Spring 2024

NFPA Fluid Power Vehicle Challenge

Rachel Wasik
rmw104@uakron.edu

Kimberly Griggy
The University of Akron, kmg206@uakron.edu

NFPA Fluid Powered Vehicle

Written by:

Nicholas Briggs

Kimberly Griggy

Hana Haddad

Ryan Souders

Rachel Wasik

Final Report 4600:471, Senior Design, Spring 2024

Faculty Advisor: Dr. Scott Sawyer & Dr. Saikishan Suryanarayana

Abstract

The Fluid Power Vehicle Challenge is an annual event that is sponsored by the National Fluid Power Association (NFPA). This capstone event requires each team to design and assemble a hydraulic bicycle. As a result, hydraulic fluid must be pumped from a reservoir, through a hydraulic circuit that generates energy to cause motion. This bicycle will then be tested in competition and ranked according to efficiency, endurance, and performance. The vehicle selected for this application was a tandem bike, to optimize ergonomics. The team will also participate in a competition at the end of the spring semester before graduation.

Table of Contents

Abstract 1
Introduction/Project Description 3
Technical Background 4
Design Requirements 7
Design 7
Hydraulic Circuit 7
Frame Design 15
CAD Modeling 19
Electronics 21
Programming: HMI 28
Programming: PLC 30
Programming: Modbus 42
Pneumatics 46
Codes and Standards 48
Societal, Safety, & Sustainability Considerations 50
Societal Considerations 50
Safety 50
Sustainability 51
Cost 51
Conclusion 52
References 53
Appendix 54
Appendix A: HMI Program 54
Appendix B: PLC Program 59
Appendix C: Modbus 75
Appendix D: CAD Models: 77
Appendix E: Hydraulic Circuit: 79

Introduction/Project Description

The objective of this project is to compete in the National Fluid Power Association's (NFPA) Fluid Power Vehicle Challenge. Our team has collaborated to design a vehicle that runs on fluid power, which will compete against other university teams to test the vehicle in various aspects. The University of Akron has competed in this challenge in years past and has used a bike design to convert human power into motion using hydraulic power. The vehicle will be examined in multiple ways during the competition, including a sprint race, an efficiency test, and an endurance race. The vehicle will also be evaluated for the safety of the hydraulic system designed, and proof of concept. Overall, this project will foster growth and understanding of the fluid power industry. This will, in turn, generate clean energy solutions that could be further explored. In designing our vehicle, the ultimate goals are to maximize efficiency at low speeds and maintain a manageable weight of the system. For this project, our team built on the past knowledge of the University of Akron teams that have participated over the years. This will aid our overall design process and allow us to further push into the development of new and innovative designs for the hydraulic systems and electronics featured in the vehicle.

Technical Background

Due to the technical nature of this project and the potential safety concerns, it is important to understand the theory behind calculating the performance of the bike. The NFPA has provided every team with access to information regarding making any necessary calculations (*Educational Webinars*). One of the first tasks is calculating the size of the drive motor, hydraulic lines, and pump. To accomplish this, first, the system pressure must be determined. This is based on the pressure of the weakest component in the system. Additionally, outside factors must be considered, such as the maximum incline and rolling resistance of the system. With an assumed road grade (x), the angle of the incline (α) can be calculated as follows:

$$\alpha = \tan^{-1}(x) \tag{1}$$

The rolling resistance (μ_R) depends on the material of the road, which can be found in charts online (For example: concrete is 0.002). Using an assumed load (L) in addition, allows the amount of force in the x and y direction to be calculated in equations 2 and 3 respectively.

$$F_x(lb) = \cos(\alpha) * L * \mu_R \tag{3}$$

$$F_y(lb) = \sin(\alpha) * L \tag{4}$$

Then the force required to move the bike up $\left(F_{up}\right)$ and down (F_{down}) the hill can be calculated as follows:

$$F_{up}(lb) = F_x + F_y \tag{5}$$

$$F_{down}(lb) = F_y - F_x \tag{6}$$

The torque (T) must also be calculated using our maximum Force $\left(F_{up}\right)$ and radius (r) of the wheels, as shown in equation 7.

$$T(lb * in) = F_{up} * r \tag{7}$$

With an assumed pressure (P), the displacement of the hydraulic motor can be calculated (CIR) can be calculated, as shown in Equation 8. It is important to note that this value would assume 100% efficiency. Typically, hydraulic gears are closer to 90% efficiency. As a result, the displacement value should be modified $\left(CIR_{modify}\right)$ to compensate for any inefficiencies within the system. That can be calculated by dividing CIR by the assumed efficiency (η).This gives us the specifications for our pump size.

$$CIR\ \left(\frac{in^3}{revolution}\right) = \frac{2\pi * T}{P} \tag{8}$$

$$CIR_{modify} = \frac{CIR}{\eta} \tag{9}$$

Next, the wheel rotation per minute $(Wheel_{RPM})$ and the system's volumetric flow rate (Q_{GPM}) can be calculated, by considering the velocity of the bike (v) and the diameter of the wheel (d) (*Educational Webinars*).

$$Wheel_{RPM} = \frac{336 * v(mph)}{d(in)} \tag{10}$$

$$Q_{GPM} = \frac{CIR * Wheel_{RPM}}{231} \tag{11}$$

Next, the lines must be sized. This is a critical step in our process as undersized lines create excess heat and pressure drops. First, the net area must be calculated by using a set velocity. This allows us to calculate the diameter of the hose (*Educational Webinars*).

$$A_{net}(in^2) = \frac{0.32 * Q_{GPM}}{v_{oil}\ (ft/sec)} \tag{12}$$

$$d_{hose} = \sqrt{\frac{A}{0.7854}} \tag{13}$$

This project also requires teams to determine the useable amount of oil. If all of the oil in the accumulator is used, the usable amount of oil (oil_{usable}) can be determined by using the gas laws and the pressure of a fully charged $(p_{\max})$ and empty accumulator $\left(p_{precharge}\right)$.

$$p_1 * V_1 = p_2 * V_2 \quad (14)$$

$$oil_{usable}(in^3) = V_2 - V_1 \quad (15)$$

The absolute pressure value of p_1 and p_2 can be calculated by adding 14.7 psi to our pressure values, as shown below. Additionally, let V_1 represent the volume of the accumulator in cubic inches and let V_2 represent the volume of the gas.

$$p_1(psi) = p_{precharge} + 14.7 \quad (16)$$

$$p_2(psi) = p_{max} + 14.7 \quad (17)$$

If the bike is operating between the minimum and maximum pressures, the following equations can be used to depict the usable amount of oil. Let p_1 represent the pre-charge pressure, p_2 be the minimum working pressure, and p_3 be the maximum working pressure. Additionally, let V_1 represent the volume of gas in the accumulator, V_2 be the volume of gas at minimum working pressure, and V_3 be the volume of gas at maximum working pressure.

$$p_1 < p_2 < p_3 \quad (18)$$

$$p_1(psi) = p_1 + 14.7 \quad (19)$$

$$p_2(psi) = p_2 + 14.7 \quad (20)$$

$$p_3(psi) = p_3 + 14.7 \quad (21)$$

$$p_1 * V_1 = p_2 * V_2 = p_3 * V_3 \quad (22)$$

$$oil_{usable}(in^3) = V_3 - V_2 \quad (23)$$

It is important to note if the accumulator is charged too quickly and turned off, turning the system back on a minute later can result in less gas pressure than intended (*Educational Webinars*).

Design Requirements

Our vehicle must include two pressure indicators, an energy storage device, hydraulic propulsion (from the pump and motor), and must weigh under 210 pounds. Additionally, various safety factors must be taken into consideration. In any part of the system, pressure cannot exceed 3000 psi. Additionally, it is important to include an independent braking system, as well as a fail-safe braking condition. These brakes must be able to stop the vehicle against the full charge of the accumulator.

Design

Hydraulic Circuit

To improve the performance of the hydraulic circuit, it was important to understand the performance of the previously used hydraulic circuit. Their circuit is provided below for reference in Figure 1.

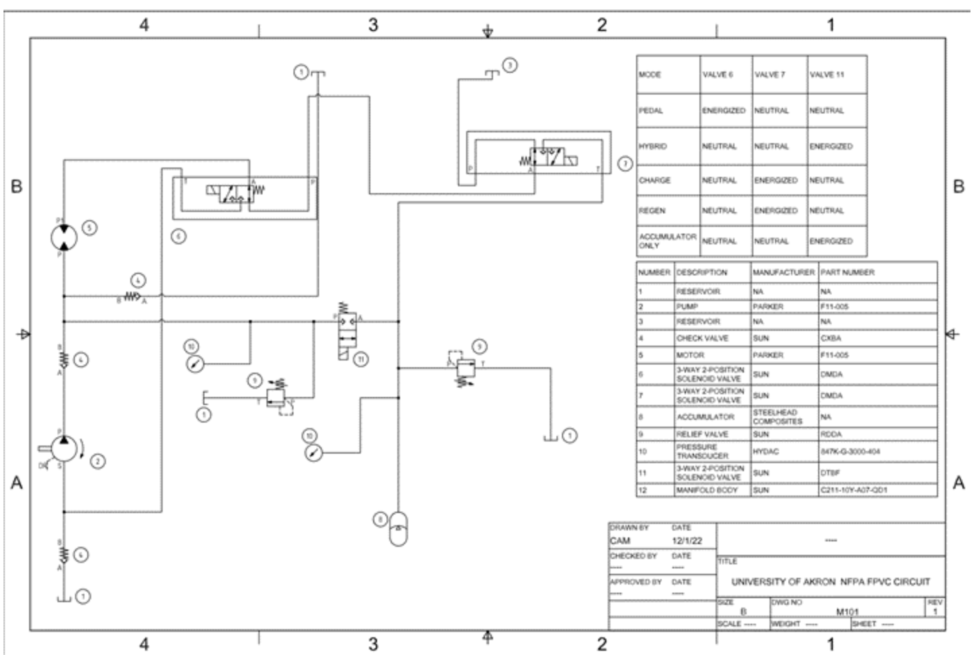


Figure 1: 2022-2023 Fluid Power Vehicle team's hydraulic circuit.

Through this process, several design iterations were created. One notable idea includes combining the two three-way switches into one four-way switch. By utilizing a four-way switch that has one input and three outputs, a switch can be eliminated. This would substantially reduce the complexity of the circuit. However, upon researching this idea it was discovered that four-way switches do not operate in this manner.

Another design iteration focused on the quality of the components used in the circuit. After reviewing the three-way switch datasheet, it was found that two of the switches have a leakage rating of 2 cubic inches per minute (in^3/min). This means that 2 in^3/min of hydraulic fluid per minute will still leak through despite being blocked off. This is a significant issue, as when the accumulator is fully charged to 3000 psi, the leakage of the three-way solenoids causes it to lose charge regardless of being used very quickly.

As a result, the circuit was revised to split the three-way switches into two two-way switches. The purpose of doing so was to eliminate the usage of spool style switches, which as previously discussed have high leakage, in favor of using poppet style switches which have significantly less leakage. Even with the minor addition of complexity to the manifold design, it was important to prioritize minimizing the overall fluid leakage. Additionally, a line with a check valve was introduced to bypass the pump. In the event the motor is turning over while the pump is not, this addition helps to prevent cavitation in the pump. All of this is reflected in the first circuit revision, which is pictured below in Figure 2.

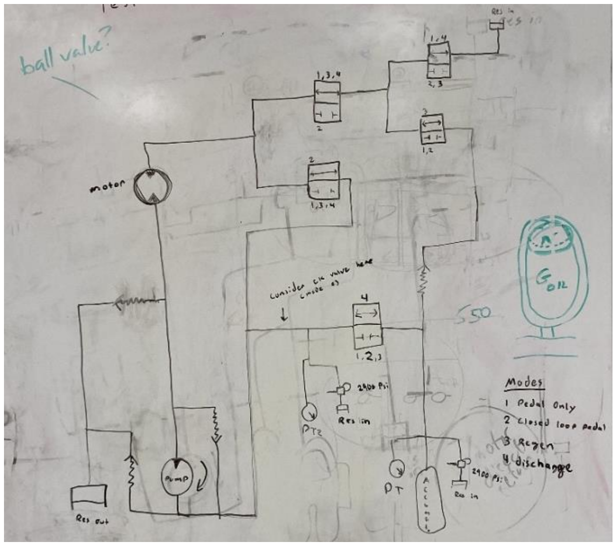


Figure 2: First circuit revision

After discussing the bike's operation modes in more detail, the circuit was revised a second time. Another check valve was added to help prevent leakage from the accumulator when fully charged. With the design finalized, the manifold was built using Sun Hydraulics' QuickDesign software. The diagram of the initially created manifold block is provided in Figure 3.

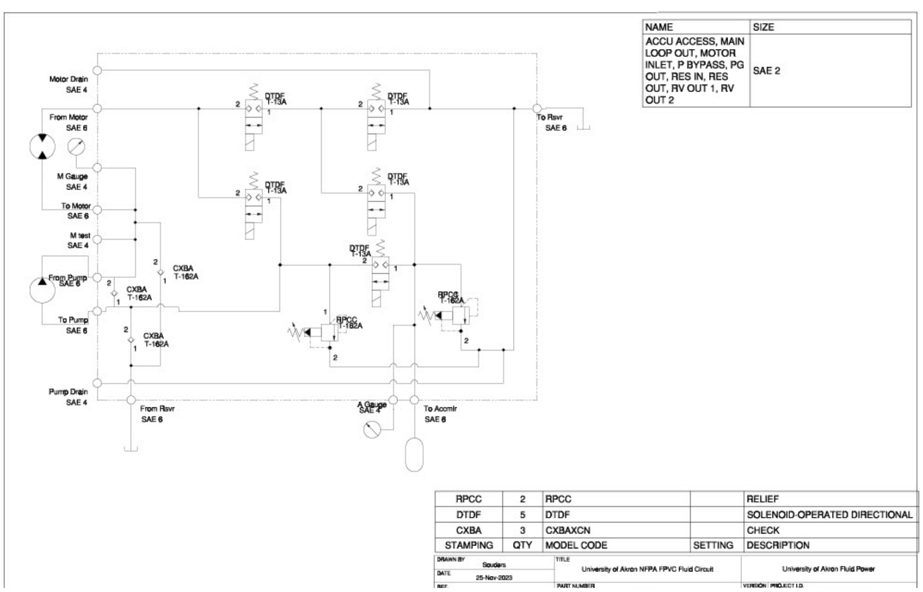


Figure 3: Initial manifold block schematic

With the general manifold design now realized, we scheduled a meeting with Ernie Parker to have him review both our fluid circuit and our manifold design. He was instrumental from here on out in making meaningful changes to our circuit and manifold. He recommended that we remove the upper right two switches as he felt what they do could be implemented into what the other switches do. He also recommended that we move one of the relief valves to help protect the pump. The schematic of these revisions is shown in Figure 4.

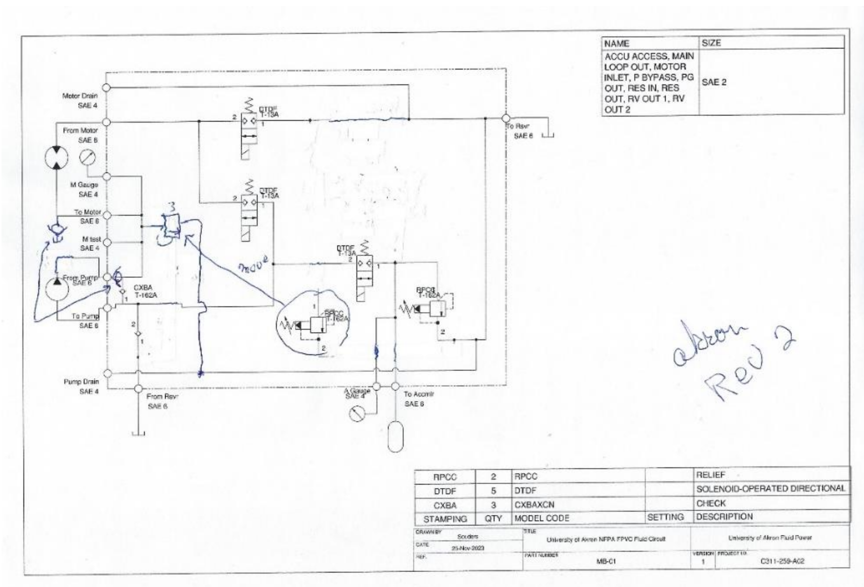


Figure 4: Manifold schematic revisions after discussions with Ernie Parker

From here, we deliberated about the pros and cons of removing both switches that Ernie Parker recommended and ultimately decided to add back one of them to ideally help with charging the accumulator. We folded the function of the extra reservoir output with a check valve in front of the motor into the manifold, as we didn’t want to have any external junctions or increase the complexity of the hard lining that is planned. After several back-and-forth conversions with Ernie, the final design was created, as show in Figure 5.

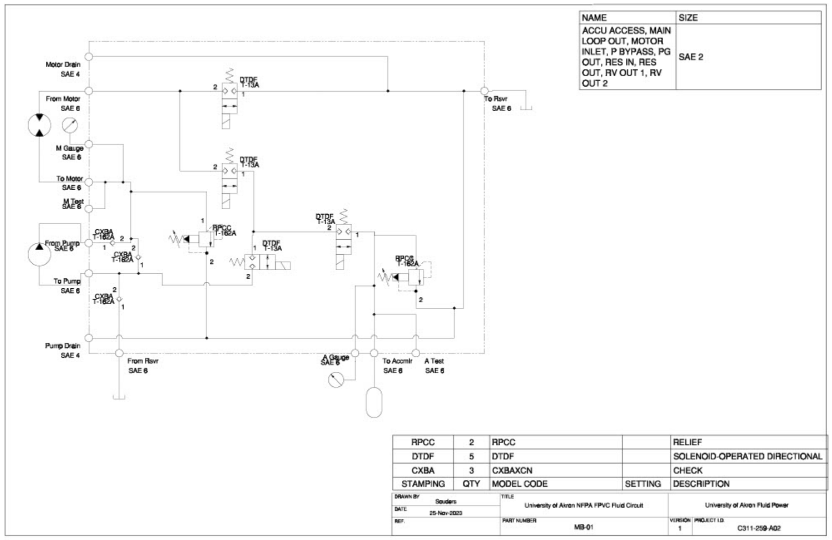


Figure 5: Final manifold schematic

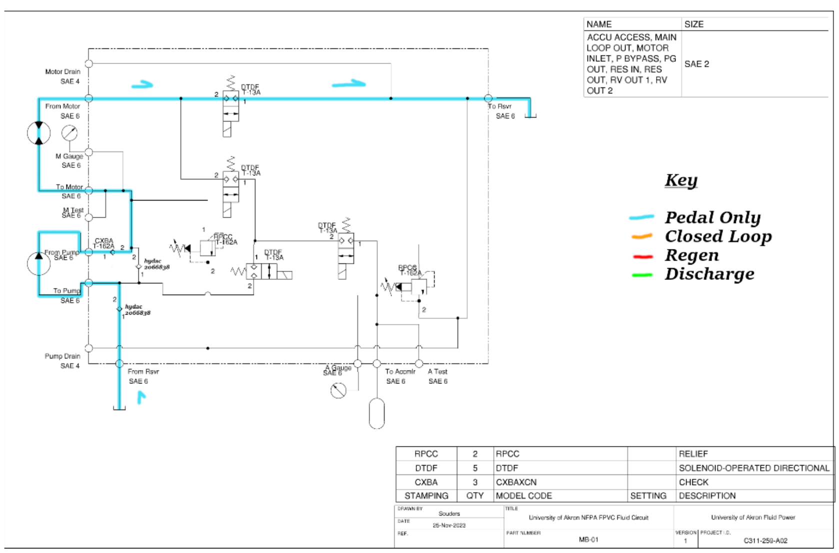


Figure 6: Manifold schematic with pedal only mode highlighted

Pedal-only mode is the main mode of operation for the bike. When you turn the pedals, it will turn the pump. This in turn moves fluid from the reservoir, through the pump and to the motor. As fluid moves through the motor, it will spin an output shaft that is directly connected to the drive wheel on the bike. When the fluid passes through the motor, it will be deposited back into the reservoir. If the rider chooses to stop pedaling while the bike is moving, there is a bypass in place that allows the fluid to move from the reservoir to the motor with little resistance. This helps prevent cavitation from occurring. There is also a relief valve in place so if too much pressure builds up while pedaling, fluid can be released back into the reservoir.

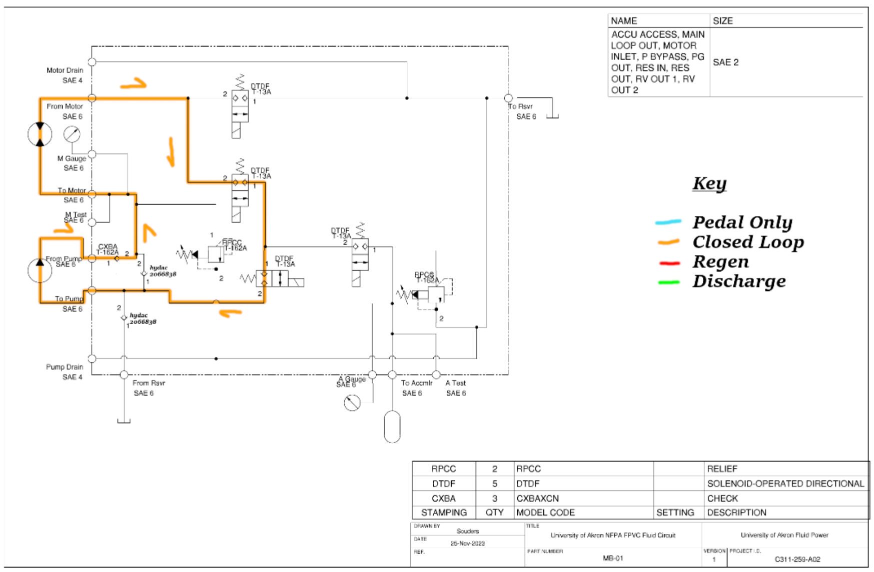


Figure 7: Manifold schematic with closed loop mode highlighted.

Closed Loop mode is intended to mainly be used during the endurance race and sprint race post initial acceleration. The point of this mode is to reduce losses from cycling fluid through the reservoir, thereby making the bike easier to ride for the rider. Do note that fluid is required to have already been cycled into the system for this mode to work. Fluid will be cycled from the pump to the motor which will then be cycled back to the pump. Similarly, to the last mode, if the rider chooses to stop pedaling while the bike is moving, there is a bypass in place that allows the fluid to move from the reservoir to the motor with little resistance. This helps prevent cavitation from occurring. There is also a relief valve in place so if too much pressure builds up while pedaling, fluid can be released back into the reservoir.

