats_set_1 = SavedTestData.Test_3.Cleats_set_1;
SavedTestData.Test_4.Cleats_set_2 = SavedTestData.Test_3.Cleats_set_2;

// ------------------------------------------------------------------------------
// ------------------------ LOAD TEST 2 TO TEST 3 -------------------------------
// ------------------------------------------------------------------------------

SavedTestData.Test_3.Total_Runtime_Wheel_1 = SavedTestData.Test_2.Total_Runtime_Wheel_1;
SavedTestData.Test_3.Total_Runtime_Wheel_2 = SavedTestData.Test_2.Total_Runtime_Wheel_2;
SavedTestData.Test_3.Total_Runtime_Wheel_3 = SavedTestData.Test_2.Total_Runtime_Wheel_3;

SavedTestData.Test_3.Set_Time_Wheel_1 = SavedTestData.Test_2.Set_Time_Wheel_1;
SavedTestData.Test_3.Set_Time_Wheel_2 = SavedTestData.Test_2.Set_Time_Wheel_2;

SavedTestData.Test_3.Test_Timestamp = SavedTestData.Test_2.Test_Timestamp;
SavedTestData.Test_3.Speed = SavedTestData.Test_2.Speed;

SavedTestData.Test_3.Force_Cyl_1 = SavedTestData.Test_2.Force_Cyl_1;
SavedTestData.Test_3.Force_Cyl_2 = SavedTestData.Test_2.Force_Cyl_2;
SavedTestData.Test_3.Force_Cyl_3 = SavedTestData.Test_2.Force_Cyl_3;

SavedTestData.Test_3.Diameter_Wheel_1 = SavedTestData.Test_2.Diameter_Wheel_1;
SavedTestData.Test_3.Diameter_Wheel_2 = SavedTestData.Test_2.Diameter_Wheel_2;

SavedTestData.Test_3.Status_Wheel_1 = SavedTestData.Test_2.Status_Wheel_1;
SavedTestData.Test_3.Status_Wheel_2 = SavedTestData.Test_2.Status_Wheel_2;

SavedTestData.Test_3.Wheel1_Used = SavedTestData.Test_2.Wheel1_Used;
SavedTestData.Test_3.Wheel2_Used = SavedTestData.Test_2.Wheel2_Used;
saveData
SavedTestData.Test_3.Wheel3_Used = SavedTestData.Test_2.Wheel3_Used;

SavedTestData.Test_3.rpm_1 = SavedTestData.Test_2.rpm_1;
SavedTestData.Test_3.rpm_2 = SavedTestData.Test_2.rpm_2;
SavedTestData.Test_3.rpm_3 = SavedTestData.Test_2.rpm_3;

SavedTestData.Test_3.Distance_1 = SavedTestData.Test_2.Distance_1;
SavedTestData.Test_3.Distance_2 = SavedTestData.Test_2.Distance_2;
SavedTestData.Test_3.Distance_3 = SavedTestData.Test_2.Distance_3;

SavedTestData.Test_3.Revolutions_1 = SavedTestData.Test_2.Revolutions_1;
SavedTestData.Test_3.Revolutions_2 = SavedTestData.Test_2.Revolutions_2;
SavedTestData.Test_3.Revolutions_3 = SavedTestData.Test_2.Revolutions_3;

SavedTestData.Test_3.CleatHits_1 = SavedTestData.Test_2.CleatHits_1;
SavedTestData.Test_3.CleatHits_2 = SavedTestData.Test_2.CleatHits_2;
SavedTestData.Test_3.CleatHits_3 = SavedTestData.Test_2.CleatHits_3;

SavedTestData.Test_3.Cleats_set_1 = SavedTestData.Test_2.Cleats_set_1;
SavedTestData.Test_3.Cleats_set_2 = SavedTestData.Test_2.Cleats_set_2;

// ------------------------------------------------------------------------------
// ------------------------ LOAD TEST 1 TO TEST 2 -------------------------------
// ------------------------------------------------------------------------------

SavedTestData.Test_2.Total_Runtime_Wheel_1 = SavedTestData.Test_1.Total_Runtime_Wheel_1;
SavedTestData.Test_2.Total_Runtime_Wheel_2 = SavedTestData.Test_1.Total_Runtime_Wheel_2;
SavedTestData.Test_2.Total_Runtime_Wheel_3 = SavedTestData.Test_1.Total_Runtime_Wheel_3;

SavedTestData.Test_2.Set_Time_Wheel_1 = SavedTestData.Test_1.Set_Time_Wheel_1;
SavedTestData.Test_2.Set_Time_Wheel_2 = SavedTestData.Test_1.Set_Time_Wheel_2;
SavedTestData.Test_2.Set_Time_Wheel_3 = SavedTestData.Test_1.Set_Time_Wheel_3;

SavedTestData.Test_2.Test_Timestamp = SavedTestData.Test_1.Test_Timestamp;
SavedTestData.Test_2.Speed = SavedTestData.Test_1.Speed;

SavedTestData.Test_2.Force_Cyl_1 = SavedTestData.Test_1.Force_Cyl_1;
SavedTestData.Test_2.Force_Cyl_2 = SavedTestData.Test_1.Force_Cyl_2;
SavedTestData.Test_2.Force_Cyl_3 = SavedTestData.Test_1.Force_Cyl_3;

SavedTestData.Test_2.Diameter_Wheel_1 = SavedTestData.Test_1.Diameter_Wheel_1;
SavedTestData.Test_2.Diameter_Wheel_2 = SavedTestData.Test_1.Diameter_Wheel_2;
SavedTestData.Test_2.Diameter_Wheel_3 = SavedTestData.Test_1.Diameter_Wheel_3;
SavedTestData.Test_2.Status_Wheel_1 = SavedTestData.Test_1.Status_Wheel_1;
SavedTestData.Test_2.Status_Wheel_2 = SavedTestData.Test_1.Status_Wheel_2;
SavedTestData.Test_2.Status_Wheel_3 = SavedTestData.Test_1.Status_Wheel_3;

SavedTestData.Test_2.Wheel1_Used = SavedTestData.Test_1.Wheel1_Used;
SavedTestData.Test_2.Wheel2_Used = SavedTestData.Test_1.Wheel2_Used;
SavedTestData.Test_2.Wheel3_Used = SavedTestData.Test_1.Wheel3_Used;

SavedTestData.Test_2.rpm_1 = SavedTestData.Test_1.rpm_1;
SavedTestData.Test_2.rpm_2 = SavedTestData.Test_1.rpm_2;
SavedTestData.Test_2.rpm_3 = SavedTestData.Test_1.rpm_3;

SavedTestData.Test_2.Distance_1 = SavedTestData.Test_1.Distance_1;
SavedTestData.Test_2.Distance_2 = SavedTestData.Test_1.Distance_2;
SavedTestData.Test_2.Distance_3 = SavedTestData.Test_1.Distance_3;

SavedTestData.Test_2.Revolutions_1 = SavedTestData.Test_1.Revolutions_1;
SavedTestData.Test_2.Revolutions_2 = SavedTestData.Test_1.Revolutions_2;
SavedTestData.Test_2.Revolutions_3 = SavedTestData.Test_1.Revolutions_3;