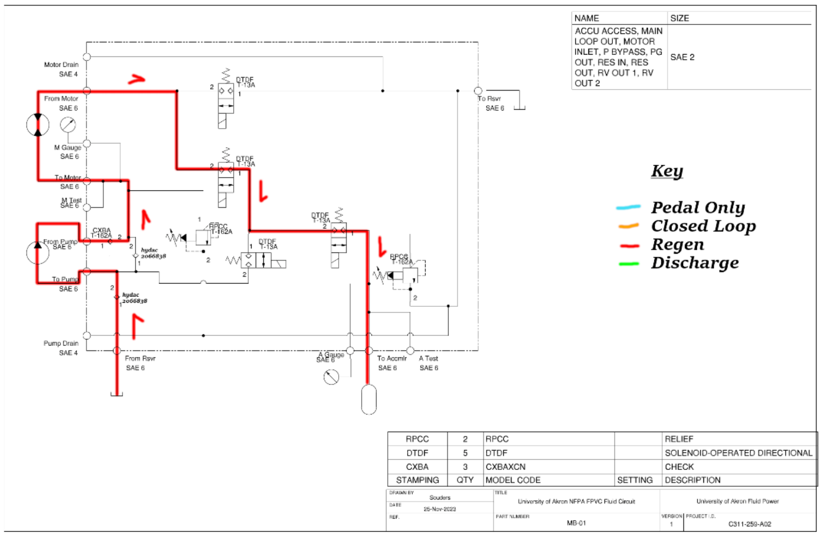


Figure 8: Manifold schematic with regen mode highlighted

Regen mode is used to pressurize the accumulator, which can then later be used to power the bike, solely on the accumulated pressure. In regen mode, as the rider pedals, it'll spin the pump and push fluid through the motor and pump. This fluid then goes down and into the accumulator, pressurizing it up to a maximum of 3000 psi. If the accumulator reaches above 3000 psi. It will open a relief valve and send fluid back to the reservoir, preventing any over-pressuring from occurring.

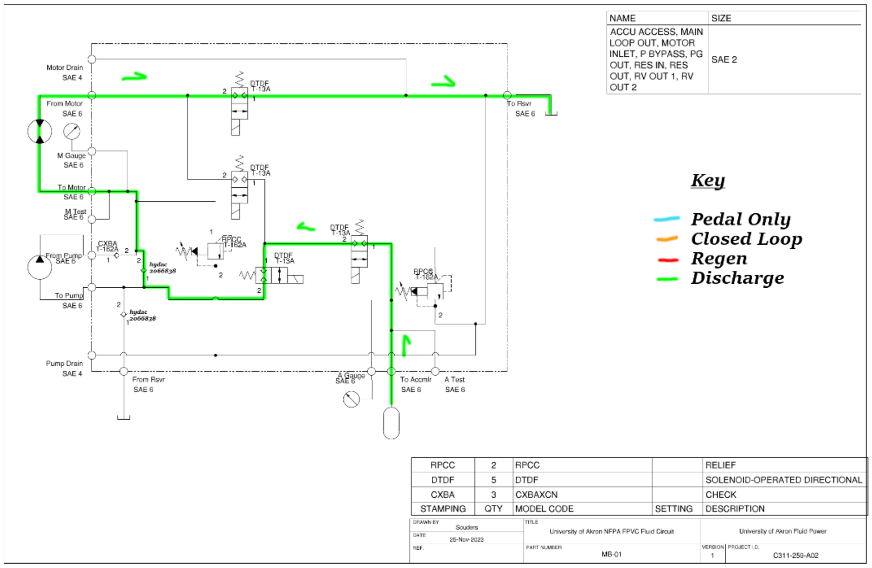


Figure 9: Manifold schematic with discharge mode highlighted

The last mode is referred to as Discharge Mode. In discharge mode, the pressurized fluid in the accumulator is released and used to power the motor, in turn pushing the vehicle forward. The pressurized fluid will be pushed through the pump bypass, as it provides the least amount of resistance. Once the fluid goes through the motor, it will return to the reservoir to be stored.

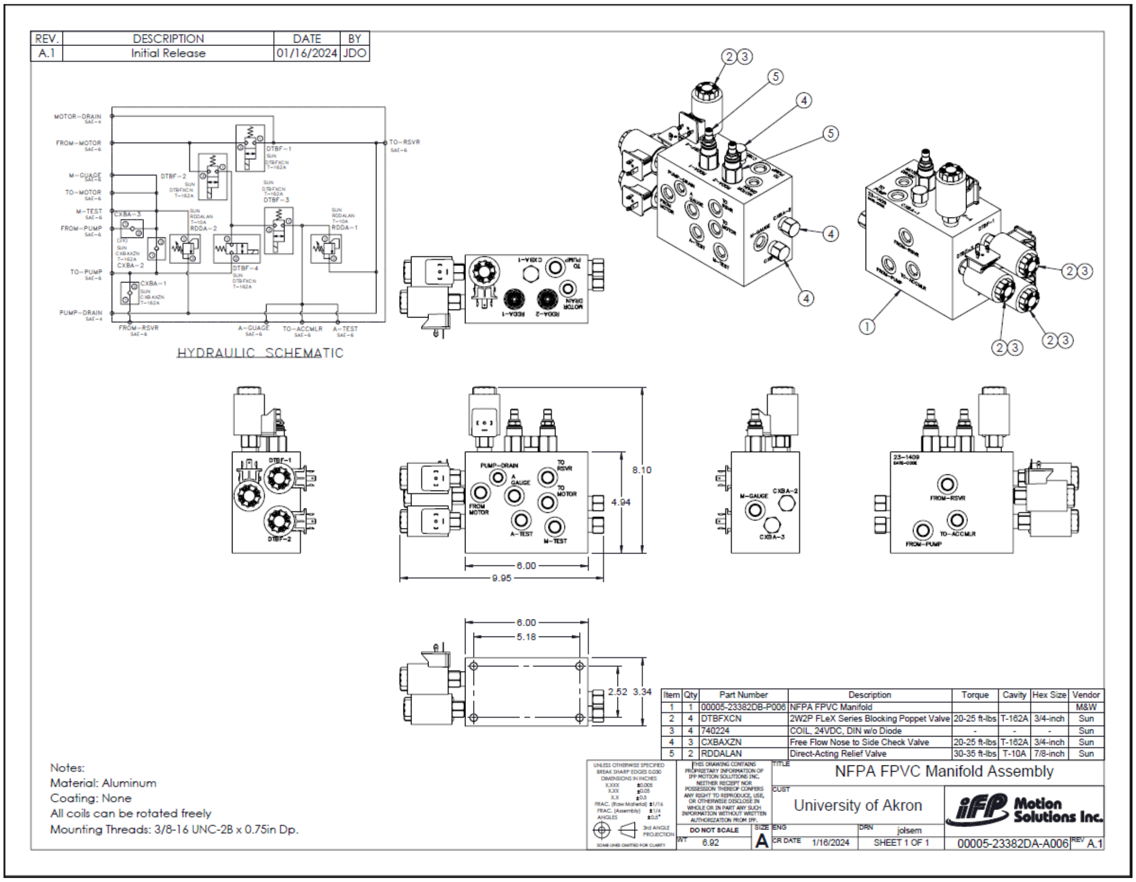


Figure 10: Manifold drawings provided by iFP

The manifold designed for the hydraulic circuit is pictured above in Figure 10. Despite a minor increase in the complexity of the circuit, the overall size of the manifold was reduced from the previous year's team by roughly 4 cubic inches. This is helpful, as its lightweight and compact design helped it to fit neatly on the bike. The manifold features ports for all component to be attached, as well as ports for pressure sensors, and the solenoids discussed previously in the design process.

Frame Design

The type of vehicle selected for this project was a tandem bike. Previously, teams have used a standard touring bike for its lightweight build and ergonomic handles. Also, the team preferred a bicycle rather than a tricycle because more wheels would add more rolling resistance. Although tricycles have more space to mount equipment, they are heavier and less agile. Due to these considerations, the team concluded that a two-wheeled frame would be more optimal for this project.

The design team chose to move forward with a tandem bike to eliminate some design problems with the previous team’s vehicle. First, the previous team had issues with the rear tire going flat. This was due to the wheel selection and weight distribution of the vehicle. The tires that were selected were slim race tires that could not support an excessive amount of weight, such as the hydraulic system that was implemented. Most of the hydraulic components were installed via an aluminum sub-frame, which encased the rear wheel and concentrated most of the bare vehicle weight directly onto the rear wheel. In addition to the weight issues, this sub-frame made it difficult to remove the rear wheel and impeded the team’s ability to replace the flat tire in last year’s competition.

Selecting a tandem bike provided a longer frame which allowed all equipment to be mounted between the two wheels. This distributes the weight more evenly and allows access to the tires for maintenance. Additionally, the extra space allows for all hydraulic components to be placed behind the rider, rather than between the rider’s legs. Previously, teams had mounted the accumulator, reservoir, and hard lining next to the rider which affected their ability to pedal and posed safety hazards. The tandem frame makes the vehicle much safer because the rider is now distanced from any high-pressure components that could injure them in the event of a failure. By placing the equipment behind the rider, they are also able to pedal without obstruction.

Although one may think a tandem bike may be difficult to maneuver, it was stable when being used during the competition. This is because the center of gravity was shifted to the center of the bike. Also, components were kept low to the ground, which lowered the center of gravity and prevented the bike from tipping over. The wider wheelbase made it slightly more difficult to make sharper turns, however, this did not affect the rider on the competition course. The vehicle had difficulty accelerating during the competition, but this was partially due to wind conditions. Once momentum was built up, the vehicle to able to sustain its motion.

The final design for the vehicle is shown below as iteration 3. This design was chosen after we purchased our bike, which is a Burley Zydeco tandem bike. Once the team had a physical frame, we implemented the design ideas from iteration 2 and placed them on the actual frame layout. Our bike does not have a diagonal bar in the center as previously thought in the earlier iterations, and therefore we were able to utilize more frame space to fit our needs. In the final design, the reservoir sat on a custom-built aluminum platform over the rear tire, distributing the weight onto the bike frame, not the axle. The manifold sits in front of the reservoir, on top of the frame, on a custom plate that is welded to the frame itself. The pump and motor sit within a steel plate that is welded vertically to the bottom and sides of the frame. Finally, the accumulator hangs from the top of the frame with hose clamps, just above the pump and motor. All the electronics and pneumatics are placed similarly as they are in iteration 2.

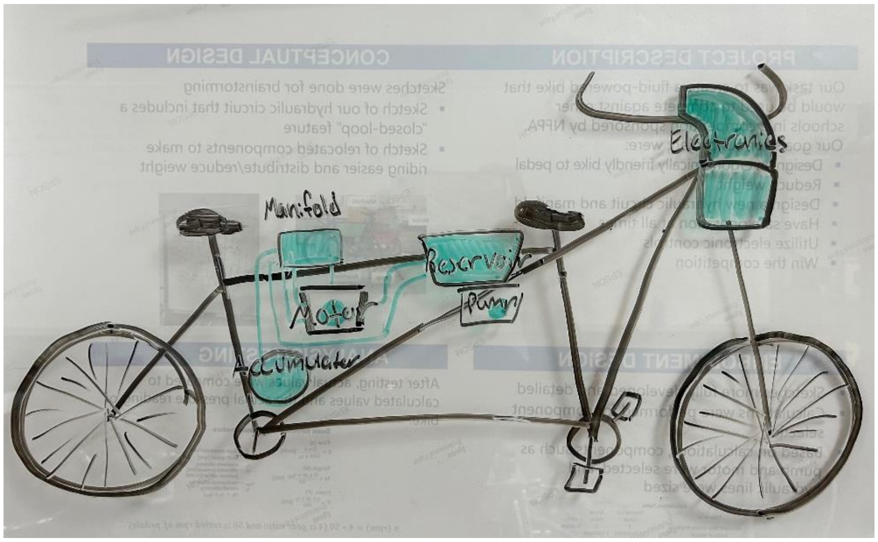


Figure 12: Vehicle frame design, iteration 1

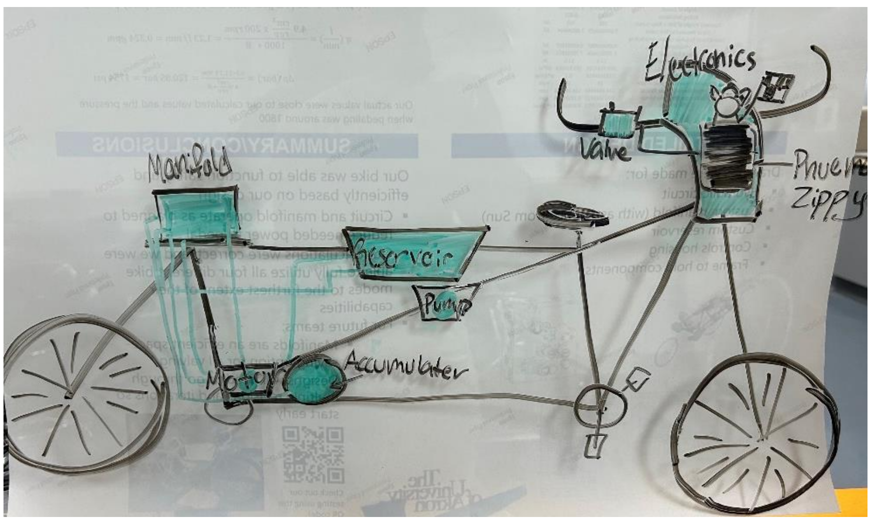


Figure 11: Vehicle frame design, iteration 2

<u>CAD Modeling</u>

Modeling the entire vehicle was crucial, as it allowed the team to make accurate design decisions and order the proper amount of building materials. Using Solidworks, the bike frame was modeled, including the gears, pedals, seats, and tires. This allows the team to see all the constraints within the design. Next, the components used on our vehicle were modeled or downloaded. This included the accumulator, pump and motor, and our manifold. Utilizing these models, plates were then designed and modeled to mount the pump and motor and plasma-cut out of 1008 steel.

Figure 13: Left-side view of the bike assembly in SolidWorks, including the electronics boxes, pneumatic box, manifold, accumulator, pump and motor, reservoir, and

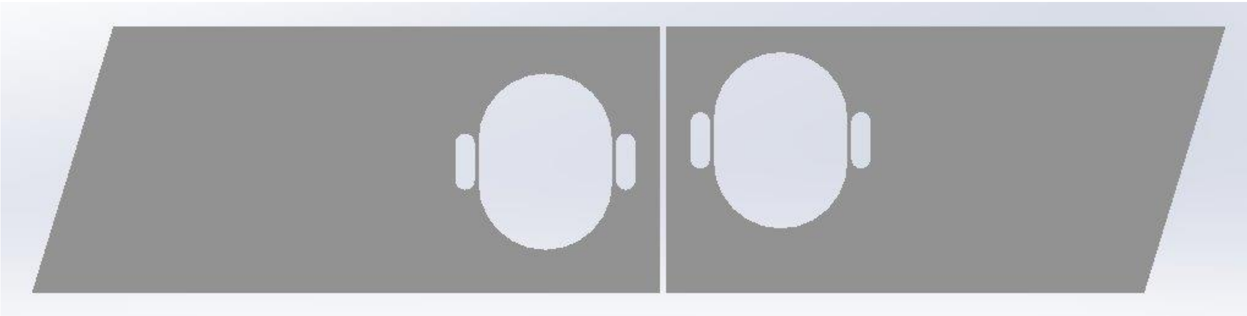


Figure 14: Pump and motor plates, designed in SolidWorks

The reservoir was designed for the vehicle based on a 2-gallon capacity of fluid within the system. It is 2 chambered, with an inlet and outlet located on the front. Differing from last year's design, it was elected to place the infill on the top of the reservoir. This leads to a decreased risk of cracking the reservoir during shipping, as there are no weak points located on the sides of the tank. Additionally, a base was included on the reservoir, that will match the frame that it is placed on. This allows the team to bolt the tank directly onto the plate and secure it properly. The reservoir was sent to SchmidtPro for manufacturing, as it was too large for the available printers on the University of Akron (UA) campus. The filament chosen for the reservoir was PETG or Polyethylene Terephthalate Glycol. PETG was chosen for its chemical resistance, thermal stability, as well as its impact resistance. Once the final reservoir arrived, it was sealed with a fuel tank sealer. This was done to prevent any leaks that may become present due to the nature of additive manufacturing.

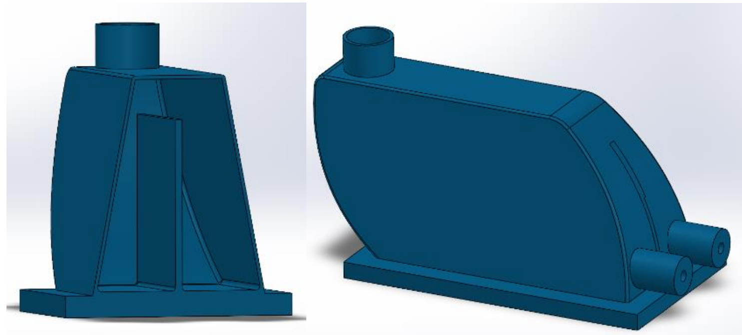


Figure 15: SolidWorks model of the reservoir, including a cutaway view and an isometric view

Numerous boxes or holders were required to be modeled for our design. These included the battery holder, the frame for the HMI screen, the box for the PLC, the pneumatics box, and a chain guard located over the exposed pinch point of the pump. These boxes attach to the frame via hose clamps and were designed with slide-out dovetail lids for easy access. All boxes were sent to the UA 3D printing lab for manufacturing. The battery holder houses the

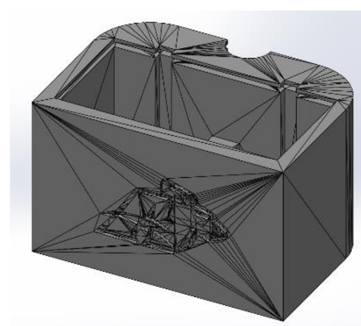


Figure 17: SolidWorks model of the battery box

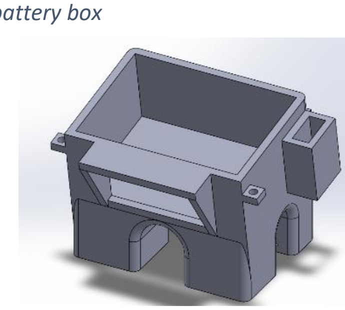


Figure 16: SolidWorks model of the HMI screen box

vehicle's two 24V batteries that power the electronics. Channels for hose clamps or zip ties are placed around a semi-circular cutout of this box, allowing it to be easily attached to the frame of the bike. This box is also equipped with a sliding lid to protect the batteries from the elements. Similarly, the HMI box houses the touch screen connected to the electronics. Designed into the box are slots for the placement of the touchscreen on top of the handlebars. This allowed the screen to be adjusted to the rider's position. Lastly, the chain guard was designed to be attached using the bolts connected to the pump. Channels surrounding the center hole of the guard are equipped with washers, allowing it to be secured into place using the nuts on the end of the pump's bolts.

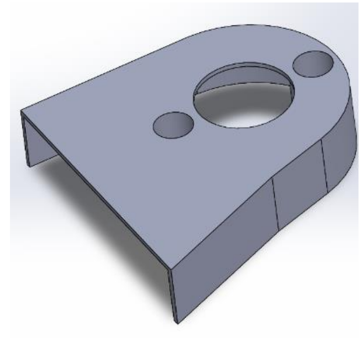


Figure 18: SolidWorks model of the chain guard

Electronics

The electronics can be described by two main components, the programmable logic controller (PLC) and the human-machine interface (HMI). A PLC allows the team to read data from pressure sensors and send signals to solenoid valves used to control the operating mode of the bike. An HMI provides the user a way to see the data analyzed by the PLC and serves to let the user manually choose the operating mode of the bike. For this project, an Exor HMI (ex705) was utilized. Regarding the PLC, several necessary qualities were considered. Due to the hydraulic circuit design, the PLC needs to allow for a minimum of four discrete output ports for the solenoids, two analog input ports for the pressure transducers, and to utilize ethernet communication. This led to the selection of IFM's ecomatController (CR710S). Without a communication protocol, these two systems remain independent of one another. While there are several methods of establishing dependence between the two components, the Modbus TCP communication protocol was selected.

The next step in the design process was taking inventory of the capabilities and necessities of the two electronic devices, to gather all the knowledge necessary to complete a wiring diagram. Shown in Figure 19 is a diagram of the discrete outputs from IFM's PLC. The first four ports are used for the four solenoids needed to operate the hydraulic circuit. This leaves the team with leftover ports that have been repurposed into a backup port for each corresponding solenoid. For example, planning on programming "Solenoid 1" and "Solenoid 1 Back Up" in the same way, allows the team to move the wiring quickly without editing the program. This can prove useful in the event the "Solenoid 1" port stops working in the middle of the competition. By preparing for the worst, the bike will be able to operate at its best.

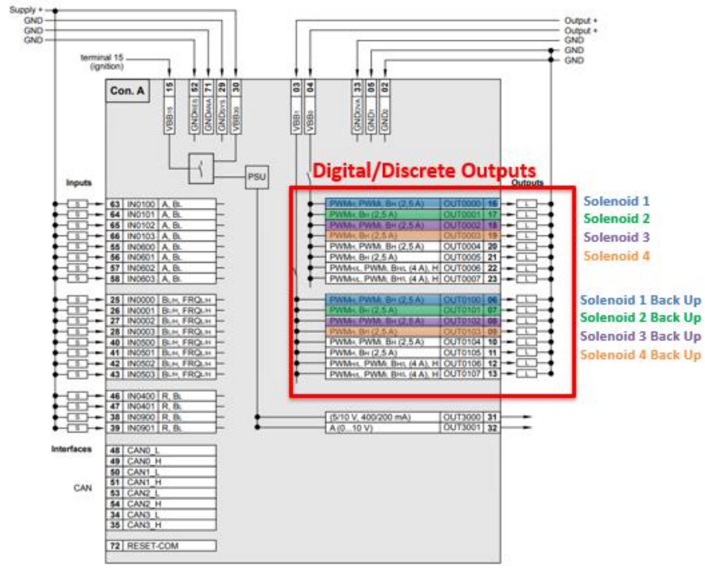


Figure 19: Discrete outputs from IFM's PLC

The analog inputs for the two pressure sensors are shown in Figure 20. The sensors will also be programmed with backup ports. By verifying the layout of the PLC, it was determined that we needed a power distribution block, DIN rail, end brackets, and splice connectors.

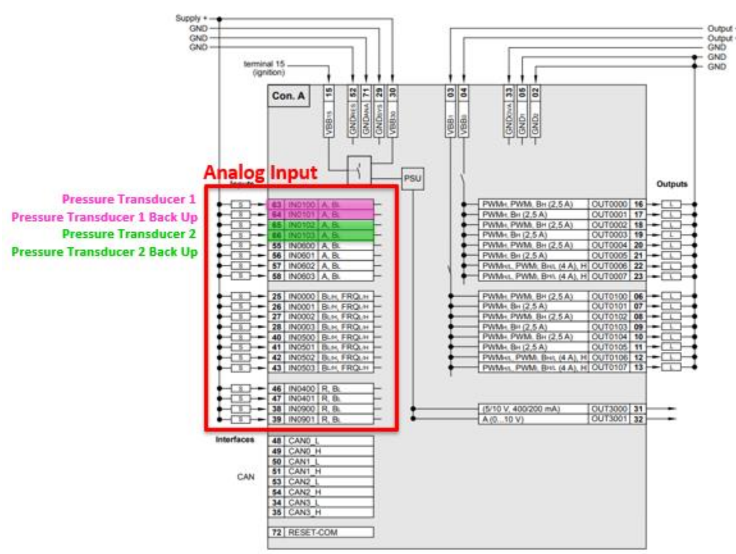


Figure 20: Analog inputs for the two pressure sensors in the PLC

The power distribution block will allow the electrical circuit to distribute from a single input source to the remaining devices in the branch circuit. Based on the number of devices shown in the two prior diagrams, the block must include 7 ports that go to power (24V) and 22 ports that go to ground (GND). Connecting multiple blocks can waste space and wiring, so a block was chosen that contained 24 ports for each power and ground (Phoenix Contact 2903800). This requires a DIN rail (Phoenix Contact 1207640) and corresponding end brackets (Phoenix Contact 3022276) to mount the power distribution block. It is important to note that the solenoids and pressure transducers have removable connectors with built-in leads. A splice connector (Wago 221-412/996-010) is needed to join the device to the PLC input/output (I/O) port's built-in leads.

Since the PLC's I/O ports are accounted for, we can now take an expanded view of the PLC itself, as shown in Figure 21. On the right side of the figure is a communication port referred to as ETH0. This communication port allows the user to connect the PLC to the HMI.

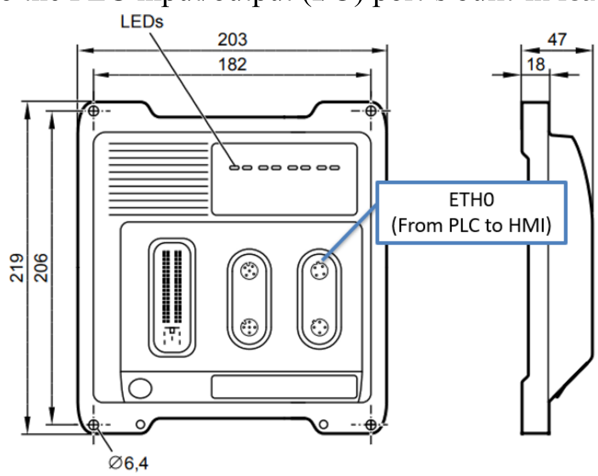


Figure 21: Expanded view of the PLC

Shown in Figure 22, is an overall view of the PLC's power distribution. The main attribute to note is that there are 2 amp (A) and 15 A fuses (Eaton-Bussmann BK/ATM-2 and BK/ATM-15). By utilizing fuses, the equipment will be protected from any overcurrent

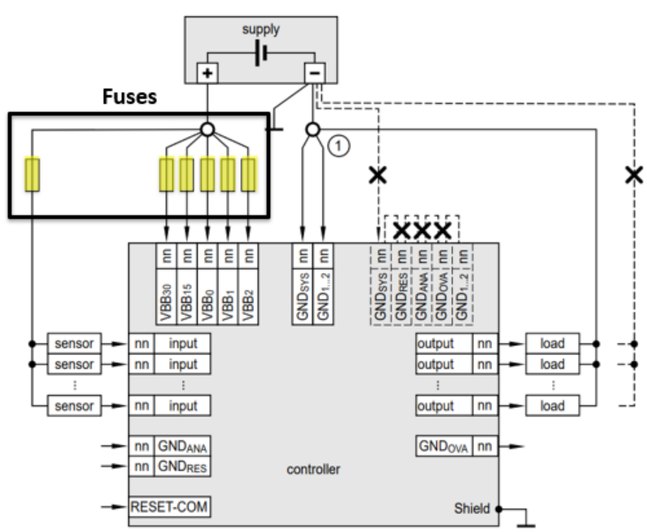


Figure 22: PLC power distribution diagram

flow. Additionally, a fuse terminal block (Phoenix Contact 2903800) must be utilized to distribute the battery power to each circuit.

With this information, the wiring diagram was developed into seven sections: Battery & Frame Grounding, Power Distribution, PLC Power, PLC Inputs & Outputs, HMI & PLC Communications, Programming Setup, and Box Wiring Interconnection.

Battery & Frame Grounding, as shown in Figure 23 contains the start of the circuit, naming conventions, and wire standards. This diagram shows two 12V batteries wired in series. This is used to power and ground the PLC. The power generated by the batteries is sent through a master shutoff switch, which is mounted beside the HMI. Following general wiring standards, the blue wire is utilized to show low voltage, the blue and white striped wire shows the return wire, and green and yellow striped wire indicates ground.

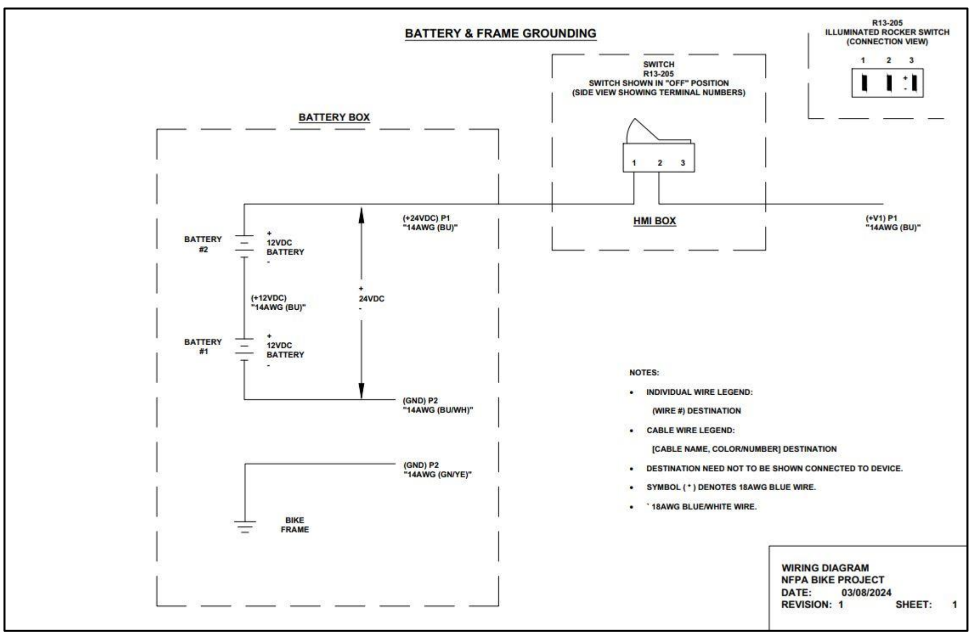


Figure 23: Battery and frame grounding

Power Distribution, as shown in Figure 24 shows the electrical current from the previous figure moving through the power distribution block (left side of diagram). Additionally, it shows that all the HMI and PLC components are grounded, as well as grounded to the bike (middle of diagram). Lastly, the current leaving the power distribution block then passes through the fuses before entering the PLC and HMI (right side of the diagram).

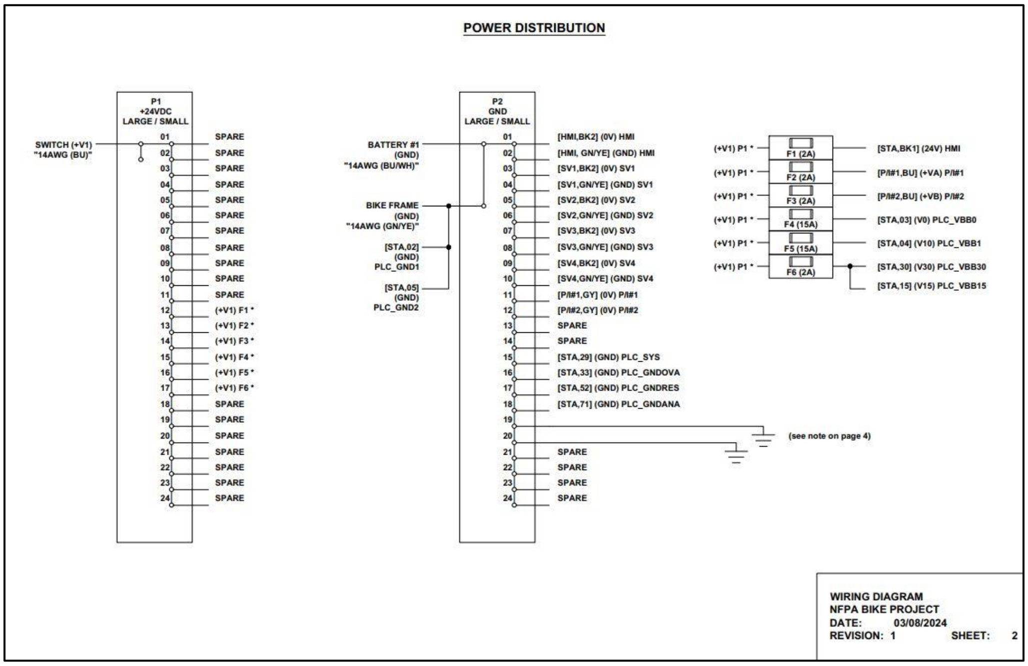


Figure 24: Overall power distribution

PLC Power, seen in Figure 25, shows the continuation of the electrical current from the previous figure by supplying power to the various connection points on the PLC.

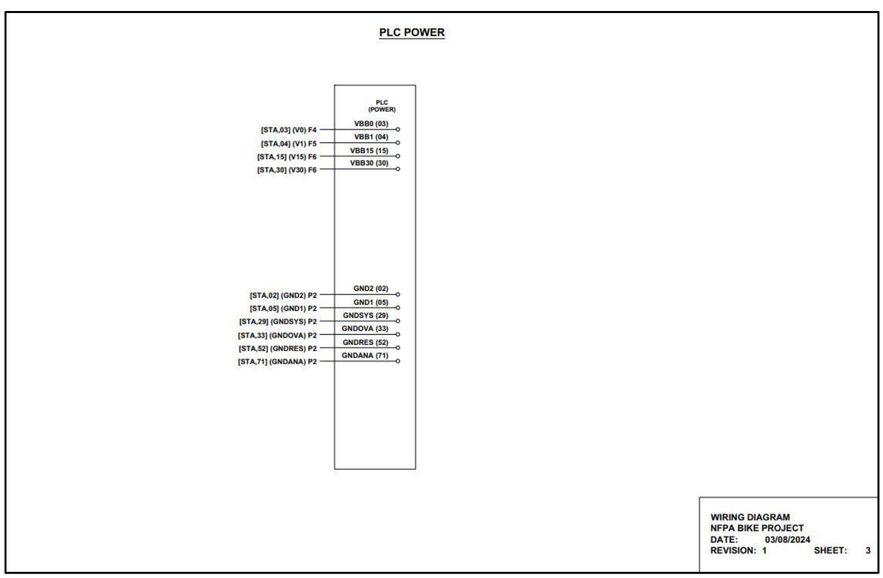


Figure 25: Diagram of PLC power

PLC Inputs & Outputs, as shown in Figure 26 shows the analog inputs on the left side and the discrete outputs on the right side. The two-port splice connectors are used to connect the pressure sensors and solenoids to the PLC. Note that there are two channels for each function (primary and backup ports).

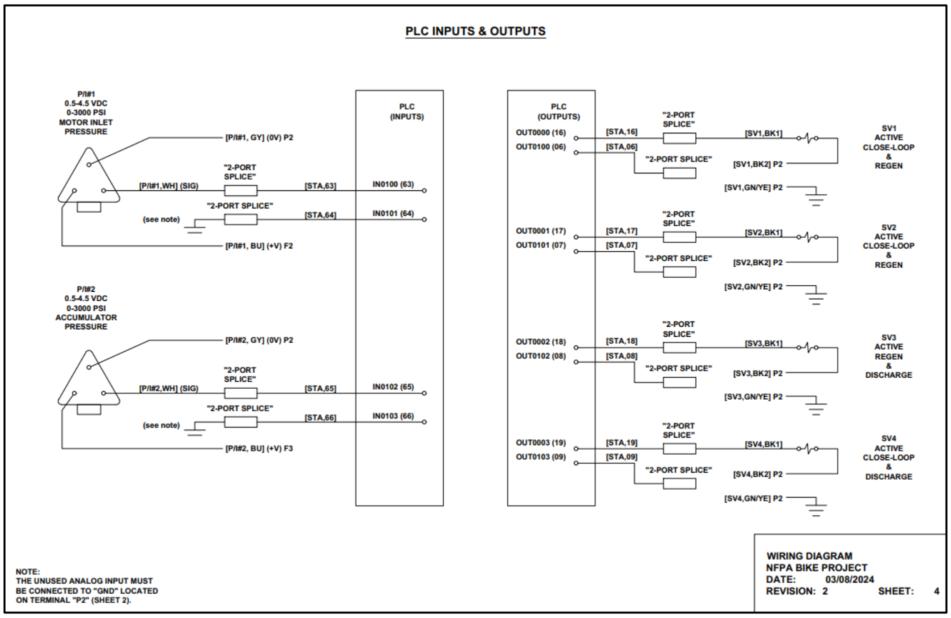


Figure 26: PLC inputs and outputs

HMI & PLC Communications is shown in Figure 27. This shows the connection of the ethernet communication cable from the PLC to the HMI. In addition, the HMI power wiring is depicted in the figure as well.

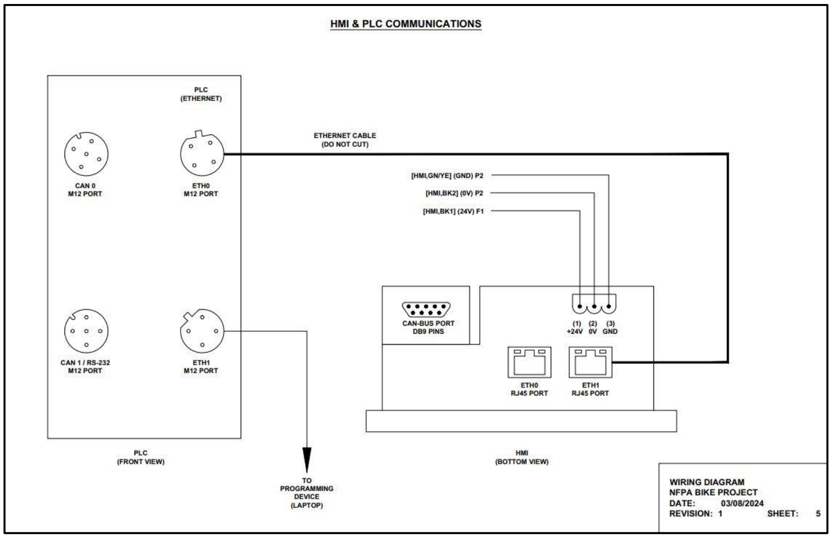


Figure 27: HMI & PLC communications

Programming Setup, as shown in Figure 28, documents all basic communication setup information needed in the programming software. All required software utilized in this project can be found in this documentation.

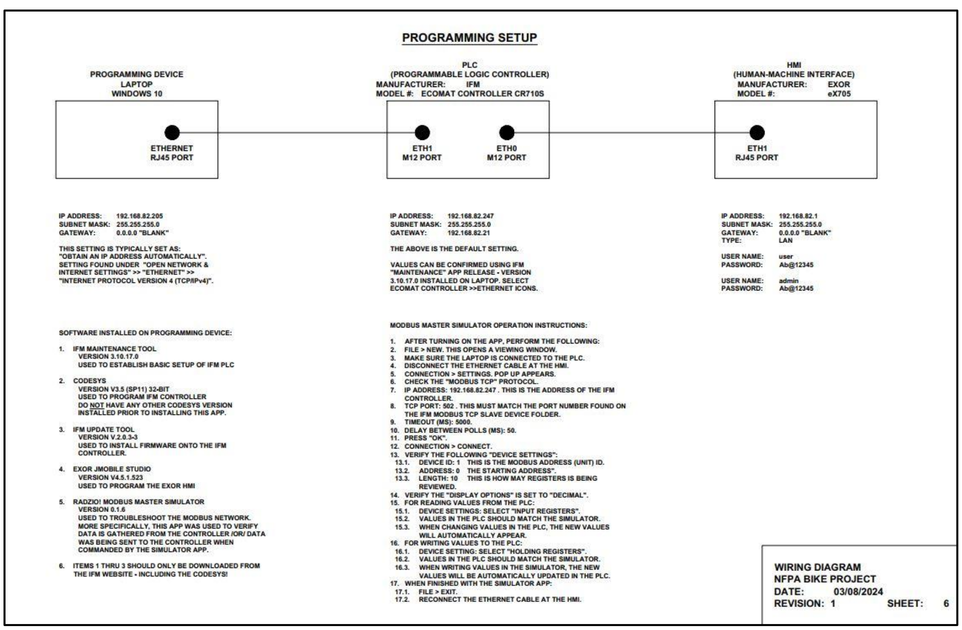


Figure 28: Programming setup

Lastly, Box Wiring Interconnection is shown in Figure 29. This provides a general layout of the wiring paths between the different electrical boxes.

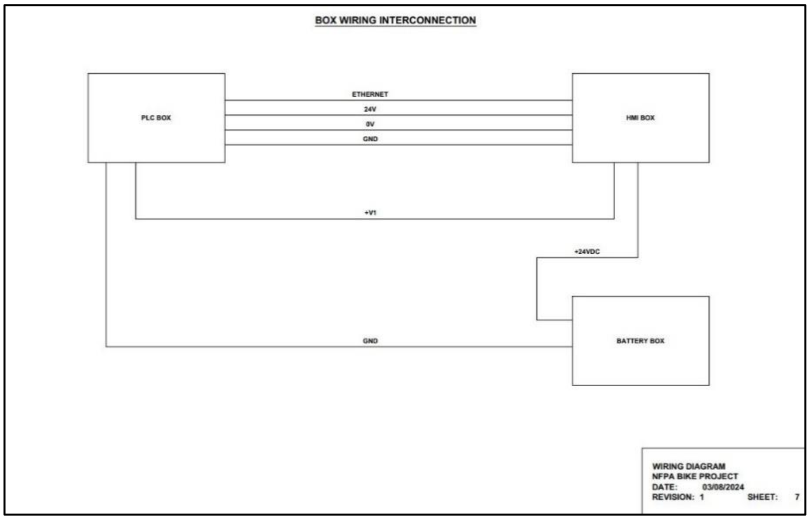


Figure 29: Box wiring interconnection

Programming: HMI

The HMI acts as the driver of the program. It can read and write information to the PLC. For this project, the HMI has a couple of necessary buttons and informational features, as shown in Figure 30. Most importantly, there are four buttons on the HMI labeled as the following: Pedal, Close Loop, Discharge, and Regen. These correspond to the bike's four modes of operation. Initially, the four buttons are visually displayed with a white background. Once a button is selected, it will illuminate with a green background. This acts as a visual indicator, so the user can know what mode the bike is currently in. Additionally, once a button

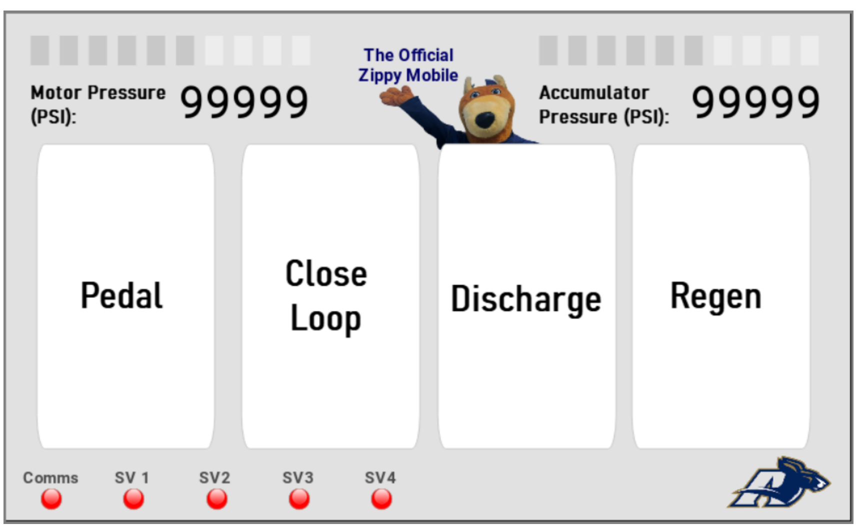


Figure 30: HMI screen as used on the bike

is selected, all other buttons are turned off.

For further troubleshooting information, in the bottom left corner of the HMI screen, are five indicator lights. On the leftmost side, denoted as Comms, is a pilot light to indicate whether the PLC and HMI are communicating. If there are no issues, the button will remain green. However, if the signal is interrupted, the button will become red to alert the user of this issue. The remaining four pilot lights correspond to the four solenoids (SV1, SV2, SV3, SV4). When the user selects a mode, the active solenoids will change from grey to green. This is mostly utilized in the testing phase to verify that the buttons are sending signals to the correct solenoids.

Lastly, at the top of the HMI are two meters referred to as Motor Pressure and Accumulator Pressure. To improve and understand the performance of the bike, as well as keep the rider safe, our hydraulic circuit includes two pressure sensors. Motor Pressure displays the pressure (psi) of the fluid at the inlet of the motor. On the above graph above the pressure reading, lower pressures (0-1,999 psi) will display as green, middle range pressures (2,000-2,799 psi) will display as yellow, and higher pressures (>2,800 psi) will display as red. Accumulator pressure indicates how much pressure will be saved in our accumulator. Once a pressure of 3,000 psi is met, fluid will start flowing through relief valves in the circuit. As a result, the color scheme of the graph on top of the pressure reading can be used to remind the rider to switch to the discharge phase so the rider can efficiently use the energy the bike has stored. Lower pressures (0-1,999 psi) are displayed as yellow to help indicate that there is more room to build up energy. Middle pressures (2,000-2,799 psi) are displayed as green to indicate that this is the most optimal time to utilize the energy. High pressures (>2,800 psi), are displayed as red to indicate the fluid is at risk to start draining through the relief valve.

The HMI interface program (Appendix A) entails exporting variables from the PLC, linking the variables to buttons and indicators, and creating rules to manipulate the visual display of the feature based on outside data. The HMI dictates how the PLC should behave and reports to the user the status of the PLC's behavior. The program that explicitly controls the inputs and outputs (solenoids, sensors, etc.) is found solely in the PLC program.