SavedTestData.Test_2.CleatHits_1 = SavedTestData.Test_1.CleatHits_1;
SavedTestData.Test_2.CleatHits_2 = SavedTestData.Test_1.CleatHits_2;
SavedTestData.Test_2.CleatHits_3 = SavedTestData.Test_1.CleatHits_3;

SavedTestData.Test_2.Cleats_set_1 = SavedTestData.Test_1.Cleats_set_1;
SavedTestData.Test_2.Cleats_set_2 = SavedTestData.Test_1.Cleats_set_2;
SavedTestData.Test_2.Cleats_set_3 = SavedTestData.Test_1.Cleats_set_3;

// -------------------------------- LOAD NEW TEST TO TEST 1 ---------------------------------
SavedTestData.Test_1.Total_Runtime_Wheel_1 = Timing.set.Cylinder_1.TimeElapsed_1;
SavedTestData.Test_1.Total_Runtime_Wheel_2 = Timing.set.Cylinder_2.TimeElapsed_2;

SavedTestData.Test_1.Set_Time_Wheel_1 = Timing.set.Cylinder_1.TimeSet_1;
SavedTestData.Test_1.Set_Time_Wheel_2 = Timing.set.Cylinder_2.TimeSet_2;

SavedTestData.Test_1.Test_Timestamp = Timing.set.Timestamp;
SavedTestData.Test_1.Speed = Speed.TestSpeed;

SavedTestData.Test_1.Force_Cyl_1 = CylinderValues.hmi.Wheel_1_force_set;
SavedTestData.Test_1.Force_Cyl_2 = CylinderValues.hmi.Wheel_2_force_set;
SavedTestData.Test_1.Force_Cyl_3 = CylinderValues.hmi.Wheel_3_force_set;
saveData
SavedTestData.Test_1.Diameter_Wheel_1 = CylinderValues.hmi.Wheel_1_Size_set;
SavedTestData.Test_1.Diameter_Wheel_2 = CylinderValues.hmi.Wheel_2_Size_set;
SavedTestData.Test_1.Diameter_Wheel_3 = CylinderValues.hmi.Wheel_3_Size_set;
SavedTestData.Test_1.Status_Wheel_1 = TestStatus.Status_Cylinder_1;
SavedTestData.Test_1.Status_Wheel_2 = TestStatus.Status_Cylinder_2;
SavedTestData.Test_1.Status_Wheel_3 = TestStatus.Status_Cylinder_3;
SavedTestData.Test_1.Wheel1_Used = CylinderFlags.Wheel1_Used;
SavedTestData.Test_1.Wheel2_Used = CylinderFlags.Wheel2_Used;
SavedTestData.Test_1.Wheel3_Used = CylinderFlags.Wheel3_Used;
SavedTestData.Test_1.rpm_1 = TestStatus.rpm_1;
SavedTestData.Test_1.rpm_2 = TestStatus.rpm_2;
SavedTestData.Test_1.rpm_3 = TestStatus.rpm_3;
SavedTestData.Test_1.Distance_1 = TestStatus.Distance_1;
SavedTestData.Test_1.Distance_2 = TestStatus.Distance_2;
SavedTestData.Test_1.Distance_3 = TestStatus.Distance_3;
SavedTestData.Test_1.Revolutions_1 = TestStatus.Revolutions_1;
SavedTestData.Test_1.Revolutions_2 = TestStatus.Revolutions_2;
SavedTestData.Test_1.Revolutions_3 = TestStatus.Revolutions_3;
SavedTestData.Test_1.CleatHits_1 = TestStatus.CleatHits_1;
SavedTestData.Test_1.CleatHits_2 = TestStatus.CleatHits_2;
SavedTestData.Test_1.CleatHits_3 = TestStatus.CleatHits_3;
SavedTestData.Test_1.Cleats_set_1 = CylinderValues.hmi.Cleats.Cleats_set_1;
SavedTestData.Test_1.Cleats_set_2 = CylinderValues.hmi.Cleats.Cleats_set_2;
void Timing.updateElapsedTime(void)

//------------------------------------------------------------------------------
//---------------------------------- Wheel 1 ----------------------------------
//------------------------------------------------------------------------------

if(CylinderFlags.Wheel1_Used && !CylinderFlags.Cyl_1_TestEnded) {
    // increment total test time by one second
    if (Timing.set.Cylinder_1.elapsed_secs_1 == 59) {
        if (Timing.set.Cylinder_1.elapsed_mins_1 == 59) {
            // Time is format hh:59:59, set to hh+1:00:00
            Timing.set.Cylinder_1.elapsed_hrs_1 =
            Timing.set.Cylinder_1.elapsed_hrs_1 + 1;
            Timing.set.Cylinder_1.elapsed_mins_1 = 0;
            Timing.set.Cylinder_1.elapsed_secs_1 = 0;
        } else {
            // Time is format hh:mm:59, set to hh:mm+1:00
            Timing.set.Cylinder_1.elapsed_mins_1 =
            Timing.set.Cylinder_1.elapsed_mins_1 + 1;
            Timing.set.Cylinder_1.elapsed_secs_1 = 0;
        }
    } else {
        // Time is format hh:mm:ss, set to hh:mm:ss+1
        Timing.set.Cylinder_1.elapsed_secs_1 =
        Timing.set.Cylinder_1.elapsed_secs_1 + 1;
    }
}

//------------------------------------------------------------------------------
//---------------------------------- Wheel 2 ----------------------------------
//------------------------------------------------------------------------------

if (CylinderFlags.Wheel2_Used && !CylinderFlags.Cyl_2_TestEnded) {
    // increment total test time by one second
    if (Timing.set.Cylinder_2.elapsed_secs_2 == 59) {
        if (Timing.set.Cylinder_2.elapsed_mins_2 == 59) {
            // Time is format hh:59:59, set to hh+1:00:00
            Timing.set.Cylinder_2.elapsed_hrs_2 =
            Timing.set.Cylinder_2.elapsed_hrs_2 + 1;
            Timing.set.Cylinder_2.elapsed_mins_2 = 0;
            Timing.set.Cylinder_2.elapsed_secs_2 = 0;
        } else {
            // Time is format hh:mm:59, set to hh:mm+1:00
            Timing.set.Cylinder_2.elapsed_mins_2 =
            Timing.set.Cylinder_2.elapsed_mins_2 + 1;
            Timing.set.Cylinder_2.elapsed_secs_2 = 0;
        }
    } else {
        // Time is format hh:mm:ss, set to hh:mm:ss+1
        Timing.set.Cylinder_2.elapsed_secs_2 =
        Timing.set.Cylinder_2.elapsed_secs_2 + 1;
    }
}
updateElapsedTime

// Time is format hh:59:59, set to hh+1:00:00
Timing.set.Cylinder_2.elapsed_hrs_2 = Timing.set.Cylinder_2.elapsed_hrs_2 + 1;
Timing.set.Cylinder_2.elapsed_mins_2 = 0;
Timing.set.Cylinder_2.elapsed_secs_2 = 0;