Programming: PLC

The IFM's PLC requires the user to be familiar with CodeSys, which is an open-source, third-party operating system. This operating system can be used in a variety of industry applications. Please note that for this project, version V3.5 (SP 11) was utilized. This is the only compatible version for the PLC. CodeSys allows the programmer to utilize a variety of different implementation languages (continuous function chart, function block diagram, ladder logic, structured text, etc.). Ladder logic was chosen for this project since it requires less training time.

To ensure functional cohesion within the program, the project is divided into methods that execute a single objective. This results in elements that are easy to reuse, troubleshoot, and test. However, for the individual program organization units (POUs) to share data, global variables have been added. A complete list of global variables is shown in Figure 31.

```
{attribute 'qualified_only'}
VAR_GLOBAL

    PI1_psi:                REAL;
    PI2_psi:                REAL;
    SV1_boolean:            BOOL;
    SV2_boolean:            BOOL;
    SV3_boolean:            BOOL;
    SV4_boolean:            BOOL;
    Pedal_Mode:             BOOL;
    Discharge_Mode:         BOOL;
    Close_Loop_Mode:        BOOL;
    Regen_Mode:             BOOL;
    Accumulator_Pressure_Range: WORD;
    Motor_Pressure_Range: WORD;
    discharge_pb: BOOL;
    pedal_pb: BOOL;
    close_loop_pb: BOOL;
    regen_pb: BOOL;
    discharge_lt: BOOL;
    pedal_lt: BOOL;
    close_loop_lt: BOOL;
    regen_lt: BOOL;
    test:   BOOL;

END_VAR
```

Figure 31: List of complete global variables

To set up a network that handles all HMI inputs and outputs, one program was created: HMI_data. Before the ladder logic can be created within this program, all required variables are declared at the top of the program, as shown in Figure 32. Note that variables beginning with “p_h” indicate the variable is taking data from the PLC to the HMI. Additionally, variables beginning with “h_p” indicate the variable is taking data from the HMI to the PLC.

```
PROGRAM HMI_data
VAR

    p_h_pi1_psi:        UINT;
    p_h_pi2_psi:        UINT;
    p_h_acc_range:      WORD;
    p_h_motor_range:    WORD;
    p_h_sv1_bool:       BOOL;
    p_h_sv2_bool:       BOOL;
    p_h_sv3_bool:       BOOL;
    p_h_sv4_bool:       BOOL;
    p_h_discharge_lt:   BOOL;
    p_h_pedal_lt:       BOOL;
    p_h_close_loop_lt:  BOOL;
    p_h_regen_lt:       BOOL;

    h_p_discharge_pb:   BOOL;
    h_p_pedal_pb:       BOOL;
    h_p_close_loop_pb:  BOOL;
    h_p_regen_pb:       BOOL;

END_VAR
```

Figure 32: Declaring required variables for HMI_data

The first rung in the network is used purely for testing purposes, as shown in Figure 33. When the test bit is turned on, an arbitrary pressure value is sent to the HMI. When the test bit is off, the value from the pressure transmitters is sent to the HMI.

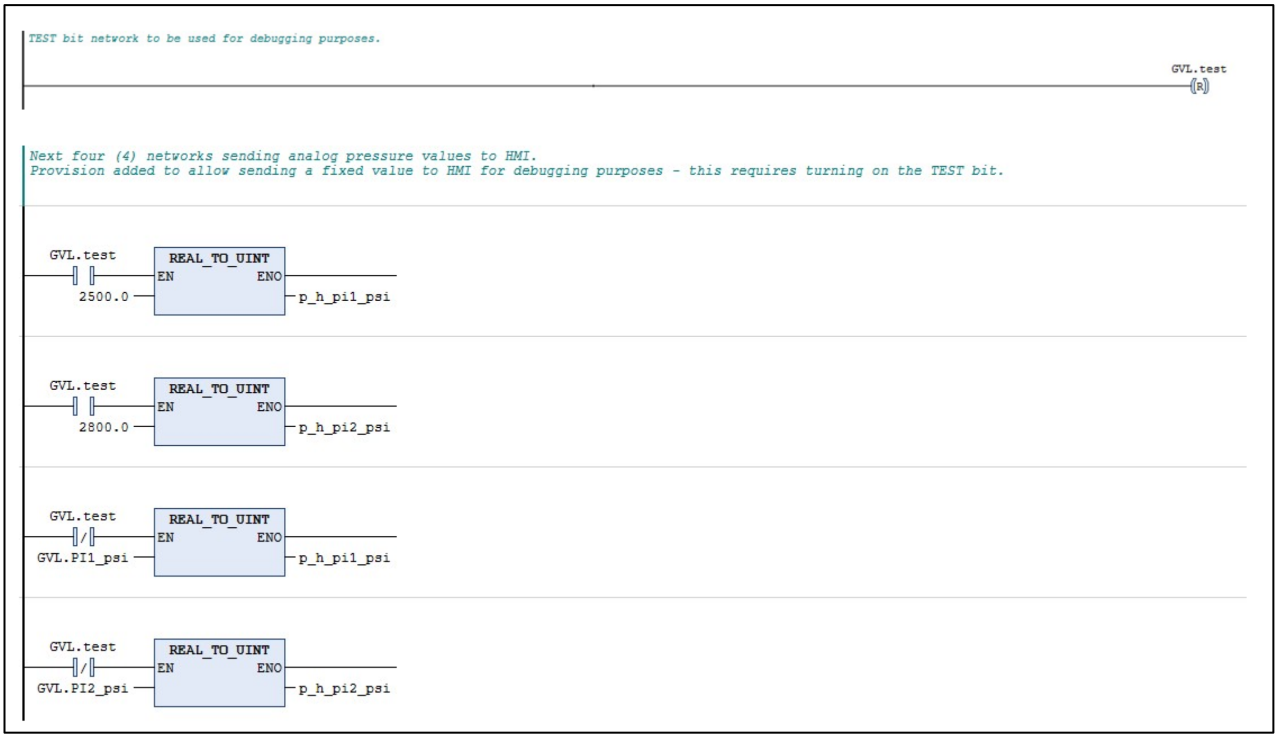


Figure 33: Testing HMI display

The following networks, in Figure 34, send feedback to the HMI indicating which solenoid values are energized and which operating modes are activated

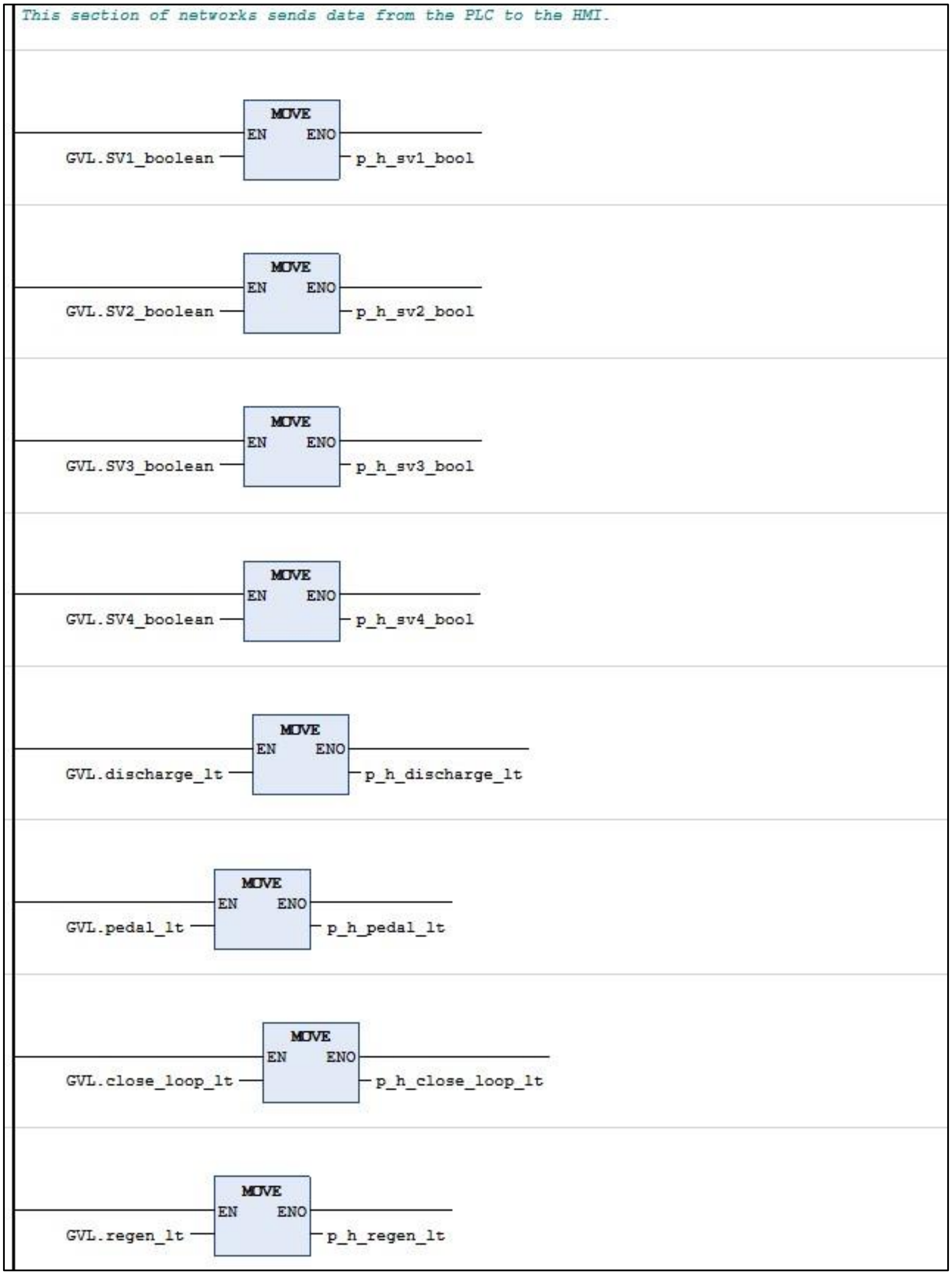


Figure 34: Rungs that send feedback to HMI to indicate solenoid energization

Figure 35 shows variables that contain values associated with pressure ranges and colors. This allows the HMI to display the appropriate color based on the current pressure value.

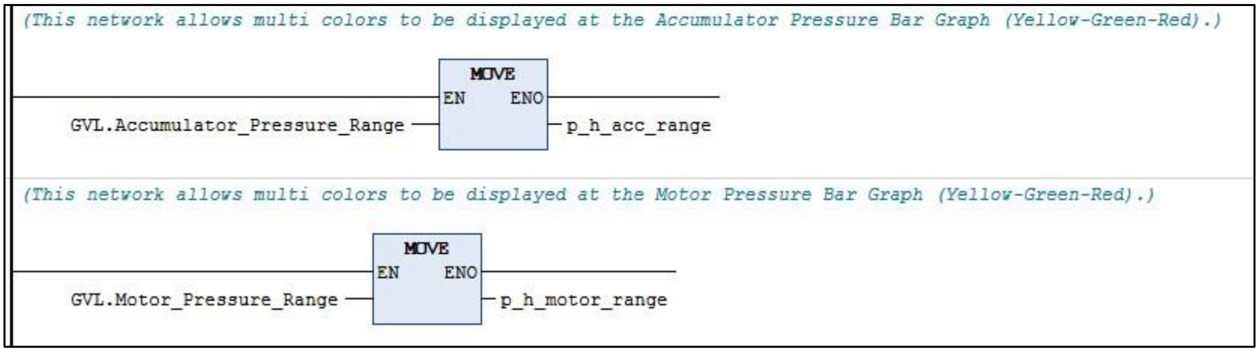


Figure 35: Send pressure ranged from PLC to variables that can communicate with the HMI

The last section of this program (Figure 36) contains variables that contain data coming from the HMI. This data is then set to global variables, which are used in the following POUs.

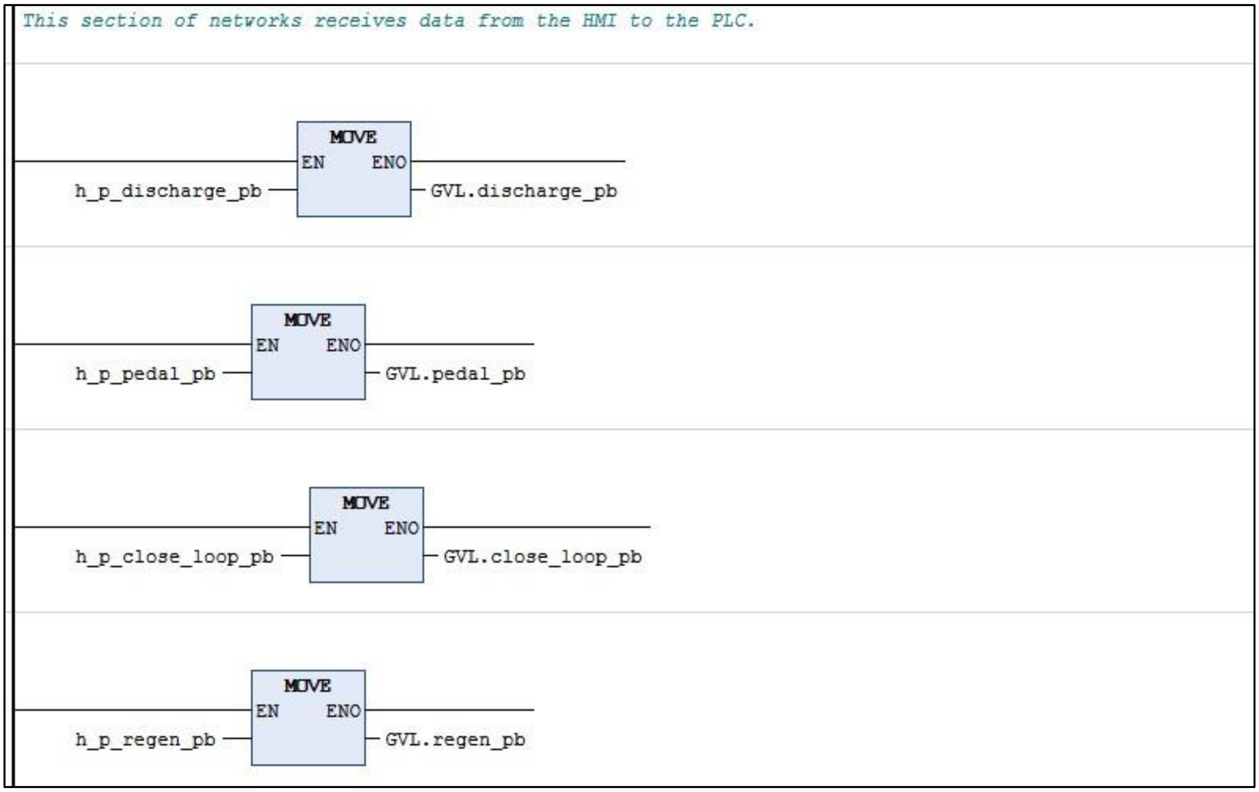


Figure 36: HMI related variables sending data to the PLC

To set up the analog inputs, two programs were created: analog_input_PI_1 and analog_input_PI_2. For demonstration purposes, the variables and functions in analog_input_PI_1 are shown below. Refer to Appendix B for the entirety of the PLC program.

Before the ladder logic can be created, all required variables are declared at the top of the program, as shown in Figure 37. Note that variables Input_0100 and Input_0101 are set to a type of variable called ifmIOcommon.Input. This is a prebuilt structure that is an IFM-specific template. This structure allows the programmer to create and control inputs into the PLC.

```
PROGRAM analog_input_PI_1

VAR

    Input_0100: ifmIOcommon.Input;      // Creating input port (Input_0100) using ifm template
    PI1_CH1_mv: UINT;                   // Result of conversion from input signal to tag in mV (unipolar 16 bit tag- UINT)
    Input_0101: ifmIOcommon.Input;      // Creating input port (Input_0101) using ifm template. BACK UP PORT
    PI1_CH2_mv: UINT;                   // Result of conversion from input signal to tag in mV (unipolar 16 bit tag- UINT). BACK UP PORT
    PI1_selected_mv: UINT;              // Value of the higher analog input channel (higher pressure)
    PI1_selected_mv_real: REAL;         // Result of conversion from UNIT to float
    mul_1: REAL;                        // Temp tag used in pressure calculation of mV to psi

END_VAR
```

Figure 37: Declaring variables in analog_input_PI1

The following figures show the ladder logic used to convert the electric signal from the pressure sensor to a readable numerical value (psi). The first rung shows the configuration of the primary sensor. The pressure transmitter data sheet provided by the manufacturer states that the sensor is scaled for 0-3,000 psi. Additionally, 0.5 volts (V) corresponds to 0 psi, while 4.5 V corresponds to 3,000 psi. As a result, when utilizing IFM's input template, we selected the smallest corresponding voltage range, 0-10 V. Additionally, different filters can be selected to ensure no background noise is altering the quality of the signal. For this input, IFM's standard filter, FILTER_0, was selected. After the signal enters through port 0100 on the PLC, the analog data in millivolts (mV) is sent to PI1_CH1_mv.

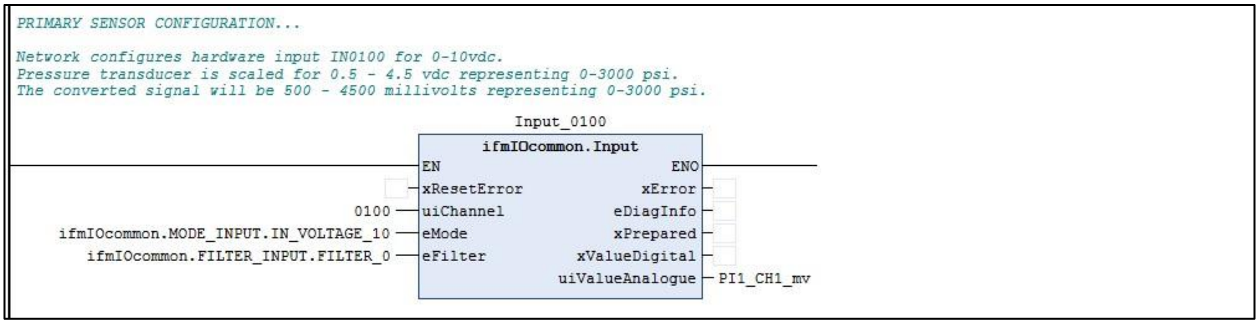


Figure 38: Primary sensor configuration

The following figure (Figure 39) is a repeat of the previous figure for the backup sensor.

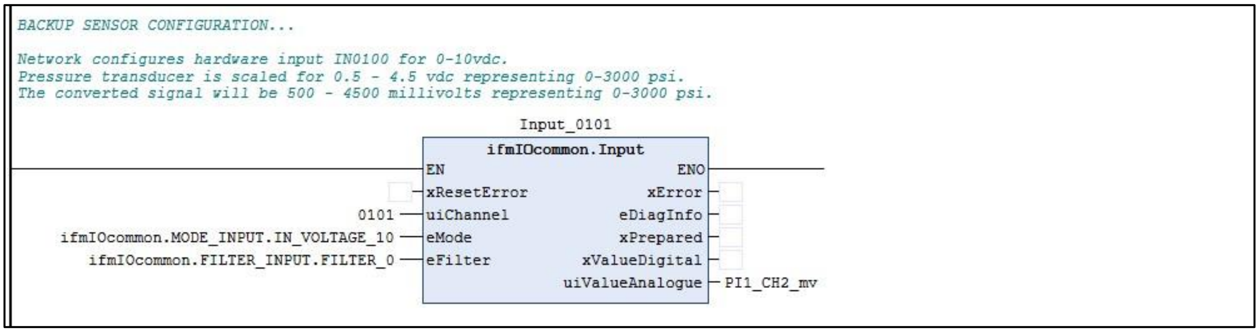


Figure 39: Backup sensor configuration

Figure 40, as shown below, is created to find the active port (original or backup). The network takes the larger signal and promotes it to PI1_selected_mv. All further calculations will be performed with the data from this variable.

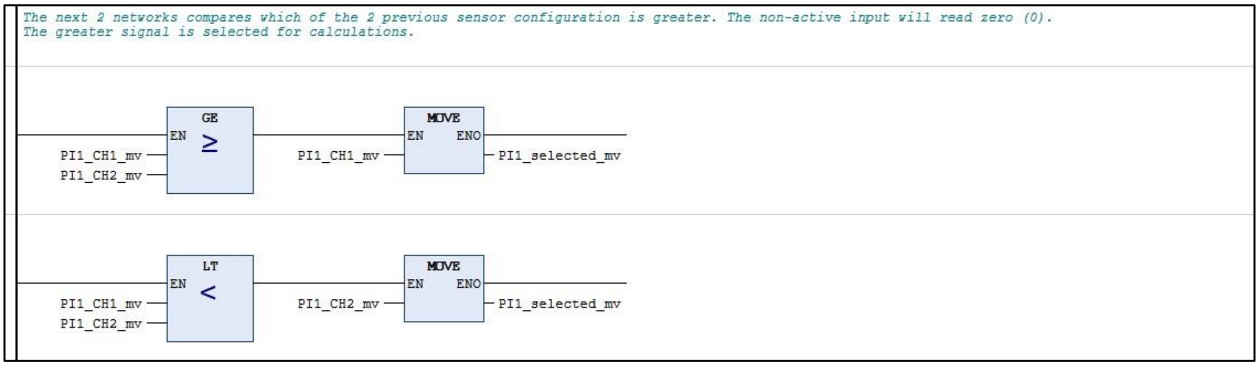


Figure 40: Rungs to find active port

The following rung (Figure 41) shows the conversion from UNIT to REAL. This conversion is crucial, as the signal is changed from a 16-bit integer to a number that can contain a fractional component.

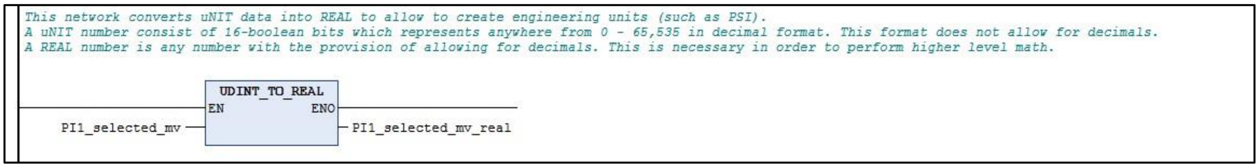


Figure 41: Converting UNIT to REAL

The pressure transducer relationship between voltage and pressure is linear. As a result, by completing the below calculations, a voltage-to-pressure conversion formula was created that can be utilized in the program, as shown in Figure 42.

$$x_1 = 0.5\,V,\;\; x_2 = 4.5\,V \tag{24}$$

$$y_1 = 0\,psi,\;\; y_2 = 3{,}000\,psi \tag{25}$$

$$0 = 0.5m + b \tag{26}$$

$$3{,}000 = 4.5m + b \tag{27}$$

$$-0.5m = 3{,}000 - 4.5m \tag{28}$$

$$m = 0.75 \tag{29}$$

$$3{,}000 = (0.75) * (4.5) + b \tag{30}$$

$$b = 375 \tag{31}$$

$$y = 0.75x + 375 \tag{32}$$

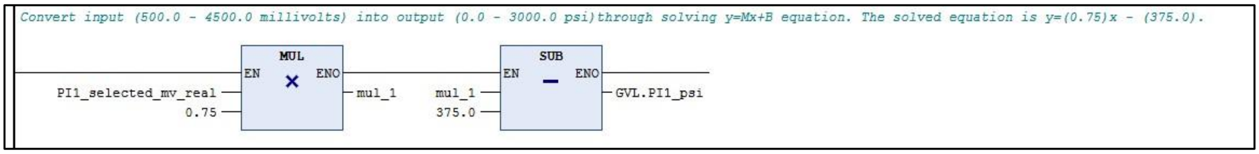


Figure 42: Voltage to pressure conversion

In the event that the pressure transmitter becomes disconnected, and the value read from a signal is negative, the output pressure is set to zero, as shown in Figure 43.

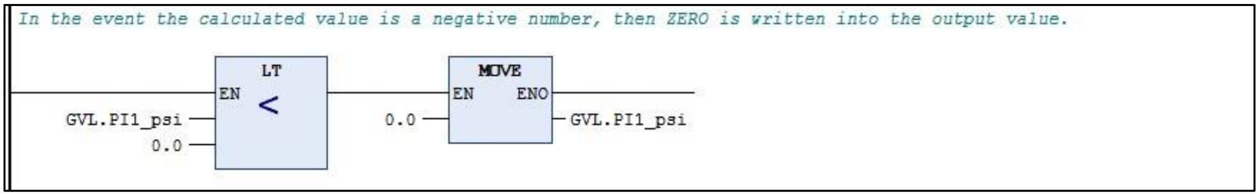


Figure 43: Setting output pressure to zero in the event of a disconnected pressure transmitter

There are four programs created for each of the corresponding solenoids: discrete_output_SV_1, discrete_output_SV_2, discrete_output_SV3, and discrete_output_SV4. For demonstration, the variables and functions for discrete_output_SV_1 are shown below. Refer to Appendix B for the entire program.

Similarly to the pressure sensor program, before the ladder logic can be created, all required variables are declared at the top of the program, as shown in Figure 44. Note that variables Input_0000 and Input_0100 are set to a type of variable called ifmIOcommon.Output. This is also a prebuilt structure that is an IFM-specific template. This structure allows the programmer to create and control outputs from the PLC.

```
PROGRAM discrete_output_SV_1

VAR

    Output_0000: ifmIOcommon.Output;
    Output_0100: ifmIOcommon.Output;
    SV1_uNIT_primary: UINT;
    SV1_uNIT_backup: UINT;

END_VAR
```

Figure 44: Declaring variables for discrete_output_SV_1

The following figures show the ladder logic used to create the output from the PLC to the solenoid. The first rung shows the configuration of the primary valve (Figure 45). Based on a boolean input (0 or 1), the physical output channel is told to turn off or on. First, the boolean variable is converted into UNIT which is required by IFM's output template. For our purposes, the output must be discrete (on or off), not a range of values. As a result, IFM's standard OUT_DIGITAL_CSO mode was selected in the template. Similar to the input template, different filters can be selected to ensure no background noise is altering the quality of the signal. For this, IFM's standard filter, FILTER_0, was selected again.

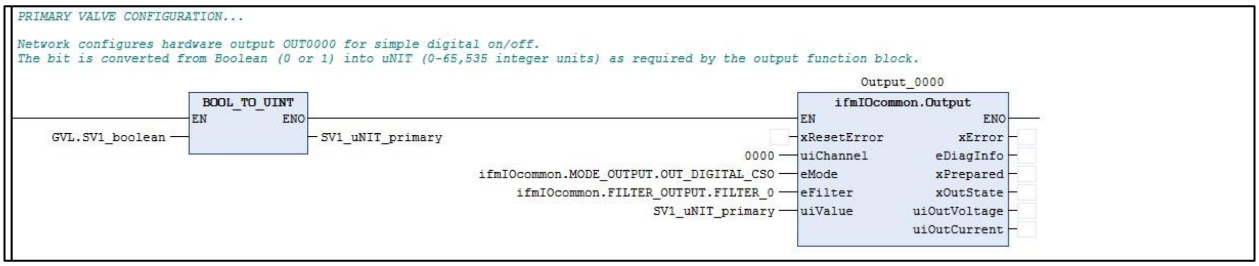


Figure 45: Solenoid 1 primary configuration

Figure 46 shows a repeat of the previous figure for the backup solenoid port.

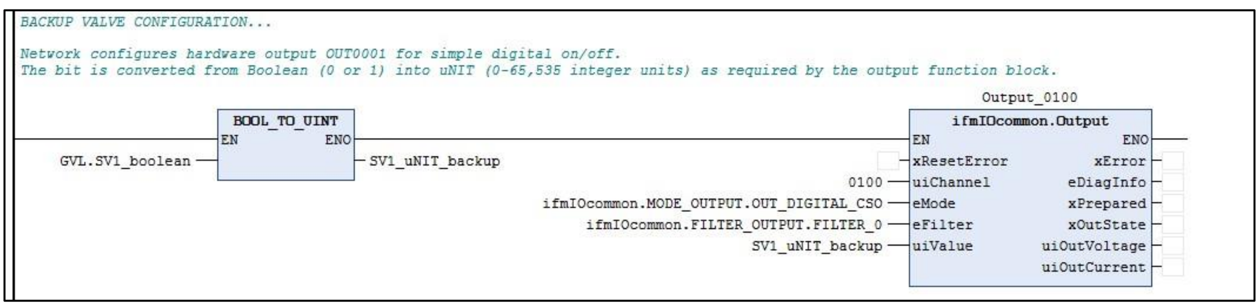


Figure 46: Solenoid 1 back-up configuration

The last POU created is referred to as the main. This contains all the combinational logic that fledges out the relationship between the HMI and PLC. Following the same format as other POUs, all variables are declared before the ladder logic, which is shown in Figure 47.

```
PROGRAM main
VAR

    Accumulator_Test_Value_psi: REAL;
    Motor_Test_Value_psi: REAL;
    R_TRIG_Discharge: R_TRIG;
    R_TRIG_Pedal: R_TRIG;
    R_TRIG_Close_Loop: R_TRIG;
    R_TRIG_Regen: R_TRIG;

END_VAR
```

Figure 47: Declaring variables for Main program

The following section of the program, as shown in Figure 48, handles the setting and resetting of the bike's operating modes. The logic is set up so that only one mode is active at any given time.

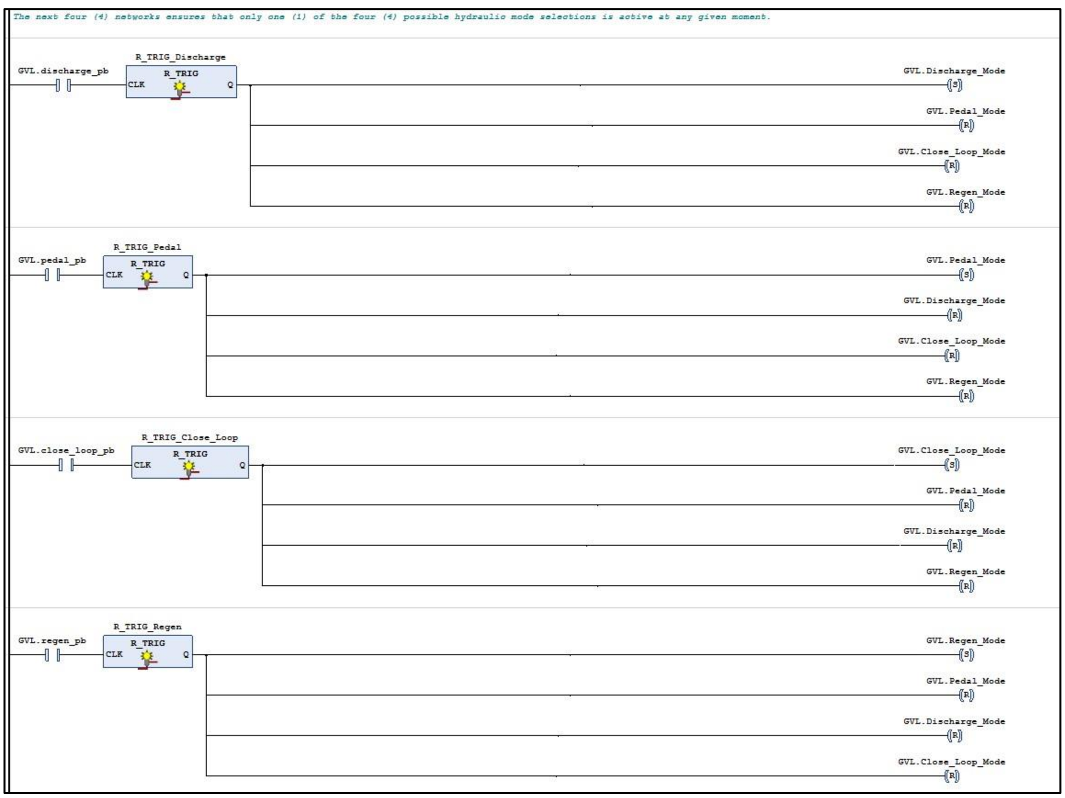


Figure 48: Setting and resetting bike's operating modes

Based on which mode is activated, the following rungs set variables that are later used in changing the color of the push buttons on the HMI (Figure 49).

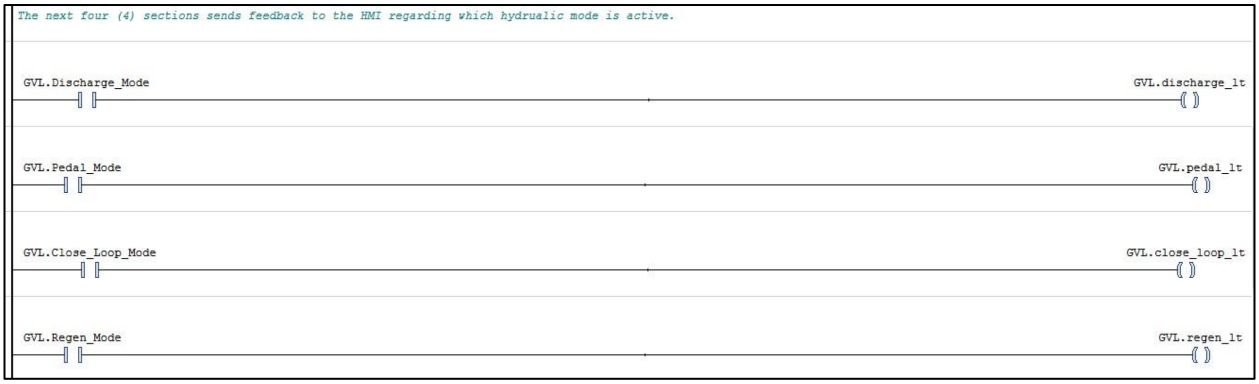


Figure 49: Setting variables for changing color of HMI buttons

Additionally, depending on which mode is activated, the appropriate solenoid values are turned on, as shown below in Figure 50.

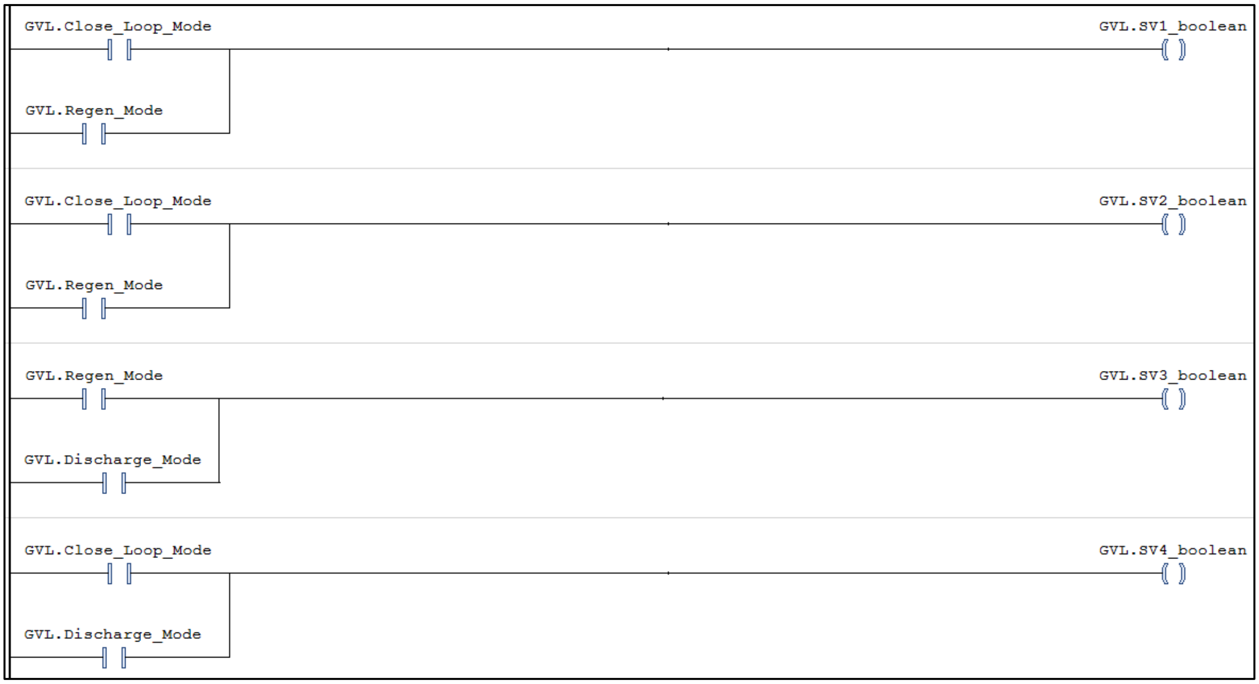


Figure 50: Setting active solenoid values

The last section in the program (Figure 51), dictates which colors should be on by setting numerical values based on pressure ranges. Ultimately, this impacts the bar meters on the HMI.

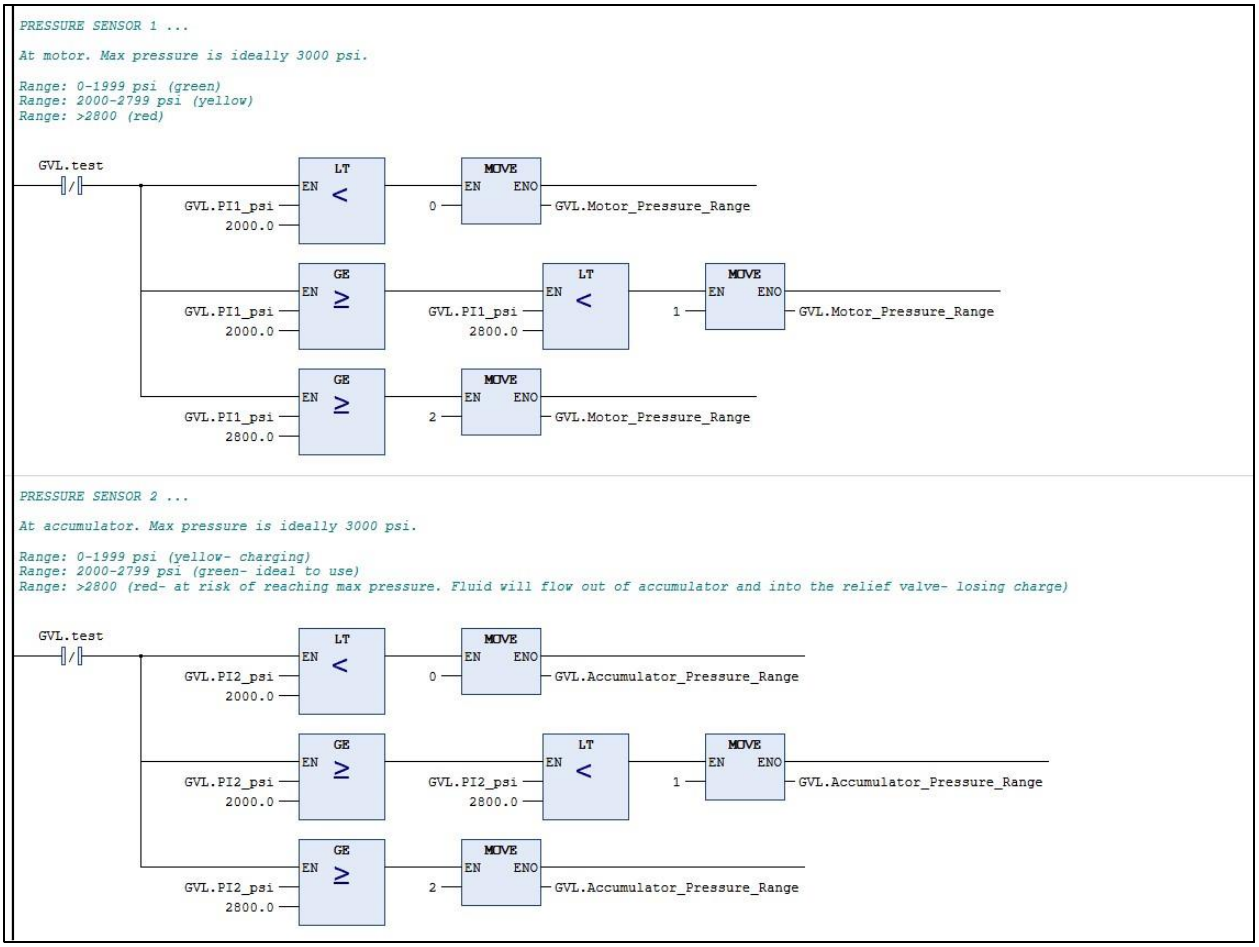


Figure 51: Setting numerical values for colors on pressure bars

<u>Programming: Modbus</u>

To allow two or more electronic devices to send and receive data, a communication protocol must be selected. Modbus TCP communication protocol was selected for this project. By using this template, the user can easily assign registers to communicate with other devices on the network. For this to be successfully implemented, the setup must take place in the PLC, as well as the HMI. The HMI will act as the Modbus master device. The master device is responsible for initiating and verifying communications with the slave devices in the network. The PLC is assigned to be a Modbus slave device. A slave device is unable to push or pull data and must wait for commands from the Modbus master device.

To begin setup in the PLC, there is general information that needs to be filled in, as shown in Figure 52. The slave port is a virtual location on a network device that is necessary for identifying itself on the network. The unit ID, also known as a Modbus address, is used in addition to the slave port to identify itself on a Modbus network. A Modbus network can accommodate up to 255 devices. A holding register contains data that is transferred from the master device. An input register contains data that is transferred to the master device. In both cases, 10 registers were allocated for this purpose. The starting address for both the holding and input registers starts at zero. This means the formal Modbus address range for the input registers is 300000 through 300009. Additionally, the formal Modbus address range for the holding registers is 400000 through 400009.

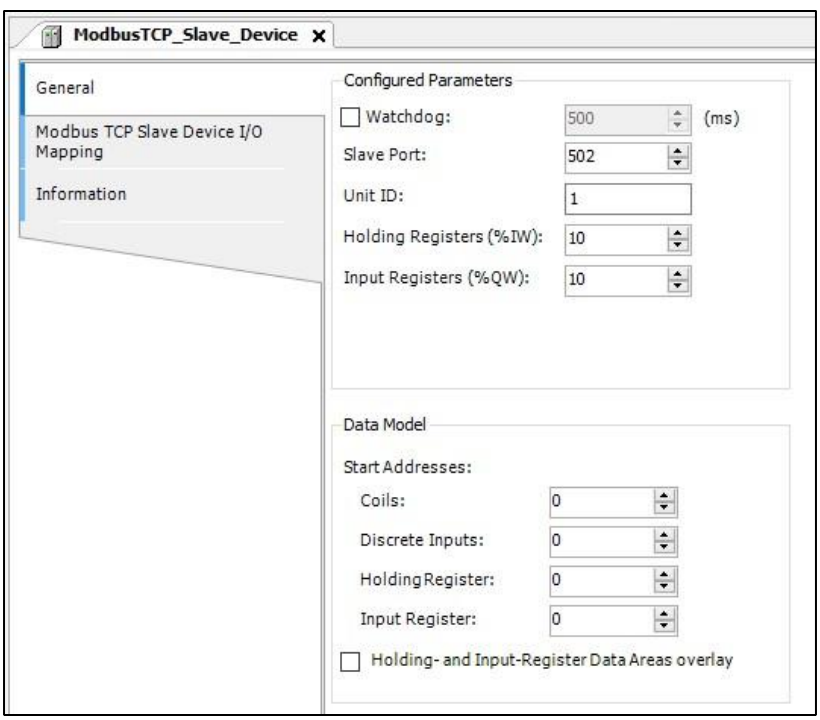
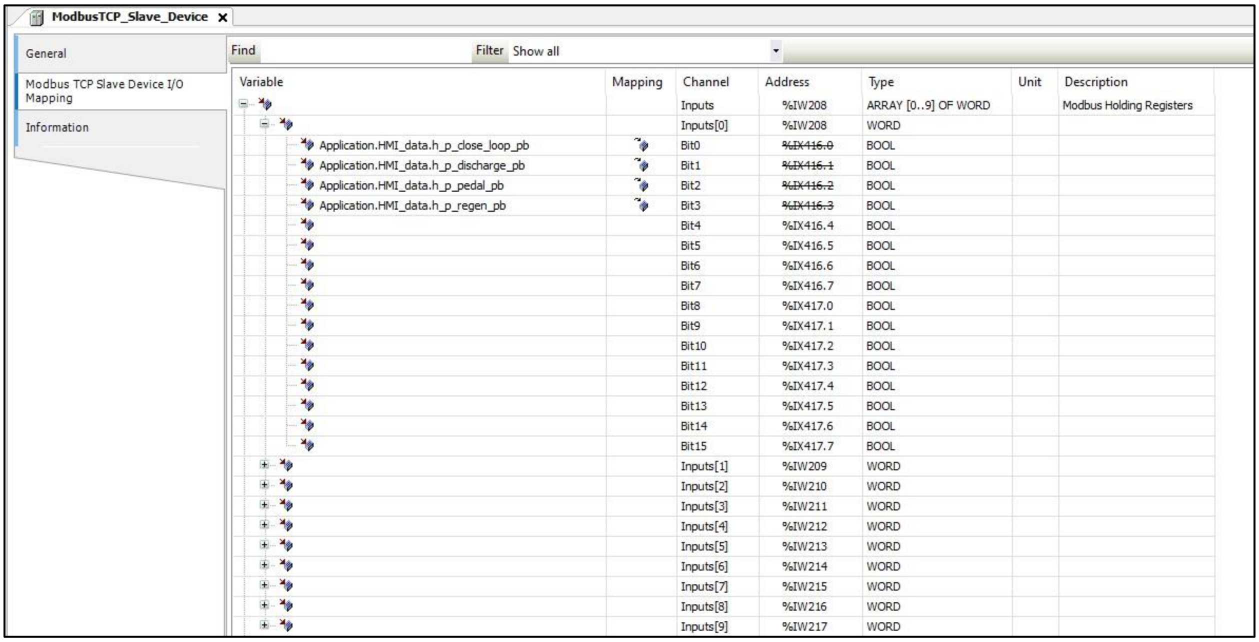