} else
{

// Time is format hh:mm:59, set to hh:mm+1:00
Timing.set.Cylinder_2.elapsed_mins_2 = Timing.set.Cylinder_2.elapsed_mins_2 + 1;
Timing.set.Cylinder_2.elapsed_secs_2 = 0;
}

} else
{

// Time is format hh:mm:ss, set to hh:mm:ss+1
Timing.set.Cylinder_2.elapsed_secs_2 = Timing.set.Cylinder_2.elapsed_secs_2 + 1;
}

//-----------------------------------------------------------------------
//---------------------------------- Wheel 3 ----------------------------------
//-----------------------------------------------------------------------

if (CylinderFlags.Wheel3_Used && !CylinderFlags.Cyl_3_TestEnded) {

    // increment total test time by one second

    if (Timing.set.Cylinder_3.elapsed_secs_3 == 59) {

        if (Timing.set.Cylinder_3.elapsed_mins_3 == 59) {

            // Time is format hh:59:59, set to hh+1:00:00
            Timing.set.Cylinder_3.elapsed_mins_3 = 0;
            Timing.set.Cylinder_3.elapsed_secs_3 = 0;

        } else
{

            // Time is format hh:mm:59, set to hh:mm+1:00

        }

    } else
{

        // Time is format hh:mm:ss, set to hh:mm:ss+1
    }

}
updateElapsedTime
Timing.set.Cylinder_3.elapsed_secs_3 = 0;
}
else
{
    // Time is format hh:mm:ss, set to hh:mm:ss+1
    Timing.set.Cylinder_3.elapsed_secs_3 =
    Timing.set.Cylinder_3.elapsed_secs_3 + 1;
}

Timing.updateElapsedTimeStrings();
void Timing.updateElapsedTimeStrings(void)

// Set the Time strings for the 3 cylinders

if (CylinderFlags.Wheel1_Used) {
    if (Timing.set.Cylinder_1.elapsed_hrs_1 > 99) // if >99, then use 3 digits for hours
        {
            Timing.set.Cylinder_1.TimeElapsed_1 =
            IntToText(Timing.set.Cylinder_1.elapsed_hrs_1,10,3)
            + ":" + IntToText(Timing.set.Cylinder_1.elapsed_mins_1,10,2)
            + ":" +
            IntToText(Timing.set.Cylinder_1.elapsed_secs_1,10,2);
        } else // else, use 2 digits for the hours
        {
            Timing.set.Cylinder_1.TimeElapsed_1 =
            IntToText(Timing.set.Cylinder_1.elapsed_hrs_1,10,2)
            + ":" + IntToText(Timing.set.Cylinder_1.elapsed_mins_1,10,2)
            + ":" +
            IntToText(Timing.set.Cylinder_1.elapsed_secs_1,10,2);
        }
}

if (CylinderFlags.Wheel2_Used) {
    if (Timing.set.Cylinder_2.elapsed_hrs_2 > 99) // if >99, then use 3 digits for hours
        {
            Timing.set.Cylinder_2.TimeElapsed_2 =
            IntToText(Timing.set.Cylinder_2.elapsed_hrs_2,10,3)
            + ":" + IntToText(Timing.set.Cylinder_2.elapsed_mins_2,10,2)
            + ":" +
            IntToText(Timing.set.Cylinder_2.elapsed_secs_2,10,2);
        } else // else, use 2 digits for the hours
        {
            Timing.set.Cylinder_2.TimeElapsed_2 =
            IntToText(Timing.set.Cylinder_2.elapsed_hrs_2,10,2)
            + ":" + IntToText(Timing.set.Cylinder_2.elapsed_mins_2,10,2)
            + ":" +
            IntToText(Timing.set.Cylinder_2.elapsed_secs_2,10,2);
        }
}
updateElapsedTimeStrings

if (CylinderFlags.Wheel3_Used) {

    if (Timing.set.Cylinder_3.elapsed_hrs_3 > 99) // if >99, then use 3 digits
    for hours
    {
        IntToText(Timing.set.Cylinder_3.elapsed_hrs_3,10,3)
        + ":" + IntToText(Timing.set.Cylinder_3.elapsed_mins_3,10,2)
        + ":" +
        IntToText(Timing.set.Cylinder_3.elapsed_secs_3,10,2);
    } else // else, use 2 digits for the hour
    {
        IntToText(Timing.set.Cylinder_3.elapsed_hrs_3,10,2)
        + ":" + IntToText(Timing.set.Cylinder_3.elapsed_mins_3,10,2)
        + ":" +
        IntToText(Timing.set.Cylinder_3.elapsed_secs_3,10,2);
    }
}
updateTempTimeStrings

void Timing.updateTempTimeStrings(void)

// Set the Time strings for the 3 cylinders

if (CylinderFlags.Wheel1_Used) {
    if (Timing.temp.Cyl_1.Hr_temp > 99) // if >99, then use 3 digits for hours
    {
        Timing.temp.Cyl_1.Time_temp = IntToText(Timing.temp.Cyl_1.Hr_temp,10,3)
            + ":" + IntToText(Timing.temp.Cyl_1.Min_temp,10,2)
            + ":" + IntToText(Timing.temp.Cyl_1.Sec_temp,10,2);
    } else // else, use 2 digits for the hours
    {
        Timing.temp.Cyl_1.Time_temp = IntToText(Timing.temp.Cyl_1.Hr_temp,10,2)
            + ":" + IntToText(Timing.temp.Cyl_1.Min_temp,10,2)
            + ":" + IntToText(Timing.temp.Cyl_1.Sec_temp,10,2);
    }
}

if (CylinderFlags.Wheel2_Used) {
    if (Timing.temp.Cyl_2.Hr_temp > 99) // if >99, then use 3 digits for hours
    {
        Timing.temp.Cyl_2.Time_temp = IntToText(Timing.temp.Cyl_2.Hr_temp,10,3)
            + ":" + IntToText(Timing.temp.Cyl_2.Min_temp,10,2)
            + ":" + IntToText(Timing.temp.Cyl_2.Sec_temp,10,2);
    } else // else, use 2 digits for the hours
    {
        Timing.temp.Cyl_2.Time_temp = IntToText(Timing.temp.Cyl_2.Hr_temp,10,2)
            + "":" + IntToText(Timing.temp.Cyl_2.Min_temp,10,2)
            + "":" + IntToText(Timing.temp.Cyl_2.Sec_temp,10,2);
    }
}

if (CylinderFlags.Wheel3_Used) {
    if (Timing.temp.Cyl_3.Hr_temp > 99) // if >99, then use 3 digits for hours
    {
            + "":" + IntToText(Timing.temp.Cyl_3.Min_temp,10,2)
            + "":" + IntToText(Timing.temp.Cyl_3.Sec_temp,10,2);
    } else // else, use 2 digits for the hours
    {
        Timing.temp.Cyl_3.Time_temp = IntToText(Timing.temp.Cyl_3.Hr_temp,10,2)
            + "":" + IntToText(Timing.temp.Cyl_3.Min_temp,10,2)
            + "":" + IntToText(Timing.temp.Cyl_3.Sec_temp,10,2);
    }
}
updateTempTimeStrings
   + ":" + IntToText(Timing.temp.Cyl_3.Min_temp,10,2)
   + ":" + IntToText(Timing.temp.Cyl_3.Sec_temp,10,2);
void Timing.updateTestData(void)

// drum circumference = 33.4375 inches.
// rpm = speed ( mi / hr) * [(5280 ft / mi) * (12 in / ft) * (1 hr / 60 min) * (1 revolution / 33.4375 in)]
// revolutions per second = speed ( mi / hr) * [(5280 ft / mi) * (12 in / ft) * (1 hr / 60 min) * (1 min / 60 sec) * (1 revolution / 33.4375 in)]
// [(5280 ft / mi) * (12 in / ft) * (1 hr / 60 min) * (1 min / 60 sec) * (1 revolution / 33.4375 in)] = 0.5263551402 ((rev * hr) / (mi*sec))
float scale_rpm = 5280 * 12 * 1/(float)60 * (1/(float)60) * 1/33.4375;
// revolutions per second = speed (mi / hr) * scale_rpm ((rev * hr) / (mi*sec))
// revolutions = revolutions per second * total seconds
float drum_revolutions = Speed.TestSpeed * scale_rpm *
Timing.set.Cylinder_1.tot_elapsed_secs;

if(CylinderFlags.Wheel1_Used && !CylinderFlags.Cyl_1_TestEnded) {

    // Update total revolutions
    // Rev = rpm (rev/min) * (1 min / 60 sec) * time(secs)
    TestStatus.Revolutions_1 = TestStatus.rpm_1 * (1/(float)60) *
    Timing.set.Cylinder_1.tot_elapsed_secs;

    // Update total cleat hits => rounds down, may be off by 1
    // Cleat hits = drum_revolutions * no of cleats (cleats / revolution)
    TestStatus.CleatHits_1 = UtilityMethods.RoundInt(drum_revolutions *
    CylinderValues.hmi.Cleats.Cleats_set_1);

    // Update total distance traveled
    // circumference (in) = pi * diameter (in)
    // Distance = revolutions * circumference of wheel (in) * (1 ft / 12 in) * (1 mi / 5280 ft)
    TestStatus.Distance_1 = TestStatus.Revolutions_1 * Pi() *
    CylinderValues.hmi.Wheel_1_Size_set * (1/(float)12) * (1/(float)5280);
}

if (CylinderFlags.Wheel2_Used && !CylinderFlags.Cyl_2_TestEnded) {

    // Update total revolutions
    // Rev = rpm (rev/min) * (1 min / 60 sec) * time(secs)
    TestStatus.Revolutions_2 = TestStatus.rpm_2 * (1/(float)60) *
updateTestData
Timing.set.Cylinder_2.totelapsed_secs;

// Update total cleat hits => rounds down, may be off by 1
// Cleat hits = drum_revolutions * no of cleats (cleats / revolution)
TestStatus.CleatHits_2 = UtilityMethods.RoundInt(drum_revolutions * CylinderValues.hmi.Cleats.Cleats_set_2);

// Update total distance traveled
// circumference (in) = pi * diameter (in)
// Distance = revolutions * circumference of wheel (in) * (1 ft / 12 in) *
// (1 mi / 5280 ft)
TestStatus.Distance_2 = TestStatus.Revolutions_2 * Pi() * CylinderValues.hmi.Wheel_2_Size_set * (1/(float)12) * (1/(float)5280);

}

//-----------------------------------------------------------------------------
//---------------------------------- Wheel 3 ----------------------------------
//-----------------------------------------------------------------------------

if (CylinderFlags.Wheel3_Used && !CylinderFlags.Cyl_3_TestEnded) {

// Update total revolutions
// Rev = rpm (rev/min) * (1 min / 60 sec) * time(secs)
TestStatus.Revolutions_3 = TestStatus.rpm_3 * (1/(float)60) * Timing.set.Cylinder_3.tot_elapsed_secs;

// Update total cleat hits => rounds, may be off by 1
// Cleat hits = drum_revolutions * no of cleats (cleats / revolution)
TestStatus.CleatHits_3 = UtilityMethods.RoundInt(drum_revolutions * CylinderValues.hmi.Cleats.Cleats_set_3);

// Update total distance traveled
// circumference (in) = pi * diameter (in)
// Distance = revolutions * circumference of wheel (in) * (1 ft / 12 in) *
// (1 mi / 5280 ft)
TestStatus.Distance_3 = TestStatus.Revolutions_3 * Pi() * CylinderValues.hmi.Wheel_3_Size_set * (1/(float)12) * (1/(float)5280);

}

updateTempTimeStrings

int UtilityMethods.RoundInt(float Arg1)

// Program created by Red Lion Controls

// this program accepts a Real argument and rounds it before converting to an integer

if(int(Arg1*10)%10 >= 5)
    return int(Arg1)+1;
else return int(Arg1);
CurrentDateString

cstring UtilityMethods.CurrentDateString(void)

// Program created by Red Lion Controls

// Converts Current Date into a string formatted as "MM/DD/YY" or "MM/DD/YYYY"

// Declare local variables
cstring mon, day, year;

// Convert current month into string
mon = IntToText(GetMonth(GetNow()), 10, 2);

// Convert current date into a string
day = IntToText(GetDate(GetNow()), 10, 2);

// Convert current year to string
// 2 digit year
year = IntToText(GetYear(GetNow()), 10, 2);

// 4 digit year
//year = IntToText(GetYear(GetNow()), 10, 4);

// Return formatted string
return mon + "/" + day + "/" + year;
CurrentTimeString12

cstring UtilityMethods.CurrentTimeString12(void)

// Program created by Red Lion Controls

/* 12 HOUR */

// Converts Current Time into a string formatted as "hh:mm:ss"

// declare local variables
cstring hour, min, sec, ap;

if(GetHour(GetNow()) > 12)
{
    // convert current hours into string
    hour = IntToText(GetHour(GetNow())-12, 10, 2);
    ap = "PM";
}
else
{
    // convert current hours into string
    hour = IntToText(GetHour(GetNow()), 10, 2);
    ap = "AM";
}

// convert current minutes to a string
min = IntToText(GetMin(GetNow()), 10, 2);

// convert current seconds to a string
sec = IntToText(GetSec(GetNow()), 10, 2);

// return formatted string
return hour + ":" + min + ":" + sec + " " + ap;
Appendix B: PLC Allocation List & Code
P0 - Start Up

""Start Up Program

"" Only called on startup or E-stop.
STEP 5
IF NOP
THEN LOAD V0
   TO OW0   'Output Word 0 (Soft Start Valve)
LOAD V2100
   TO TP1   'timer preset cyl 1 over cycle
   TO TP2   'timer preset cyl 2 over cycle
   TO TP3   'timer preset cyl 3 over cycle