Figure 52: General information for PLC setup

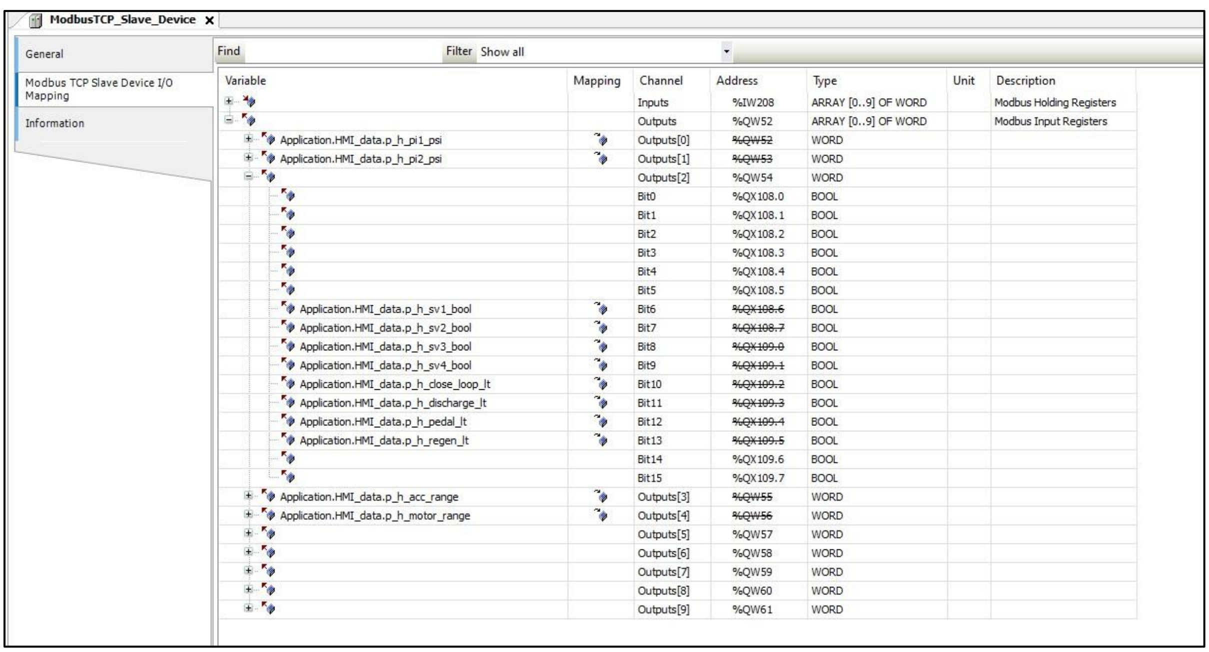
The following figure details the data being received from the HMI (Figure 53).



Variable	Mapping	Channel	Address	Type	Unit	Description
		Inputs	%IW208	ARRAY [0..9] OF WORD		Modbus Holding Registers
		Inputs[0]	%IW208	WORD		
Application.HMI_data.h_p_close_loop_pb		Bit0	~~%IX416.0~~	BOOL		
Application.HMI_data.h_p_discharge_pb		Bit1	~~%IX416.1~~	BOOL		
Application.HMI_data.h_p_pedal_pb		Bit2	~~%IX416.2~~	BOOL		
Application.HMI_data.h_p_regen_pb		Bit3	~~%IX416.3~~	BOOL		
		Bit4	%IX416.4	BOOL		
		Bit5	%IX416.5	BOOL		
		Bit6	%IX416.6	BOOL		
		Bit7	%IX416.7	BOOL		
		Bit8	%IX417.0	BOOL		
		Bit9	%IX417.1	BOOL		
		Bit10	%IX417.2	BOOL		
		Bit11	%IX417.3	BOOL		
		Bit12	%IX417.4	BOOL		
		Bit13	%IX417.5	BOOL		
		Bit14	%IX417.6	BOOL		
		Bit15	%IX417.7	BOOL		
		Inputs[1]	%IW209	WORD		
		Inputs[2]	%IW210	WORD		
		Inputs[3]	%IW211	WORD		
		Inputs[4]	%IW212	WORD		
		Inputs[5]	%IW213	WORD		
		Inputs[6]	%IW214	WORD		
		Inputs[7]	%IW215	WORD		
		Inputs[8]	%IW216	WORD		
		Inputs[9]	%IW217	WORD		

Figure 5353: Data being received from HMI

The next figure details the data waiting to be sent to the HMI (Figure 54).



Variable	Mapping	Channel	Address	Type	Unit	Description
		Inputs	%IW208	ARRAY [0..9] OF WORD		Modbus Holding Registers
		Outputs	%QW52	ARRAY [0..9] OF WORD		Modbus Input Registers
Application.HMI_data.p_h_pi1_psi		Outputs[0]	~~%QW52~~	WORD		
Application.HMI_data.p_h_pi2_psi		Outputs[1]	~~%QW53~~	WORD		
		Outputs[2]	%QW54	WORD		
		Bit0	%QX108.0	BOOL		
		Bit1	%QX108.1	BOOL		
		Bit2	%QX108.2	BOOL		
		Bit3	%QX108.3	BOOL		
		Bit4	%QX108.4	BOOL		
		Bit5	%QX108.5	BOOL		
Application.HMI_data.p_h_sv1_bool		Bit6	~~%QX108.6~~	BOOL		
Application.HMI_data.p_h_sv2_bool		Bit7	~~%QX108.7~~	BOOL		
Application.HMI_data.p_h_sv3_bool		Bit8	~~%QX109.0~~	BOOL		
Application.HMI_data.p_h_sv4_bool		Bit9	~~%QX109.1~~	BOOL		
Application.HMI_data.p_h_close_loop_lt		Bit10	~~%QX109.2~~	BOOL		
Application.HMI_data.p_h_discharge_lt		Bit11	~~%QX109.3~~	BOOL		
Application.HMI_data.p_h_pedal_lt		Bit12	~~%QX109.4~~	BOOL		
Application.HMI_data.p_h_regen_lt		Bit13	~~%QX109.5~~	BOOL		
		Bit14	%QX109.6	BOOL		
		Bit15	%QX109.7	BOOL		
Application.HMI_data.p_h_acc_range		Outputs[3]	~~%QW55~~	WORD		
Application.HMI_data.p_h_motor_range		Outputs[4]	~~%QW56~~	WORD		
		Outputs[5]	%QW57	WORD		
		Outputs[6]	%QW58	WORD		
		Outputs[7]	%QW59	WORD		
		Outputs[8]	%QW60	WORD		
		Outputs[9]	%QW61	WORD		

Figure 54 54: Data waiting to be sent to HMI

To begin Modbus setup in the HMI program, the user must go to the protocol tab to select Modbus TCP and set up general information regarding the slave device. This allows the master to locate the slave device. The alias serves as a nickname for the selected slave device. The IP address is the ethernet address on the network for the PLC. The port number and Modbus ID must match the corresponding numbers in the PLC setup. Timeout is the amount of time the master device can attempt to reconnect to the PLC before declaring a communication error. Lastly, the PLC must be set with generic Modbus, which states the starting address is zero.

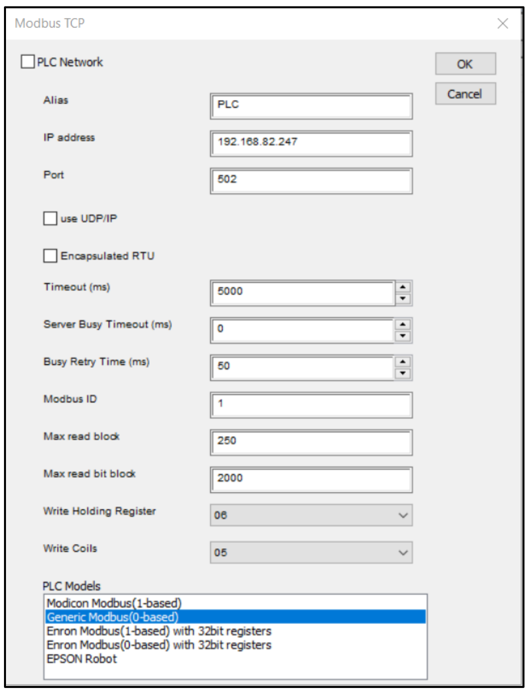


Figure 55 55: Modbus TCP setup information

Next all the applicable variables that contain data that is transferred between the HMI and PLC are uploaded to the HMI setup, as shown in Figure 57. Additional information regarding Modbus can be found in Appendix C.

Data	Type	Tag name	Tag URI	R/W
Modbus TCP:prot2 Alias: PLC Model: Generic Modbus(0-based)	Container			
p_h_motor_range	unsignedShort	PLC/p_h_motor_range	1?IREG?300004?unsignedShort	R
p_h_acc_range	unsignedShort	PLC/p_h_acc_range	1?IREG?300003?unsignedShort	R/W
p_h_regen_lt	boolean	PLC/p_h_regen_lt	1?IREG?300002.13?boolean	R
p_h_pedal_lt	boolean	PLC/p_h_pedal_lt	1?IREG?300002.12?boolean	R
p_h_discharge_lt	boolean	PLC/p_h_discharge_lt	1?IREG?300002.11?boolean	R
p_h_close_loop_lt	boolean	PLC/p_h_close_loop_lt	1?IREG?300002.10?boolean	R
p_h_sv4_bool	boolean	PLC/p_h_sv4_bool	1?IREG?300002.9?boolean	R
p_h_sv3_bool	boolean	PLC/p_h_sv3_bool	1?IREG?300002.8?boolean	R
p_h_sv2_bool	boolean	PLC/p_h_sv2_bool	1?IREG?300002.7?boolean	R
p_h_sv1_bool	boolean	PLC/p_h_sv1_bool	1?IREG?300002.6?boolean	R
p_h_pi2_psi	unsignedShort	PLC/p_h_pi2_psi	1?IREG?300001?unsignedShort	R
p_h_pi1_psi	unsignedShort	PLC/p_h_pi1_psi	1?IREG?300000?unsignedShort	R/W
h_p_regen_pb	boolean	PLC/h_p_regen_pb	1?HREG?400000.3?boolean	W
h_p_pedal_pb	boolean	PLC/h_p_pedal_pb	1?HREG?400000.2?boolean	W
h_p_discharge_pb	boolean	PLC/h_p_discharge_pb	1?HREG?400000.1?boolean	W
h_p_close_loop_pb	boolean	PLC/h_p_close_loop_pb	1?HREG?400000.0?boolean	W

Figure 56: Variables that contain data that are transferred between the HMI and PLC

Pneumatics

Although having pneumatic features are not required in the overall design, it does allow the team to earn extra points when being critiqued in the competition. The idea for this team was to have a pop-up Zippy that appears as the rider crosses the finish line. This way, the team can have a celebratory moment and represent our school with our mascot.

To implement this feature, the team 3D printed a case for Zippy to hide in. On the bottom, there will be a piston that is attached to a platform. The piston was attached to a manual valve that can be switched to activate it. When the pressure is released, the platform will pop up, with Zippy attached on top, and she will also have a checkered flag to wave at the finish line. Below is a sketch of the original idea from brainstorming.

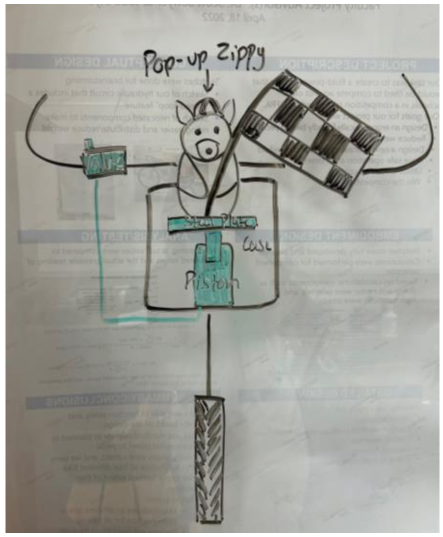


Figure 57: Initial sketch of pneumatic system

To utilize pneumatic features while riding the vehicle, the system needed to retain the air in the system. Although attaching a small air compressor to the bike was possible, this would have added additional weight, which is not ideal for an optimal design. To accomplish this, a toggle valve was attached to a reservoir to provide entry for air, while keeping it enclosed when the switch is flipped.

Another critical component in this circuit was the regulator. When the pressure was uncontrolled or set too high, it would cause Zippy to fly off her platform. So, to prevent this failure from occurring, the regulator was set to 20 psi. This was an ideal pressure for this utilization. Also, setting the pressure to this allowed us to use Zippy more throughout the race, since less air was being used to push up the Zippy. Below is a schematic of the circuit created for this vehicle, as well as the components on the bike.

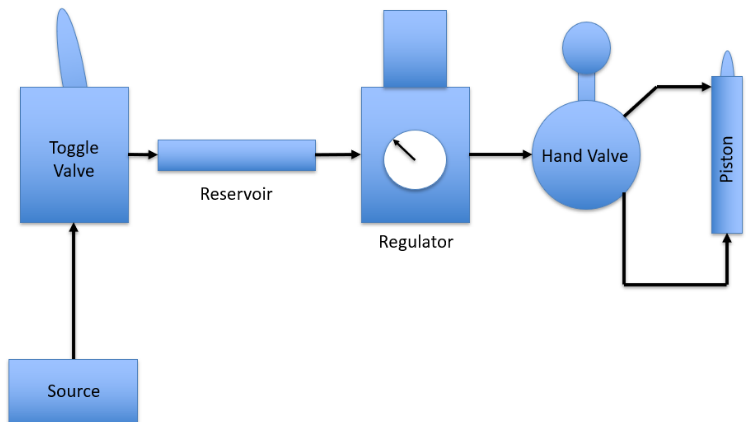


Figure 58: Pneumatic system schematic

Overall, the pneumatics were able to work well during the competition. They were able to switch on and off about 6-10 times. The system did lose air as the pneumatics were utilized, and the system needed to be charged regularly. Perhaps if a larger reservoir was used or if the fittings were tighter, the air would have lasted longer in the system. However, the judges at the competition were impressed with the creativity that was put into the pneumatics.

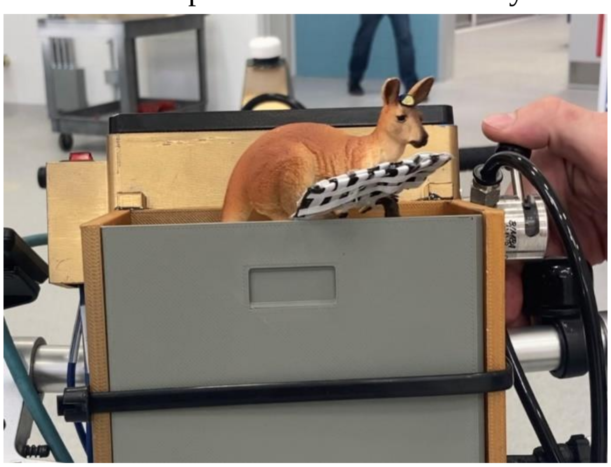


Figure 59: Zippy placed onto the piston inside of the pneumatics box

Figure 60: Pneaumatics box placement on the back of the HMI box

Codes and Standards

Codes and standards are highly important in the day-to-day operations of an engineer. These codes and standards, overall, exist to set a rule of consistency across engineering using mandatory legal requirements. They set the roles for the designing of systems, manufacturing or construction of systems, and the maintenance of these systems in the field. The consistency that codes and standards set to achieve are focused on safety and quality.

Working with a hydraulics circuit has many dangers that we need to be aware of. Our system will be highly pressurized, and proper precautions must be taken to ensure the safety of our team while working on or around the bike. As a result, a variety of codes and standards can be referenced. One of these is ISO 4413, which specifies general rules and safety requirements to be adhered to when designing hydraulic fluid power systems. This standard includes specifications for drawing circuit diagrams, assembly of fluid power components, adjusting these systems, and more. Similarly, it is key to adhere to OSHA standards regarding personal safety when working with the fluid-power vehicle. For example, OSHA-approved eyewear must be worn when working with a pressurized system. Additionally, when working with a system that used pressurized fluid, the team must be aware of leaks in the system that may cause injection injuries. Below is a table of codes and standards that apply to our project.

Table 1: Codes and Standards

Code/Standard Name and Institution	**Application Used**
ISO 4413:2010 Hydraulic Fluid Power	General assembly, installation, adjustment, and operation
ASME Boiler and Pressure Vessel Code, Section VIII	Accumulator safety
ISO 1875:2022 Hoses and hose assemblies	Hoses connecting our components, hard lines

Throughout our careers, we will be heavily influenced by codes and standards set in place. As previously discussed, safety is of the utmost importance in the design of all products and systems. Every engineer must understand the importance of making decisions and recommendations that keep the safety of others in mind while maintaining a quality of product that is consistent within the market.

Societal, Safety, & Sustainability Considerations

Societal Considerations

By converting human motion into mechanical power with hydraulic technology, we can create a more accessible biking experience. Unlike conventional chain-linked bicycles, energy can be stored and released based on the user's needs. Energy can be stored during times that have low energy requirements, like riding a bike down a hill. This energy can be released in a controlled manner during high-demand loads, such as when quick acceleration is needed or when a bike is moving up a hill. This feature helps to create a more comfortable transportation experience, as the user does not need to exert the same amount of strain, as shown during a normal biking encounter. As a result, this provides an exercise and transportation method that allows those with health issues to utilize. Additionally, by encouraging the bicycle market, densely populated areas can reduce the amount of traffic encountered.

Safety

As engineers, it is important to prioritize the safety of the public and the environment. It is critical to recognize that utilizing hydraulic fluid at high pressures comes with safety concerns, as high-pressure fluid can pierce the skin. In severe cases, fluid released from the system at high pressures can result in a loss of limbs and death. Additionally, it is important to protect the components handling the fluid. Fluid leaks can be hazardous, as any leakage can cause a slick surface which can result in an increased risk of accidents. Fluid leaks can also harm the environment.

Sustainability

A hydraulic bike provides a green form of transportation. While a traditional chain-linked bicycle is also a green form of transportation, the hydraulic bike has features mentioned previously that provide a more convenient way to travel. While considering convenience and sustainability, a hydraulic bike accomplishes both these qualities more than traditional methods, such as an automobile. In comparison to a car, which relies on the continuous input of fossil fuel, the hydraulic fluid in the bike is recycled through the system. This results in less consumption, and therefore does not constantly require money to operate. This can be more appealing for people to utilize. Additionally, the bike does not emit exhaust, unlike a car, which impacts the pollution in our world.

Cost

The following table lists the costs of the bike's components.

Table 2: Project finances breakdown

Component	Description	Quantity	Cost
Adafruit Industries Wire Ferrule Kit		1	$ 17.95
Burley Zydeco Tandem Bicycle		1	$ 300.00
DB9 to M12 Adapter Cable		1	$ 79.01
Eaton-Bussman 15A Fuse		6	$ 6.71
Eaton-Bussman 2A Fuse		10	$ 16.55
Exor JMobile Software		1	$ 350.00
Grainger Steel Sheet	1008 12"x36"x1/8" sheet	1	$ 87.68
Handlebar Grip Set	2 grips per set	2	$ 13.98
IFM Cable		1	$ 154.53

Infinity Tire 26x1	Replacement tires	2	$ 60.00
Phoenix Din Rail		1	$ 8.97
Phoenix End Bracket		2	$ 2.60
Phoenix Fused Terminal Block		6	$ 84.36
Phoenix Power Dist. Block		1	$ 99.05
Pneumatic Quick-Release Plug	Male 1/4" connector	1	$ 3.59
Pneumatic Reduction Coupling	1/4"x1/8"	1	$ 5.99
POR-15 Fuel Tank Sealer		1	$ 17.46
SchmidtProto Custom Reservoir	PETG 3D Print	1	$ 472.68
Sun Hydraulics Check Valve	Free flow, nose-to-side check valve	3	$ 71.82
Sun Hydraulics Coil	24 VDC coil with ISO/DIN 43650, Form A connector with TVS Diode	4	$ 105.64
Sun Hydraulics Solenoid	2-way, solenoid-operated directional poppet valve	4	$ 211.28
Syn Hydraulics Relief Valve	Direct-acting relief valve	2	$ 238.46
TE Connectors		10	$ 3.32
Wago Compact Splicing Connectors		2	$ 13.68
Total			**$ 2,411.63**

Conclusion

The Fluid Power Vehicle Challenge provided exposure to the fluid power field. This allowed us to combine our knowledge from a variety of disciplines learned throughout the honors engineering program, including the following: Fluid Mechanics, Thermodynamics, Heat Transfer, and Controls System Design. By developing our design into a physical product, we also gain more hands-on experience with the machine shop. This gave us a deeper understanding of the flow from design to production, which helped us understand the needs and requirements of operators in a manufacturing setting. The challenge presented by this project of combining human power with hydraulic power provides us with a unique opportunity to think creatively and promote original thinking.