"" Resets Test Outputs and Registers
STEP 10
IF NOP
THEN LOAD V0
   TO OW32   'Output Word 32 (Cylinders)
   TO OW33   'Output Word 33 (Cylinders)
   TO OW64   'PRESSURE REGULATOR ANALOG OUTPUT 1
   TO OW65   'PRESSURE REGULATOR ANALOG OUTPUT 2
   TO OW66   'PRESSURE REGULATOR ANALOG OUTPUT 3
   TO OW67   'VFD ANALOG OUTPUT
   TO FW0   'Flag Word 0 (Start/Stop Test Select)
   TO FW1   'Flag Word 1 (Cylinder Select)
   TO FW2   'Flag Word 2 (Overstroke Error)
   TO FW5   'Flag Word 5 (Overcycle Error)
   TO FW55  'Flag Word 55 (Retract Cylinder)
   TO R1    'Register 1, psi to cylinder 1
   TO R2    'Resister 2, psi to cylinder 2
   TO R3    'Register 3, psi to cylinder 3
   TO R4    'Register 4, vfd
RESET P1   'SEQUENCE PROGRAM CYLINDER 1
RESET P2   'SEQUENCE PROGRAM CYLINDER 2
RESET P3   'SEQUENCE PROGRAM CYLINDER 3
RESET P4   'VFD OUTPUT
RESET P5   'PRESSURE REGULATOR OQUPTUT
SET P63    'Error Reset Program

IF O0.0
THEN JMP TO 15
 OTHERWISE JMP TO 13

STEP 13
IF I0.0
THEN SET O0.0   'Soft start valve, turn air on

"" Wait for start test flag. If e-stop, hard reset.

Page 1
P0 - Start Up

STEP 15
IF N I0.1 'E-stop not active
THEN JMP TO 5 " If emergency stop, hard reset

IF F0.0 'Start Test Select
THEN SET P1 'SEQUENCE PROGRAM CYLINDER 1
SET P2 'SEQUENCE PROGRAM CYLINDER 2
SET P3 'SEQUENCE PROGRAM CYLINDER 3
SET P4 'VFD OUTPUT
SET P5 'PRESSURE REGULATOR OUTPUT
JMP TO 20 "Once start test is selected, wait for stop test select flag

"" Wait for stop test flag. If e-stop, hard reset.
STEP 20
IF N I0.1 'E-stop not active
THEN JMP TO 5 "If emergency stop, hard reset

IF F0.1 'Stop Test Select
THEN JMP TO 10 "If stop test, then reset all output values in soft reset
OTHRW
JMP TO 20 "If no stop test or e-stop, keep looking
P1 - CYL 1 Cycle

"" Cylinder 1 Cycle Program

"" If e-stop is good and cylinder 1 is selected then continue
STEP 5
IF I0.1 'E-stop not active
AND F1.0 'CYLINDER 1 SELECT
THEN NOP

"" If the reset button is not being pressed then continue
STEP 7
IF N I0.0 'Reset Pushbutton
THEN NOP

"" If cylinder 1 selected, retract flag is not set, and cylinder is currently retracted
"" then set cylinder to advance, set the over cycle timer to start
STEP 10
IF ( F1.0 'CYLINDER 1 SELECT
AND N F55.0 'Stop/Retract CYL 1
AND I0.7 'CYL 1 RETRACTED
THEN SET O32.0 'CYL 1 ADVANCE
SET T1 'timer cyl 1 over cycle

"" If timer expires and cylinder 1 is not at drum, then set overcycle flag
"" and reset the pressure valve to retract cylinder.
"" Once cylinder is at drum, continue
STEP 20
IF N T1 'timer cyl 1 over cycle
AND N I0.6 'CYL 1 AT DRUM
THEN RESET O32.0 'CYL 1 ADVANCE
RESET F1.0 'CYLINDER 1 SELECT
SET F5.0 'CYL 1 OVERCYCLE

IF I0.6 'CYL 1 AT DRUM
THEN NOP

"" If cylinder reaches overstroke, set overstroke flag, reset valve to retract cylinder,
"" reset the cylinder select flag to disable cylinder, and reset overcycle flag for cylinder.
"" If stop cylinder flag is received from HMI, reset the valve to retract cylinder, then reset flag.
STEP 22
IF I0.5 'CYL 1 OVERSTROKE
P1 - CYL 1 Cycle

THEN SET F2.0 'CYL 1 OVERSTROKE ERROR
RESET 032.0 'CYL 1 ADVANCE
RESET F1.0 'CYLINDER 1 SELECT
RESET F5.0 'CYL 1 OVERCYCLE
JMP TO 10

IF F55.0 'Stop/Retract CYL 1
THEN RESET 032.0 'CYL 1 ADVANCE
RESET F55.0 'Stop/Retract CYL 1
JMP TO 10

OTHWR
JMP TO 22
"" Cylinder 2 Cycle Program

"" If e-stop is good and cylinder 2 is selected then continue
STEP 5
  IF          I0.1     'E-stop not active
    AND       F1.1     'CYLINDER 2 SELECT
  THEN       NOP

"" If the reset button is not being pressed then continue
STEP 7
  IF          N     I0.0     'Reset Pushbutton
  THEN       NOP

"" If cylinder 2 selected, retract flag is not set, and cylinder is currently retracted
""
then set cylinder to advance, set the over cycle timer to start
STEP 10
  IF (          F1.1     'CYLINDER 2 SELECT
    AND     N     F55.1     'Stop/Retract CYL 2
    AND     I1.2     'CYL 2 RETRACTED
  THEN
    SET     O32.2     'CYL 2 ADVANCE
    SET     T2        'timer cyl 2 over cycle

"" If timer expires and cylinder 2 is not at drum, then set overcycle flag
"" and reset the pressure valve to retract cylinder.
"" Once cylinder is at drum, continue
STEP 20
  IF          N     T2        'timer cyl 2 over cycle
    AND     N     I1.1        'CYL 2 AT DRUM
  THEN
    RESET    O32.2     'CYL 2 ADVANCE
    RESET    F1.1     'CYLINDER 2 SELECT
    SET      F5.1     'CYL 2 OVERCYCLE

  IF          I1.1     'CYL 2 AT DRUM
  THEN       NOP

"" If cylinder reaches overstroke, set overstroke flag, reset valve to retract cylinder,
"" reset the cylinder select flag to disable cylinder, and reset overcycle flag for cylinder.
"" If stop cylinder flag is received from HMI, reset the valve to retract cylinder, then reset flag.
STEP 22
P2 - CYL 2 Cycle

IF I1.0 'CYL 2 OVERSTROKE
THEN SET F2.1 'CYL 2 OVERSTROKE ERROR
RESET O32.2 'CYL 2 ADVANCE
RESET F1.1 'CYLINDER 2 SELECT
RESET F5.1 'CYL 2 OVERCYCLE
JMP TO 10

IF F55.1 'Stop/Retract CYL 2
THEN RESET O32.2 'CYL 2 ADVANCE
RESET F55.1 'Stop/Retract CYL 2
JMP TO 10

OTHWR
JMP TO 22
P3 - CYL 3 Cycle

"" Cylinder 3 Cycle Program

"" If e-stop is good and cylinder 3 is selected then continue
STEP 5
IF I0.1 'E-stop not active
   AND F1.2 'CYLINDER 3 SELECT
THEN NOP

"" If the reset button is not being pressed then continue
STEP 7
IF N I0.0 'Reset Pushbutton
THEN NOP

"" If cylinder 3 selected, retract flag is not set, and cylinder is currently retracted
"" then set cylinder to advance, set the over cycle timer to start
STEP 10
IF ( F1.2 'CYLINDER 3 SELECT
   AND N F55.2 'Stop/Retract CYL 3
   AND I1.5 'CYL 3 RETRACTED
THEN SET O33.0 'CYL 3 ADVANCE
   SET T3 'timer cyl 3 over cycle

"" If timer expires and cylinder 2 is not at drum, then set overcycle flag
"" and reset the pressure valve to retract cylinder.
"" Once cylinder is at drum, continue
STEP 20
IF N T3 'timer cyl 3 over cycle
   AND N I1.4 'CYL 3 AT DRUM
THEN
   RESET O33.0 'CYL 3 ADVANCE
   RESET F1.2 'CYLINDER 3 SELECT
   SET F5.2 'CYL 3 OVERCYCLE
   IF I1.4 'CYL 3 AT DRUM
   THEN NOP

"" If cylinder reaches overstroke, set overstroke flag, reset valve to retract cylinder,
"" reset the cylinder select flag to disable cylinder, and reset overcycle flag for cylinder.
"" If stop cylinder flag is received from HMI, reset the valve to retract cylinder, then reset flag.
STEP 22
P3 - CYL 3 Cycle

IF I1.3 'CYL 3 OVERSTROKE THEN SET F2.2 'CYL 3 OVERSTROKE ERROR
RESET O33.0 'CYL 3 ADVANCE
RESET F1.2 'CYLINDER 3 SELECT
RESET F5.2 'CYL 3 OVERCYCLE
JMP TO 10

IF F55.2 'Stop/Retract CYL 3 THEN RESET O33.0 'CYL 3 ADVANCE
RESET F55.2 'Stop/Retract CYL 3
JMP TO 10

OTHERW
JMP TO 22
"" VFD Analog Output Program

"" EXOR (Exclusive OR) will return 1 if only one of the two values are '1'.
"" If cylinder is selected and at drum, (1 EXOR 1), will return 0
"" If cylinder is not selected and not at drum, (0 EXOR 0), will return 0
"" If cylinder is selected and not at drum, (1 EXOR 0), will return 1
"" Using N EXOR will allow test to continue when the selected cylinders reach the drum

"" If all active cylinders are at drum, then load VFD output with VFD register value

STEP 10

IF ( N ( F1.0  EXOR I0.6  )  'CYLINDER 1 SELECT
    AND  N ( F1.1  EXOR I1.1  )  'CYLINDER 2 SELECT
    AND  N ( F1.2  EXOR I1.4  )  'CYLINDER 3 SELECT
THEN

    SET  O0.1  'VFD Digital Enable
    LOAD  R4  'Register 4, vfd
    TO  OW67  'VFD ANALOG OUTPUT

    JMP TO 10
STEP 10

IF NOP THEN LOAD R1 TO OW64 'Register 1, psi to cylinder 1
LOAD R2 TO OW65 'Register 2, psi to cylinder 2
LOAD R3 TO OW66 'Register 3, psi to cylinder 3

IF NOP THEN JMP TO 10
STEP 0

IF I0.0 'Reset Pushbutton
THEN RESET E 'ERROR
LOAD V0
TO EW 'ERROR WORD
RESET P63 'Error Reset Program
SET P0 'Start Up Program
Appendix C: Connection Diagram
## Appendix D: Sample Tracker Data

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>Position (inches)</th>
<th>Time (sec)</th>
<th>Position (inches)</th>
<th>Time (sec)</th>
<th>Position (inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>2.392</td>
<td>0.504</td>
<td>2.361</td>
<td>1.008</td>
<td>2.388</td>
</tr>
<tr>
<td>0.004</td>
<td>2.405</td>
<td>0.508</td>
<td>2.359</td>
<td>1.013</td>
<td>2.389</td>
</tr>
<tr>
<td>0.008</td>
<td>2.387</td>
<td>0.513</td>
<td>2.355</td>
<td>1.017</td>
<td>2.394</td>
</tr>
<tr>
<td>0.013</td>
<td>2.388</td>
<td>0.517</td>
<td>2.354</td>
<td>1.021</td>
<td>2.393</td>
</tr>
<tr>
<td>0.017</td>
<td>2.402</td>
<td>0.521</td>
<td>2.362</td>
<td>1.025</td>
<td>2.394</td>
</tr>
<tr>
<td>0.021</td>
<td>2.405</td>
<td>0.525</td>
<td>2.365</td>
<td>1.029</td>
<td>2.390</td>
</tr>
<tr>
<td>0.025</td>
<td>2.387</td>
<td>0.529</td>
<td>2.353</td>
<td>1.033</td>
<td>2.394</td>
</tr>
<tr>
<td>0.029</td>
<td>2.388</td>
<td>0.533</td>
<td>2.355</td>
<td>1.038</td>
<td>2.407</td>
</tr>
<tr>
<td>0.033</td>
<td>2.396</td>
<td>0.538</td>
<td>2.362</td>
<td>1.042</td>
<td>2.404</td>
</tr>
<tr>
<td>0.038</td>
<td>2.395</td>
<td>0.542</td>
<td>2.359</td>
<td>1.046</td>
<td>2.392</td>
</tr>
<tr>
<td>0.042</td>
<td>2.391</td>
<td>0.546</td>
<td>2.356</td>
<td>1.050</td>
<td>2.394</td>
</tr>
<tr>
<td>0.046</td>
<td>2.391</td>
<td>0.550</td>
<td>2.356</td>
<td>1.054</td>
<td>2.399</td>
</tr>
<tr>
<td>0.050</td>
<td>2.392</td>
<td>0.554</td>
<td>2.357</td>
<td>1.058</td>
<td>2.410</td>
</tr>
<tr>
<td>0.054</td>
<td>2.395</td>
<td>0.558</td>
<td>2.358</td>
<td>1.063</td>
<td>2.399</td>
</tr>
<tr>
<td>0.058</td>
<td>2.393</td>
<td>0.563</td>
<td>2.358</td>
<td>1.067</td>
<td>2.391</td>
</tr>
<tr>
<td>0.063</td>
<td>2.391</td>
<td>0.567</td>
<td>2.351</td>
<td>1.071</td>
<td>2.387</td>
</tr>
<tr>
<td>0.067</td>
<td>2.387</td>
<td>0.571</td>
<td>2.382</td>
<td>1.075</td>
<td>2.388</td>
</tr>
<tr>
<td>0.071</td>
<td>2.390</td>
<td>0.575</td>
<td>2.381</td>
<td>1.079</td>
<td>2.387</td>
</tr>
<tr>
<td>0.075</td>
<td>2.392</td>
<td>0.579</td>
<td>2.365</td>
<td>1.083</td>
<td>2.392</td>
</tr>
<tr>
<td>0.079</td>
<td>2.390</td>
<td>0.583</td>
<td>2.357</td>
<td>1.088</td>
<td>2.386</td>
</tr>
<tr>
<td>0.083</td>
<td>2.386</td>
<td>0.588</td>
<td>2.363</td>
<td>1.092</td>
<td>2.395</td>
</tr>
<tr>
<td>0.088</td>
<td>2.394</td>
<td>0.592</td>
<td>2.380</td>
<td>1.096</td>
<td>2.394</td>
</tr>
</tbody>
</table>
## Appendix E: Budget