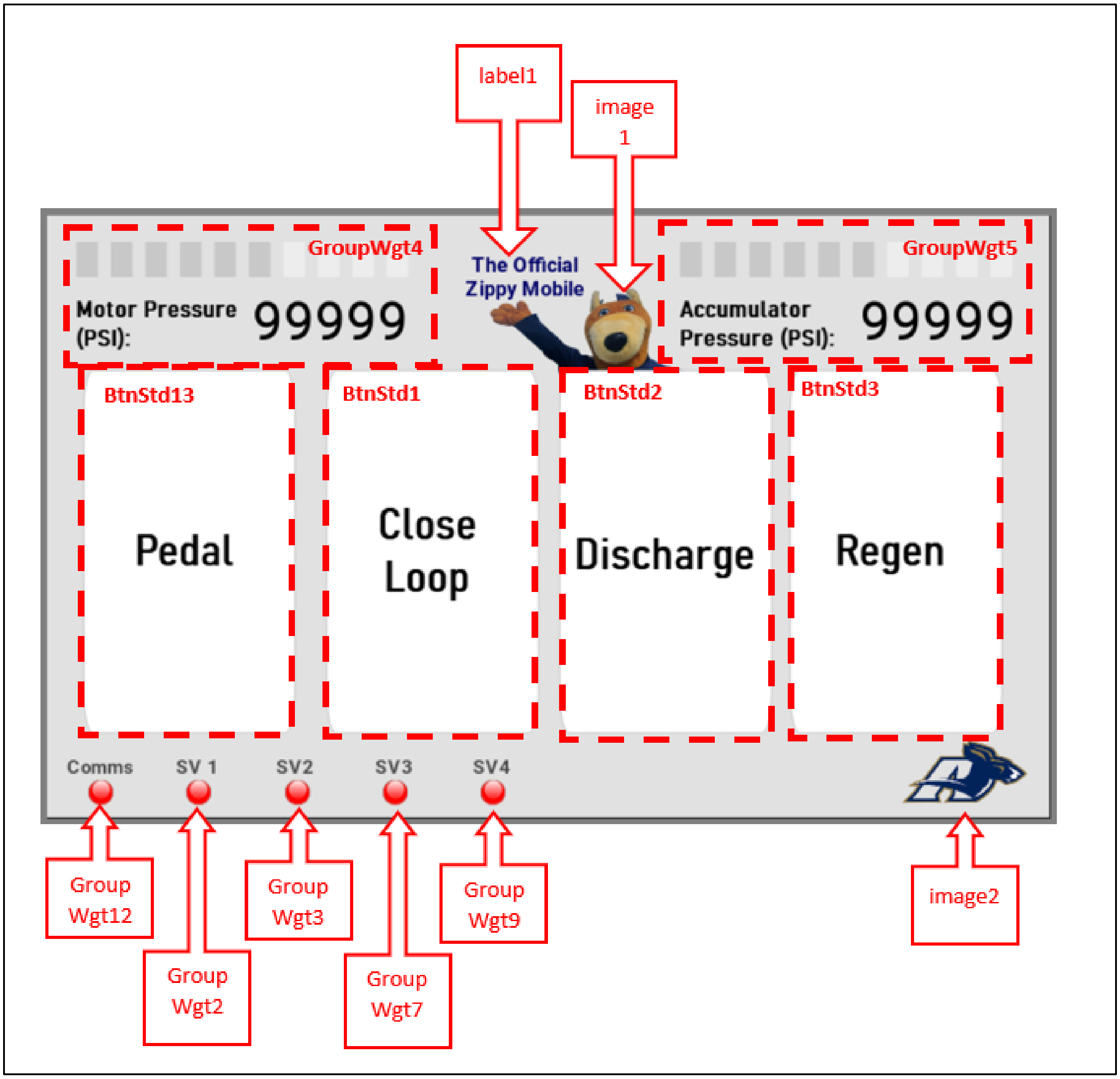
References

Educational Webinars. NFPA Foundation. (n.d.). https://nfpafoundation.org/universities/programs-resources/fluid-power-vehicle-challenge/educational-webinars/#ContentStart

Modbus Organization. Modbus Specifications and Implementation Guides. (n.d.). https://www.modbus.org/specs.php

Appendix

<u>Appendix A: HMI Program</u>



Group : GroupWgt4	
Grid Layout Group	
Enable	**false**
Events	
Bargraph Segmented : GroupWgt4.BargraphSeg1	
Value	**2000**
DataLink	**PLC/p_h_pi1_psi**
Access Type	**R**
Min	**0**
Max	**3000**
Bar Color	**[200, 200, 200]**
DataLink	**PLC/p_h_motor_range|ColorPaletteCustomXForm(1#ffff00,0#00ff00,2#ff0000)**
Access Type	**R**
Background	**[237, 237, 237]**
Reverse	**false**
Events	
Text : GroupWgt4.label1	Motor Pressure (PSI):
Text	**Motor Pressure (PSI):**
Events	
Field : GroupWgt4.numeric1	
Value	**99999**
DataLink	**PLC/p_h_pi1_psi**
Access Type	**R**
Number Format	**Numeric**
Show Thousand Separator	**false**
Decimal Digits	**0**
Leading Digits	**0**
Keypad	**numeric**
Events	

Group : GroupWgt5	
Grid Layout Group	
Enable	**false**
Events	
Bargraph Segmented : GroupWgt5.BargraphSeg1	
Value	**2000**
DataLink	**PLC/p_h_pi2_psi**
Access Type	**R**
Min	**0**
Max	**3000**
Bar Color	**[200, 200, 200]**
DataLink	**PLC/p_h_acc_range|ColorPaletteCustomXForm(0#ffff00,1#00ff00,2#ff0000)**
Access Type	**R**
Background	**[237, 237, 237]**
Reverse	**false**
Events	
Text : GroupWgt5.label1	Accumulator Pressure (PSI):
Text	**Accumulator Pressure (PSI):**
Events	
Field : GroupWgt5.numeric1	
Value	**99999**
DataLink	**PLC/p_h_pi2_psi**
Access Type	**R**
Number Format	**Numeric**
Show Thousand Separator	**false**
Decimal Digits	**0**
Leading Digits	**0**
Keypad	**numeric**
Events	

ButtonsWithLabel : BtnStd13	
Value	0
DataLink	PLC/h_p_pedal_pb|W
Access Type	W
Click Type	momentary
Style	2D
Autorepeat	Disabled
Hold Time (ms)	-1
Label	Pedal

ButtonsWithLabel : BtnStd1	
Value	0
DataLink	PLC/h_p_close_loop_pb|W
Access Type	W
Click Type	momentary
Style	2D
Autorepeat	Disabled
Hold Time (ms)	-1
Label	Close Loop

ButtonsWithLabel : BtnStd2	
Value	0
DataLink	PLC/h_p_discharge_pb|W
Access Type	W
Click Type	momentary
Style	2D
Autorepeat	Disabled
Hold Time (ms)	-1
Label	Discharge

ButtonsWithLabel : BtnStd3	
Value	0
DataLink	PLC/h_p_regen_pb|W
Access Type	W
Click Type	momentary
Style	2D
Autorepeat	Disabled
Hold Time (ms)	-1
Label	Regen

Group : GroupWgt12	
Grid Layout Group	
Enable	false
Events	
Light : GroupWgt12.LgtStd4	
Value	0
Color	rgb(255,0,0);rgb(128,255,128);rgb(255,0,0)
Label	
Show Frame	false
Events	
Text : GroupWgt12.label9	Comms
Text	Comms
Events	

Group : GroupWgt2	
Grid Layout Group	
Enable	false
Events	
Light : GroupWgt2.LgtStd4	
Value	0
DataLink	PLC/p_h_sv1_bool
Access Type	R
Color	rgb(255,0,0);rgb(0,255,0)
Label	
Show Frame	false
Events	
Text : GroupWgt2.label9	SV 1
Text	SV 1
Events	

Group : GroupWgt3	
Grid Layout Group	
Enable	false
Events	
Light : GroupWgt3.LgtStd4	
Value	0
DataLink	PLC/p_h_sv2_bool
Access Type	R
Color	rgb(255,0,0);rgb(0,255,0)
Label	
Show Frame	false
Events	
Text : GroupWgt3.label9	SV2
Text	SV2
Events	

Group : GroupWgt7	
Grid Layout Group	
Enable	**false**
Events	
Light : GroupWgt7.LgtStd4	
Value	**0**
DataLink	**PLC/p_h_sv3_bool**
Access Type	**R**
Color	rgb(255,0,0);rgb(0,255,0)
Label	
Show Frame	**false**
Events	
Text : GroupWgt7.label9	SV3
Text	**SV3**
Events	

Group : GroupWgt9	
Grid Layout Group	
Enable	**false**
Events	
Light : GroupWgt9.LgtStd4	
Value	**0**
DataLink	**PLC/p_h_sv4_bool**
Access Type	**R**
Color	rgb(255,0,0);rgb(0,255,0)
Label	
Show Frame	**false**
Events	
Text : GroupWgt9.label9	SV4
Text	**SV4**
Events	

Image : image2	
Background	**[102, 102, 102]**
Image Path	**images\akron logo.png**

Image : image1	
Background	**[230, 230, 230]**
Image Path	**images\zippy-alumni.png**

Text : label1	
Text	**The Official Zippy Mobile**
Events	

Appendix B: PLC Program

Global Variable List: GVL

```
{attribute 'qualified_only'}
VAR_GLOBAL

    PI1_psi :                REAL ;
    PI2_psi :                REAL ;
    SV1_boolean :            BOOL ;
    SV2_boolean :            BOOL ;
    SV3_boolean :            BOOL ;
    SV4_boolean :            BOOL ;
    Pedal_Mode :             BOOL ;
    Discharge_Mode :         BOOL ;
    Close_Loop_Mode :        BOOL ;
    Regen_Mode :             BOOL ;
    Accumulator_Pressure_Range :  WORD ;
    Motor_Pressure_Range :  WORD ;
    discharge_pb :  BOOL ;
    pedal_pb :  BOOL ;
    close_loop_pb :  BOOL ;
    regen_pb :  BOOL ;
    discharge_lt :  BOOL ;
    pedal_lt :  BOOL ;
    close_loop_lt :  BOOL ;
    regen_lt :  BOOL ;
    test :   BOOL ;

END_VAR
```

NFPA_Bike_Rev05.project
3/23/2024 4:04 PM
Page 1 of 1

POU: HMI_data

```
PROGRAM  HMI_data
VAR

    p_h_pi1_psi :        UINT ;
    p_h_pi2_psi :        UINT ;
    p_h_acc_range :       WORD ;
    p_h_motor_range :     WORD ;
    p_h_sv1_bool :       BOOL ;
    p_h_sv2_bool :       BOOL ;
    p_h_sv3_bool :       BOOL ;
    p_h_sv4_bool :       BOOL ;
    p_h_discharge_lt :   BOOL ;
    p_h_pedal_lt :       BOOL ;
    p_h_close_loop_lt :   BOOL ;
    p_h_regen_lt :       BOOL ;

    h_p_discharge_pb :   BOOL ;
    h_p_pedal_pb :       BOOL ;
    h_p_close_loop_pb :   BOOL ;
    h_p_regen_pb :       BOOL ;

END_VAR
```

1 *TEST bit network to be used for debugging purposes.*

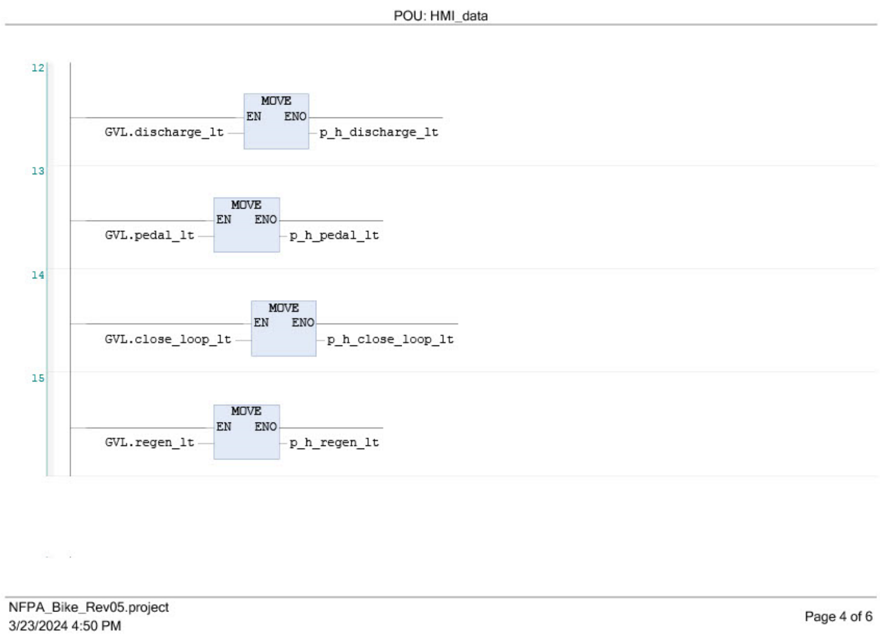
GVL.test
(R)

2 *Next four (4) networks sending analog pressure values to HMI.*
Provision added to allow sending a fixed value to HMI for debugging purposes - this requires turning on the TEST bit.

NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 1 of 6


POU: HMI_data
3
GVL.test
REAL_TO_UINT
EN
ENO
2500.0
p_h_pi1_psi
4
GVL.test
REAL_TO_UINT
EN
ENO
2800.0
p_h_pi2_psi
5
GVL.test
REAL_TO_UINT
EN
ENO
GVL.PI1_psi
p_h_pi1_psi
6
GVL.test
REAL_TO_UINT
EN
ENO
GVL.PI2_psi
p_h_pi2_psi
7
This section of networks sends data from the PLC to the HMI.
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 2 of 6


POU: HMI_data
8
MOVE
EN ENO
GVL.SV1_boolean
p_h_sv1_bool
9
MOVE
EN ENO
GVL.SV2_boolean
p_h_sv2_bool
10
MOVE
EN ENO
GVL.SV3_boolean
p_h_sv3_bool
11
MOVE
EN ENO
GVL.SV4_boolean
p_h_sv4_bool
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 3 of 6


POU: HMI_data
12
MOVE
EN ENO
GVL.discharge_lt
p_h_discharge_lt
13
MOVE
EN ENO
GVL.pedal_lt
p_h_pedal_lt
14
MOVE
EN ENO
GVL.close_loop_lt
p_h_close_loop_lt
15
MOVE
EN ENO
GVL.regen_lt
p_h_regen_lt
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 4 of 6


POU: HMI_data
16
(This network allows multi colors to be displayed at the Accumulator Pressure Bar Graph (Yellow-Green-Red).)
MOVE
EN ENO
GVL.Accumulator_Pressure_Range
p_h_acc_range
17
(This network allows multi colors to be displayed at the Motor Pressure Bar Graph (Yellow-Green-Red).)
MOVE
EN ENO
GVL.Motor_Pressure_Range
p_h_motor_range
18
This section of networks receives data from the HMI to the PLC.
19
MOVE
EN ENO
h_p_discharge_pb
GVL.discharge_pb
20
MOVE
EN ENO
h_p_pedal_pb
GVL.pedal_pb
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 5 of 6

POU: HMI_data

NFPA_Bike_Rev05.project
3/23/2024 4:50 PM

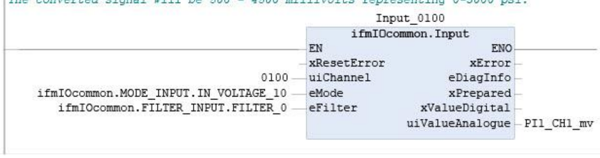
Page 6 of 6

POU: analog_input_PI_1

```
PROGRAM  analog_input_PI_1

VAR

    Input_0100 :  ifmIOcommon . Input ;        // Creating input port (Input_0100) using ifm template
    PI1_CH1_mv :  UINT ;                       // Result of conversion from input signal to tag in mV (unipolar 16 bit
 tag- UINT)
    Input_0101 :  ifmIOcommon . Input ;        // Creating input port (Input_0101) using ifm template. BACK UP PORT
    PI1_CH2_mv :  UINT ;                       // Result of conversion from input signal to tag in mV (unipolar 16 bit
 tag- UINT). BACK UP PORT
    PI1_selected_mv :  UINT ;                  // Value of the higher analog input channel (higher pressure)
    PI1_selected_mv_real :  REAL ;          // Result of conversion from UNIT to float
    mul_1 :  REAL ;                         // Temp tag used in pressure calculation of mV to psi

END_VAR
```

1 *PRIMARY SENSOR CONFIGURATION...*

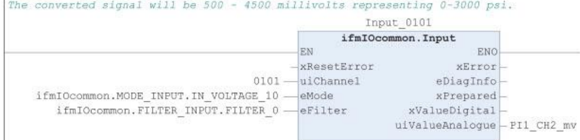
Network configures hardware input IN0100 for 0-10vdc.
Pressure transducer is scaled for 0.5 - 4.5 vdc representing 0-3000 psi.
The converted signal will be 500 - 4500 millivolts representing 0-3000 psi.



POU: analog_input_PI_1

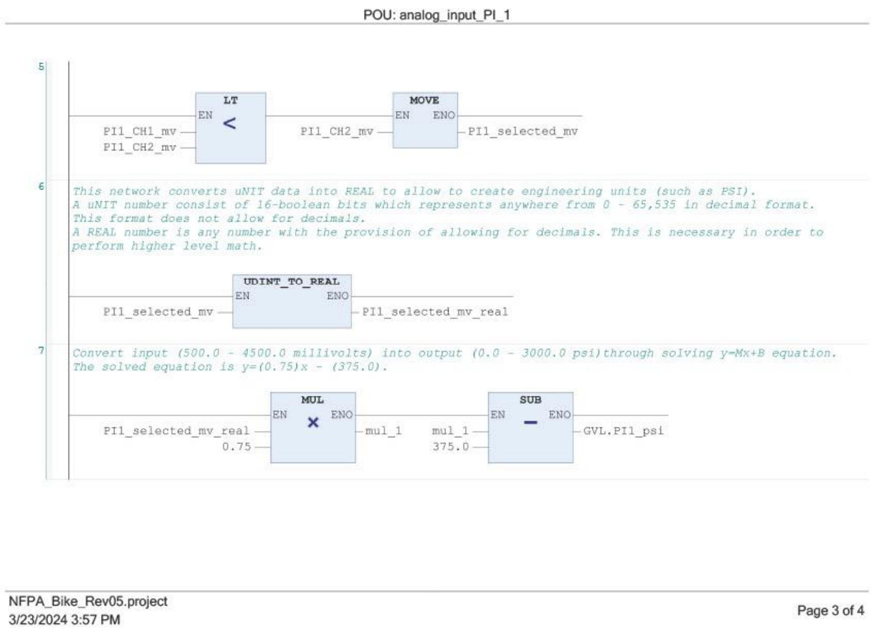
2 *BACKUP SENSOR CONFIGURATION...*

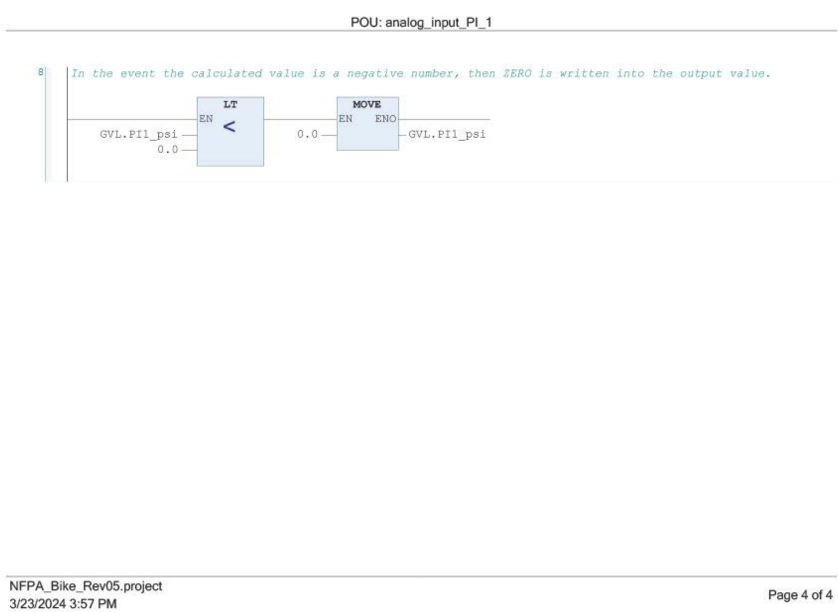
Network configures hardware input IN0100 for 0-10vdc.
Pressure transducer is scaled for 0.5 - 4.5 vdc representing 0-3000 psi.
The converted signal will be 500 - 4500 millivolts representing 0-3000 psi.



3 *The next 2 networks compares which of the 2 previous sensor configuration is greater. The non-active input will read zero (0).*
The greater signal is selected for calculations.

4


POU: analog_input_PI_1
5
LT
EN
<
PI1_CH1_mv
PI1_CH2_mv
MOVE
EN ENO
PI1_CH2_mv
PI1_selected_mv
6
This network converts uNIT data into REAL to allow to create engineering units (such as PSI).
A uNIT number consist of 16-boolean bits which represents anywhere from 0 - 65,535 in decimal format.
This format does not allow for decimals.
A REAL number is any number with the provision of allowing for decimals. This is necessary in order to perform higher level math.
UDINT_TO_REAL
EN ENO
PI1_selected_mv
PI1_selected_mv_real
7
Convert input (500.0 - 4500.0 millivolts) into output (0.0 - 3000.0 psi)through solving y=Mx+B equation.
The solved equation is y=(0.75)x - (375.0).
MUL
EN ENO
×
PI1_selected_mv_real
0.75
mul_1
SUB
EN ENO
mul_1
375.0
GVL.PI1_psi
NFPA_Bike_Rev05.project
3/23/2024 3:57 PM
Page 3 of 4

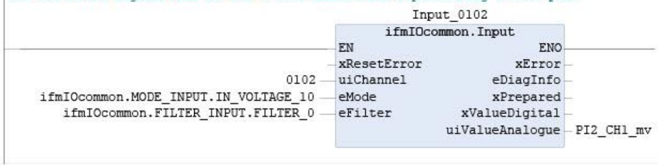

POU: analog_input_PI_1
8
In the event the calculated value is a negative number, then ZERO is written into the output value.
LT
EN
<
GVL.PI1_psi
0.0
MOVE
EN ENO
0.0
GVL.PI1_psi
NFPA_Bike_Rev05.project
3/23/2024 3:57 PM
Page 4 of 4

POU: analog_input_PI_2

```
PROGRAM analog_input_PI_2

VAR

    Input_0102 : ifmIOcommon . Input ;          // Creating input port (Input_0100) using ifm template
    PI2_CH1_mv : UINT ;                     // Result of conversion from input signal to tag in mV (unipolar 16 bit
tag- UINT)
    Input_0103 : ifmIOcommon . Input ;          // Creating input port (Input_0101) using ifm template. BACK UP PORT
    PI2_CH2_mv : UINT ;                     // Result of conversion from input signal to tag in mV (unipolar 16 bit
tag- UINT). BACK UP PORT
    PI2_selected_mv : UINT ;                    // Value of the higher analog input channel (higher pressure
    PI2_selected_mv_real : REAL ;               // Result of conversion from UNIT to float
    mul_1 : REAL ;                          // Temp tag used in pressure calculation of mV to psi

END_VAR
```

1 *PRIMARY SENSOR CONFIGURATION...*

Network configures hardware input IN0102 for 0-10vdc.
Pressure transducer is scaled for 0.5 - 4.5 vdc representing 0-3000 psi.
The converted signal will be 500 - 4500 millivolts representing 0-3000 psi.