<table>
<thead>
<tr>
<th>ITEM NO.</th>
<th>VENDOR PART NUMBER</th>
<th>Length</th>
<th>DESCRIPTION</th>
<th>VENDOR</th>
<th>QTY</th>
<th>COST($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HTSQ214 (25 in)</td>
<td>25 Inch</td>
<td>2in x 2 in x 1/4in square tube</td>
<td>Metal Supermarkets</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>HTSQ214 (63 in)</td>
<td>63 Inch</td>
<td>2in x 2 in x 1/4in square tube</td>
<td>Metal Supermarkets</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>HTSQ214 (43 in)</td>
<td>43 Inch</td>
<td>2in x 2 in x 1/4in square tube</td>
<td>Metal Supermarkets</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>HTSQ214 (36 in)</td>
<td>36 inch</td>
<td>2in x 2 in x 1/4in square tube</td>
<td>Metal Supermarkets</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>T510 (33 in)</td>
<td>33 Inch</td>
<td>10” Structural Pipe (33 in)</td>
<td>metalsdepot</td>
<td>1</td>
<td>365.24</td>
</tr>
<tr>
<td>6</td>
<td>R1112</td>
<td>44 Inch</td>
<td>1.5” Diam Hot Rolled A-36 Steel</td>
<td>metalsdepot</td>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>7</td>
<td>P11</td>
<td>12 x 24 inch</td>
<td>1” Thick A36 Steel Plate(1x 2 ft)</td>
<td>metalsdepot</td>
<td>1</td>
<td>142.52</td>
</tr>
<tr>
<td>8</td>
<td>HTSQ214 (12 in)</td>
<td>12 Inch</td>
<td>2in x 2 in x 1/4in square tube</td>
<td>Metal Supermarkets</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>HTSQ214 (49 in)</td>
<td>49 Inch</td>
<td>2in x 2 in x 1/4in square tube</td>
<td>Metal Supermarkets</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>HTSQ214 (11 in)</td>
<td>11 Inch</td>
<td>2in x 2 in x 1/4in square tube</td>
<td>Metal Supermarkets</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>6494k19</td>
<td></td>
<td>Ball Bearing with Nickel Plated Cast Iron Housing</td>
<td>mcmastercarr</td>
<td>2</td>
<td>149.78</td>
</tr>
<tr>
<td>12</td>
<td>P114</td>
<td>2 ft x 4 ft</td>
<td>1/4 inch A36 Steel Plate</td>
<td>metalsdepot</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>P114</td>
<td>4 ft x 8 ft</td>
<td>1/4 inch A36 Steel Plate</td>
<td>metalsdepot</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>HTSQ214 (32 in)</td>
<td>32 Inch</td>
<td>2in x 2 in x 1/4in square tube</td>
<td>Metal Supermarkets</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></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>
</tr>
<tr>
<td>15</td>
<td>A12214</td>
<td>1 ft</td>
<td>2 in x 2 in x 1/4 in</td>
<td><a href="http://www.metalsdepot.com">www.metalsdepot.com</a></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>P138</td>
<td>1 ft x 1 ft</td>
<td>3/8 inch A36 Steel Plate</td>
<td><a href="http://www.metalsdepot.com">www.metalsdepot.com</a></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>HSH16</td>
<td>36&quot; x 47&quot;</td>
<td>16 GA. Hot Rolled Steel Sheet</td>
<td>Metal Supermarkets</td>
<td>2</td>
<td>72.5</td>
</tr>
<tr>
<td>18</td>
<td>HSH17</td>
<td>36&quot; x 67&quot;</td>
<td>16 GA. Hot Rolled Steel Sheet</td>
<td>Metal Supermarkets</td>
<td>2</td>
<td>75.41</td>
</tr>
<tr>
<td>19</td>
<td>25-5050</td>
<td>43 Inch</td>
<td>50mmx50mm (43 inch)</td>
<td>8020</td>
<td>4</td>
<td>150.82</td>
</tr>
<tr>
<td>19</td>
<td>25-5050</td>
<td>43 Inch</td>
<td>50mmx50mm (43 inch)</td>
<td>8020</td>
<td>4</td>
<td>150.82</td>
</tr>
<tr>
<td>20</td>
<td>25-5050</td>
<td>36 Inch</td>
<td>50mmx50mm (36 inch)</td>
<td>8020</td>
<td>4</td>
<td>73.22</td>
</tr>
<tr>
<td>21</td>
<td>25-505</td>
<td>63 Inch</td>
<td>50mmx50mm (63 inch)</td>
<td>8020</td>
<td>4</td>
<td>187.39</td>
</tr>
<tr>
<td>22</td>
<td>25-5050</td>
<td>32 Inch</td>
<td>50mmx50mm (32 inch)</td>
<td>8020</td>
<td>4</td>
<td>130.7</td>
</tr>
<tr>
<td>23</td>
<td>25-2550</td>
<td>32 Inch</td>
<td>25mmx50mm (32 inch)</td>
<td>8020</td>
<td>4</td>
<td>36.64</td>
</tr>
<tr>
<td>24</td>
<td>25-2550</td>
<td>33 Inch</td>
<td>25mmx50mm (33 inch)</td>
<td>8020</td>
<td>4</td>
<td>102.36</td>
</tr>
<tr>
<td>25</td>
<td>25-2086</td>
<td></td>
<td>25 series 4 hole aluminum hinge</td>
<td>8020</td>
<td>4</td>
<td>22.14</td>
</tr>
<tr>
<td>26</td>
<td>75-3515</td>
<td></td>
<td>Bolt Assembly(Hinge Hardware)</td>
<td>8020</td>
<td>16</td>
<td>8.64</td>
</tr>
<tr>
<td>27</td>
<td>40-2079</td>
<td></td>
<td>aluminum handle</td>
<td>8020</td>
<td>2</td>
<td>18.09</td>
</tr>
<tr>
<td>28</td>
<td>25-3392</td>
<td></td>
<td>Anchor</td>
<td>8020</td>
<td>100</td>
<td>270</td>
</tr>
<tr>
<td>29</td>
<td>65-2646</td>
<td>1615 mm x 1105 mm</td>
<td>Polycarbonate Panel 6 mm thick</td>
<td>8020</td>
<td>1</td>
<td>130.84</td>
</tr>
<tr>
<td>30</td>
<td>65-2646</td>
<td>1106</td>
<td>Polycarbonate</td>
<td>8020</td>
<td>2</td>
<td>143.68</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Model Number</td>
<td>Quantity</td>
<td>Unit Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>--------------</td>
<td>----------</td>
<td>-----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>65-2646 mm x 830 mm Panel 6 mm thick</td>
<td>855 mm x 780 mm Polycarbonate Panel 6 mm thick</td>
<td>2</td>
<td>109.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>65-2646 mm x 320 mm Panel 6 mm thick</td>
<td>830 mm x 780 mm Polycarbonate Panel 6 mm thick</td>
<td>4</td>
<td>109.68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Leveling Mounts</td>
<td>2531k151</td>
<td>4</td>
<td>44.39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>1.5 hp, 460V 3Φ, TEFC, Inverter Duty Motors and control</td>
<td>00152ET3E182T-S</td>
<td>1</td>
<td>427.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>5A, 3P, 400-500V, 24VDC, CP-T Power Supply</td>
<td>1SVR 427 054 R0000</td>
<td>1</td>
<td>316.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>25 mm x 640mm length Linear Guide Rail</td>
<td>G307K200 G3 Kadet 7&quot; HMI</td>
<td>1</td>
<td>865</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Safety Switch</td>
<td>TESZ1102/30</td>
<td>2</td>
<td>478.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>3/8&quot; x 3/8&quot; Keystock</td>
<td>47815</td>
<td>1</td>
<td>12.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Festo Standard Cylinder</td>
<td>DSBC-80-320-PPVA-N3</td>
<td>3</td>
<td>974.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Festo Prop-press reg</td>
<td>VPPE-3-1-1/8-10-010-E1</td>
<td>3</td>
<td>1479.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Festo connect cable</td>
<td>SIM-M12-5GD-2,5PU</td>
<td>3</td>
<td>56.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Festo 1-Way Control valve</td>
<td>GRLA-3/8-QS-6-D</td>
<td>6</td>
<td>164.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>Manifold Assembly</td>
<td>VtuS-25-SB2F8D-G38L-GC-G14T-4J+G</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>Linear Guide Rail</td>
<td>29181 25 mm x 640mm length</td>
<td>6</td>
<td>1131.51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>Bearing Block</td>
<td>5535462</td>
<td>6</td>
<td>785.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></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>
</tr>
<tr>
<td>47</td>
<td>159584</td>
<td>Filter Regulator</td>
<td>Allied Automation</td>
<td>1</td>
<td>103.53</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>Welding/Cutting/frames</td>
<td>Steel Frame</td>
<td>Steel Frame/drum welding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>8M90QD50.SF</td>
<td>QD Bushed, 8M Pitch Timing Pulley (Large)</td>
<td></td>
<td></td>
<td>245.56</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>8M30QD50.JA</td>
<td>QD Bushed, 8M Pitch Timing Pulley (Small)</td>
<td></td>
<td></td>
<td>56.4</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>SF.1-1/2</td>
<td>Bushing (Large)</td>
<td></td>
<td></td>
<td>52.09</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>JA.1-1/8</td>
<td>Bushing (Small)</td>
<td></td>
<td></td>
<td>12.75</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>17608M50</td>
<td>HTD Timing Belt</td>
<td></td>
<td></td>
<td>95.82</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>PU6MCLR100</td>
<td>100 ft 6mm OD Tubing</td>
<td>Automation direct</td>
<td></td>
<td>18.5</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>174373</td>
<td>Cylinder Foot Mount (HNC-80)</td>
<td>festo</td>
<td></td>
<td>97.85</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>90096A537</td>
<td>Zinc-Plated Steel Thread</td>
<td>McMasterCarr</td>
<td></td>
<td>48.11</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>1139571, 1139652,1140305</td>
<td>1 package of each M6, m12, M6 Hex nuts</td>
<td>Fastenal</td>
<td></td>
<td>52.11</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>1222447,56,28579</td>
<td>Fasteners, screws, Drill bit, paint,</td>
<td>Ace Hardware Store</td>
<td></td>
<td>52.93</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>185507</td>
<td>Mending Brace, Black Steel</td>
<td>True Value</td>
<td></td>
<td>38.81</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>9517K489,90322A156, 176,90322A156, 93286A049,93286A054, 92018A540, 92018A650</td>
<td>Washers, locknuts, steel tight bar, threaded rod, screws</td>
<td>McMasterCarr</td>
<td></td>
<td>70.98</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>Fk-M20X1</td>
<td>Rod Coupler</td>
<td>Allied Automation</td>
<td></td>
<td>262.2</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>A 6A 9M06012</td>
<td>Idler Roller</td>
<td>SDP/SI</td>
<td></td>
<td>42.93</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Details</td>
<td>Supplier</td>
<td>Quantity</td>
<td>Unit Cost</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>CEW-0102-R</td>
<td>Fixture arms weldment/material</td>
<td>Diversified Metal Specialties</td>
<td>3</td>
<td>795</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>25-2116, 65-2107</td>
<td>25 S rubber Panel Gasket 31 meter, Deadbolt Latch</td>
<td>Neff Engineering</td>
<td>33</td>
<td>68.02</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>110153985-1137188</td>
<td>nuts, DIN125, grease, foam tape, nylock, and HWH</td>
<td>Fastenal</td>
<td>50</td>
<td>68.09</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>7777T21, 7081K85</td>
<td>Power cable 20ft, 10ft plc power cable</td>
<td>McMasterCarr</td>
<td>2</td>
<td>81.92</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>CPS-AP-F</td>
<td>M8 Connector</td>
<td>Automation Direct</td>
<td>9</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>CD08-0A-070-A1</td>
<td>M8 Cable sensor</td>
<td>Automation Direct</td>
<td>9</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>CD08-0A-070-A1, CD12M-0B-070-A1</td>
<td>M8 Cable, M12 cable cord</td>
<td>Automation Direct</td>
<td>3</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>195704, 175487, 18779</td>
<td>Manifold Block, Festo Plug SEA-M12, Festo Plug SEA-GS-11-Duo</td>
<td>Allied Automation</td>
<td>16</td>
<td>576.62</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>1000096109</td>
<td>IPFW expenses</td>
<td>IPFW</td>
<td></td>
<td>332.52</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>HSH/060</td>
<td>Hot Rolled Sheet 16 Ga 0.060</td>
<td>Metal Supermarkets</td>
<td>1</td>
<td>32.28</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>HSH/060</td>
<td>Hot Rolled Sheet 16 Ga 0.060</td>
<td>Metal Supermarkets</td>
<td>1</td>
<td>16.23</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>91290A093, 90592A090</td>
<td>M4 long screw and hex nut</td>
<td>McMasterCarr</td>
<td>2</td>
<td>8.23</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>7561K43 - 7422K21</td>
<td>Steel Door, panel, DIN Rail, Cable</td>
<td>McMasterCarr</td>
<td>5</td>
<td>155.94</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>541330</td>
<td>Festo Conn. Cable</td>
<td>Allied Automation</td>
<td>3</td>
<td>56.4</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>55026 - 526170</td>
<td>Interlink Block</td>
<td>Allied</td>
<td>3</td>
<td>460.31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mainfold block, analogue module</td>
<td>Automation</td>
<td>Total</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------</td>
<td>------------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>16442.53</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>