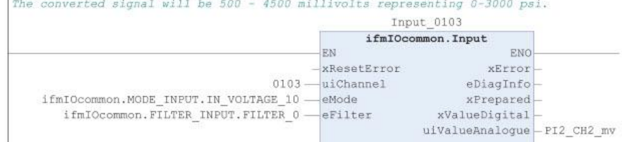


NFPA_Bike_Rev05.project
3/23/2024 3:58 PM
Page 1 of 4

POU: analog_input_PI_2

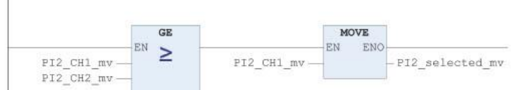
2 *BACKUP SENSOR CONFIGURATION...*

Network configures hardware input IN0103 for 0-10vdc.
Pressure transducer is scaled for 0.5 - 4.5 vdc representing 0-3000 psi.
The converted signal will be 500 - 4500 millivolts representing 0-3000 psi.

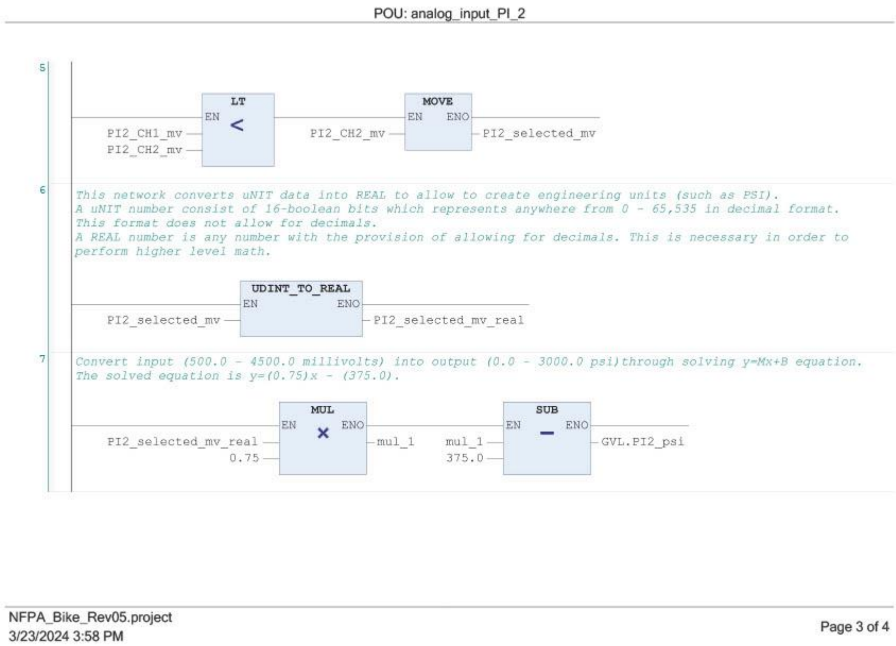


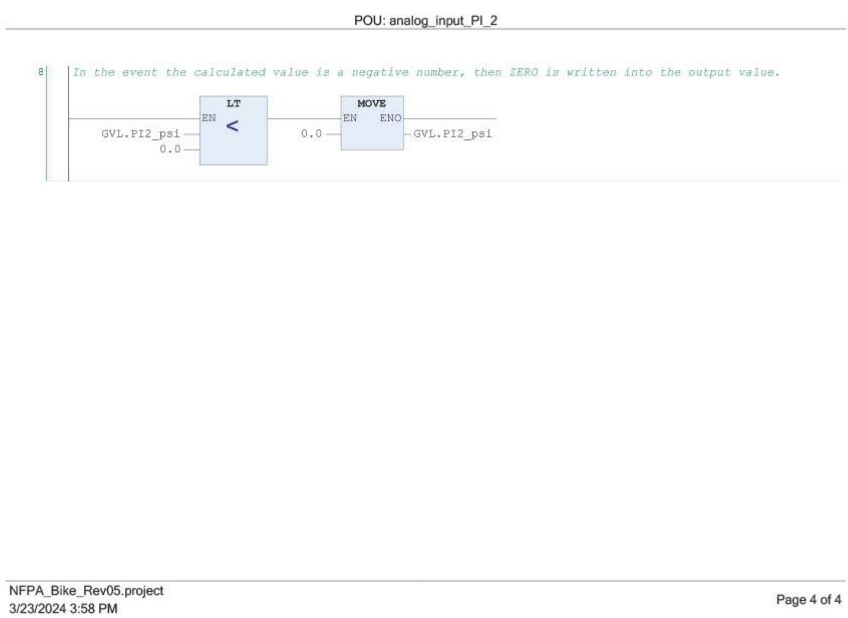
3 *The next 2 networks compares which of the 2 previous sensor configuration is greater. The non-active input will read zero (0).*
The greater signal is selected for calculations.

4



NFPA_Bike_Rev05.project
3/23/2024 3:58 PM
Page 2 of 4


POU: analog_input_PI_2
5
LT
EN
<
PI2_CH1_mv
PI2_CH2_mv
MOVE
EN ENO
PI2_CH2_mv
PI2_selected_mv
6
This network converts uNIT data into REAL to allow to create engineering units (such as PSI).
A uNIT number consist of 16-boolean bits which represents anywhere from 0 - 65,535 in decimal format.
This format does not allow for decimals.
A REAL number is any number with the provision of allowing for decimals. This is necessary in order to perform higher level math.
UDINT_TO_REAL
EN ENO
PI2_selected_mv
PI2_selected_mv_real
7
Convert input (500.0 - 4500.0 millivolts) into output (0.0 - 3000.0 psi)through solving y=Mx+B equation.
The solved equation is y=(0.75)x - (375.0).
MUL
EN ENO
PI2_selected_mv_real
0.75
mul_1
SUB
EN ENO
mul_1
375.0
GVL.PI2_psi
NFPA_Bike_Rev05.project
3/23/2024 3:58 PM
Page 3 of 4


POU: analog_input_PI_2
8
In the event the calculated value is a negative number, then ZERO is written into the output value.
LT
EN
<
GVL.PI2_psi
0.0
MOVE
EN ENO
0.0
GVL.PI2_psi
NFPA_Bike_Rev05.project
3/23/2024 3:58 PM
Page 4 of 4

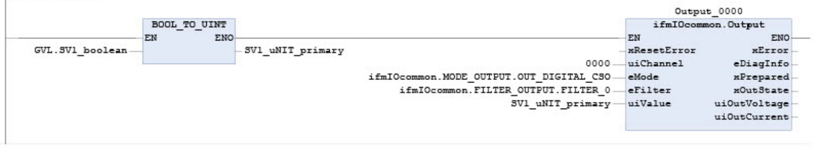
POU: discrete_output_SV_1

```
PROGRAM discrete_output_SV_1

VAR

    Output_0000 : ifmIOcommon . Output ;
    Output_0100 : ifmIOcommon . Output ;
    SV1_uNIT_primary : UINT ;
    SV1_uNIT_backup : UINT ;

END_VAR
```

1

PRIMARY VALVE CONFIGURATION...

Network configures hardware output OUT0000 for simple digital on/off.
The bit is converted from Boolean (0 or 1) into uNIT (0-65,535 integer units) as required by the output function block.

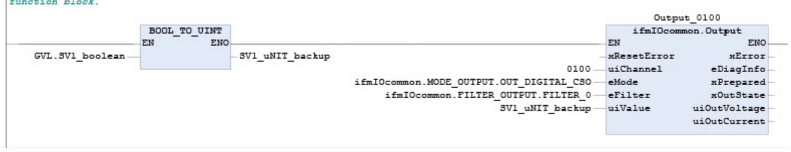


POU: discrete_output_SV_1

2

BACKUP VALVE CONFIGURATION...

Network configures hardware output OUT0001 for simple digital on/off.
The bit is converted from Boolean (0 or 1) into uNIT (0-65,535 integer units) as required by the output function block.



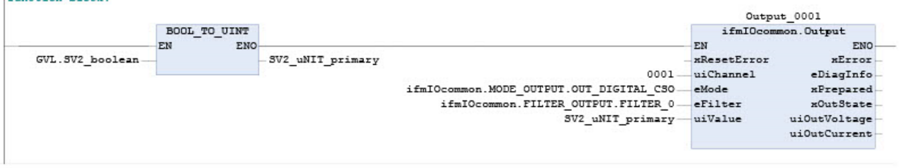
POU: discrete_output_SV_2

```
PROGRAM discrete_output_SV_2

VAR

    Output_0001 : ifmIOcommon . Output ;
    Output_0101 : ifmIOcommon . Output ;
    SV2_uNIT_primary : UINT ;
    SV2_uNIT_backup : UINT ;

END_VAR
```

1

PRIMARY VALVE CONFIGURATION...

Network configures hardware output OUT0000 for simple digital on/off.
The bit is converted from Boolean (0 or 1) into uNIT (0-65,535 integer units) as required by the output function block.

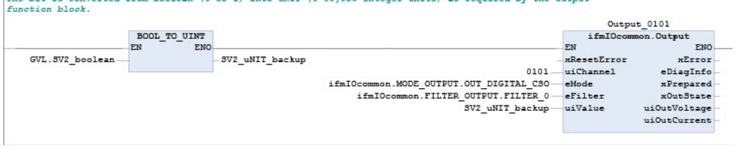


POU: discrete_output_SV_2

2

BACKUP VALVE CONFIGURATION...

Network configures hardware output OUT0001 for simple digital on/off.
The bit is converted from Boolean (0 or 1) into uNIT (0-65,535 integer units) as required by the output function block.



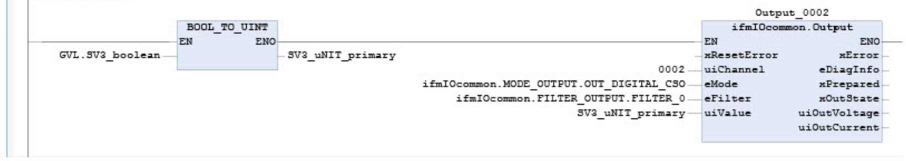
POU: discrete_output_SV_3

```
PROGRAM discrete_output_SV_3

VAR

    SV3_uNIT_primary : UINT ;
    SV3_uNIT_backup : UINT ;
    Output_0002 : ifmIOcommon . Output ;
    Output_0102 : ifmIOcommon . Output ;

END_VAR

```

1

PRIMARY VALVE CONFIGURATION...

Network configures hardware output OUT0000 for simple digital on/off.
The bit is converted from Boolean (0 or 1) into uNIT (0-65,535 integer units) as required by the output function block.



POU: discrete_output_SV_3

2

BACKUP VALVE CONFIGURATION...

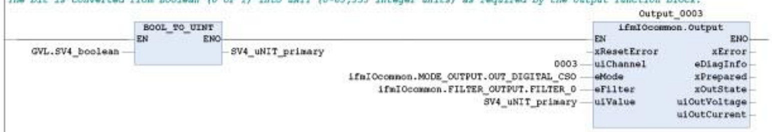
Network configures hardware output OUT0001 for simple digital on/off.
The bit is converted from Boolean (0 or 1) into uNIT (0-65,535 integer units) as required by the output function block.

POU: discrete_output_SV_4

```
PROGRAM discrete_output_SV_4

VAR

    SV4_uNIT_primary : UINT ;
    SV4_uNIT_backup : UINT ;
    Output_0003 : ifmIOcommon . Output ;
    Output_0103 : ifmIOcommon . Output ;

END_VAR

```

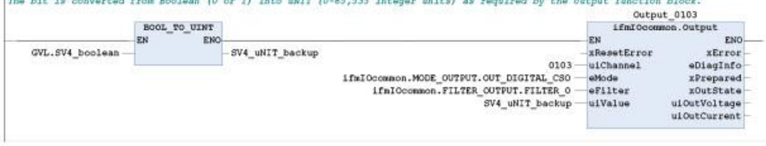
1 *PRIMARY VALVE CONFIGURATION...*

Network configures hardware output OUT0000 for simple digital on/off.
The bit is converted from Boolean (0 or 1) into uNIT (0-65,535 integer units) as required by the output function block.



2 *BACKUP VALVE CONFIGURATION...*

Network configures hardware output OUT0001 for simple digital on/off.
The bit is converted from Boolean (0 or 1) into uNIT (0-65,535 integer units) as required by the output function block.



NFPA_Bike_Rev05.project
3/23/2024 4:49 PM

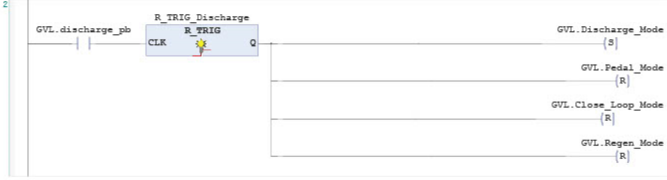
Page 1 of 1

POU: main

```
PROGRAM main
VAR

    Accumulator_Test_Value_psi : REAL ;
    Motor_Test_Value_psi : REAL ;
    R_TRIG_Discharge : R_TRIG ;
    R_TRIG_Pedal : R_TRIG ;
    R_TRIG_Close_Loop : R_TRIG ;
    R_TRIG_Regen : R_TRIG ;

END_VAR
```

1 *The next four (4) networks ensures that only one (1) of the four (4) possible hydraulic mode selections is active at any given moment.*

2

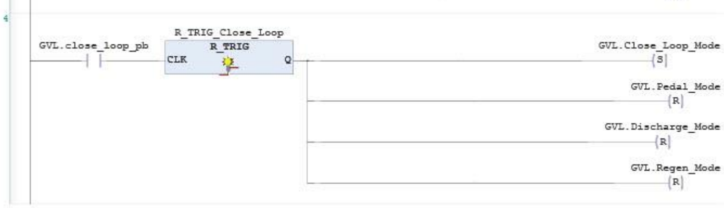


NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 1 of 8

POU: main

3

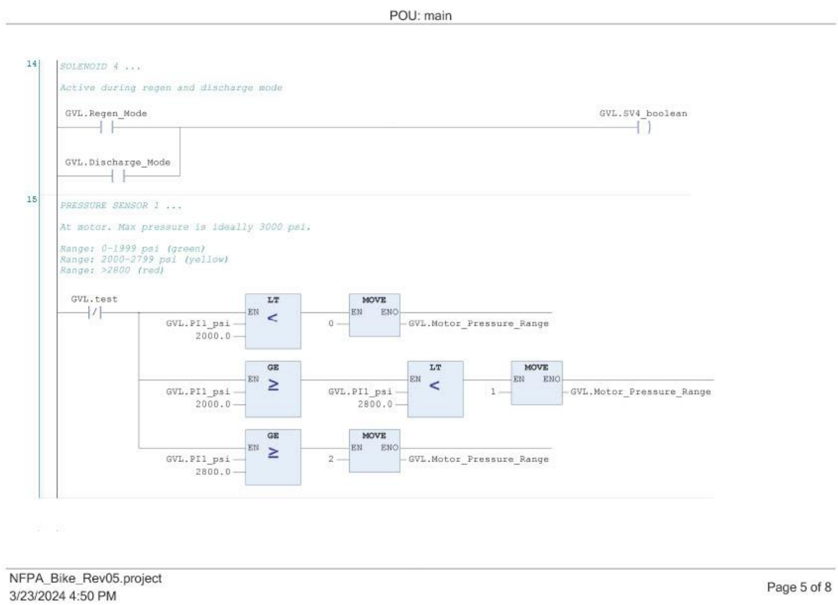
4

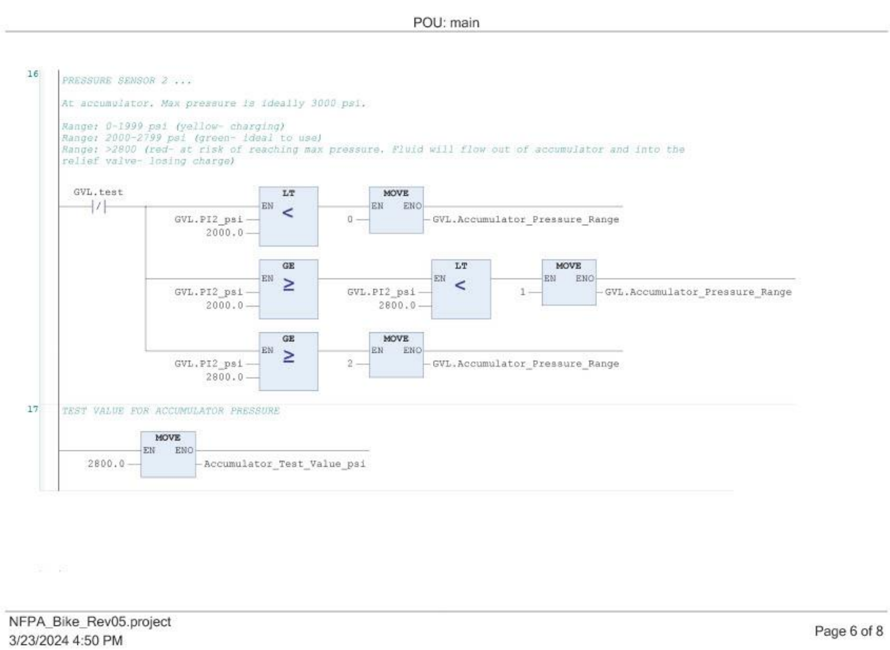


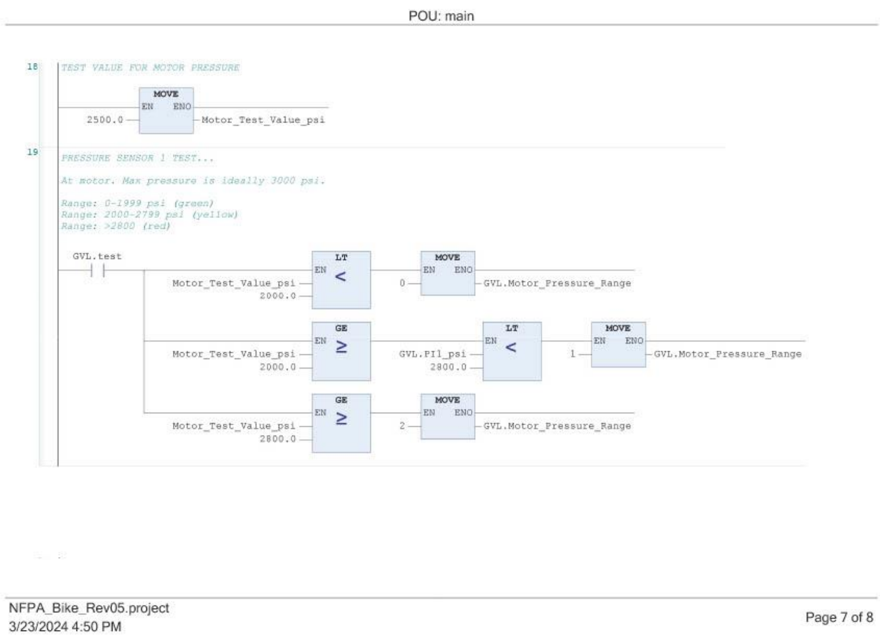
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 2 of 8

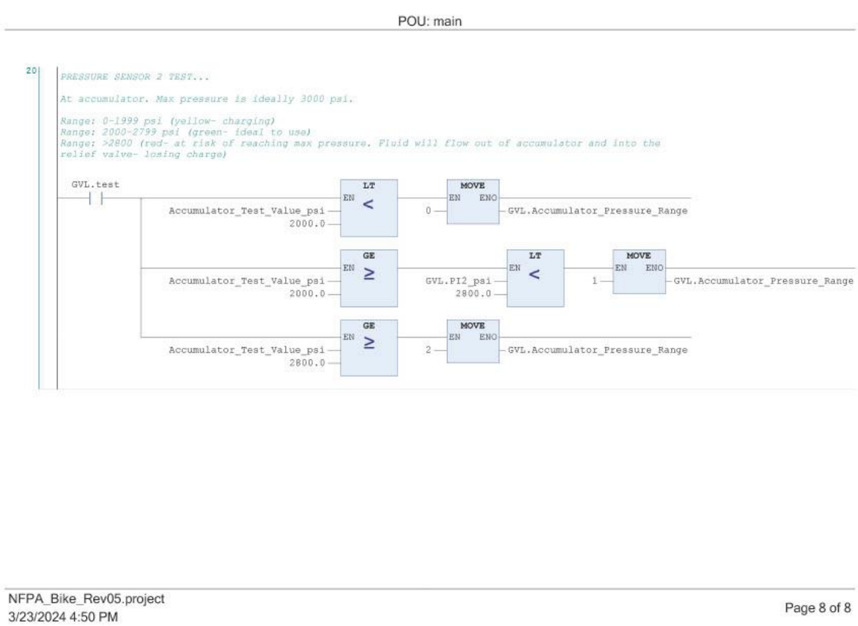

POU: main
GVL.regen_pb
R_TRIG_Regen
R_TRIG
CLK
Q
GVL.Regen_Mode
S
GVL.Pedal_Mode
R
GVL.Discharge_Mode
R
GVL.Close_Loop_Mode
R
The next four (4) sections sends feedback to the HMI regarding which hydrualic mode is active.
GVL.Discharge_Mode
GVL.discharge_lt
GVL.Pedal_Mode
GVL.pedal_lt
GVL.Close_Loop_Mode
GVL.close_loop_lt
GVL.Regen_Mode
GVL.regen_lt
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 3 of 8


POU: main
SOLENOID 1 ...
Active during discharge and closed loop mode
GVL.Discharge_Mode
GVL.SV1_boolean
GVL.Pedal_Mode
SOLENOID 2 ...
Active during closed loop and regen mode
GVL.Close_Loop_Mode
GVL.SV2_boolean
GVL.Regen_Mode
SOLENOID 3 ...
Active during closed loop and discharge mode
GVL.Close_Loop_Mode
GVL.SV3_boolean
GVL.Discharge_Mode
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 4 of 8


POU: main
14
SOLENOID 4 ...
Active during regen and discharge mode
GVL.Regen_Mode
GVL.SV4_boolean
GVL.Discharge_Mode
15
PRESSURE SENSOR 1 ...
At motor. Max pressure is ideally 3000 psi.
Range: 0-1999 psi (green)
Range: 2000-2799 psi (yellow)
Range: >2800 (red)
GVL.test
LT
MOVE
EN ENO
GVL.PI1_psi
2000.0
0
GVL.Motor_Pressure_Range
GE
GVL.PI1_psi
2000.0
LT
GVL.PI1_psi
2800.0
MOVE
1
GVL.Motor_Pressure_Range
GE
GVL.PI1_psi
2800.0
MOVE
2
GVL.Motor_Pressure_Range
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 5 of 8


POU: main
16
PRESSURE SENSOR 2 ...
At accumulator. Max pressure is ideally 3000 psi.
Range: 0-1999 psi (yellow- charging)
Range: 2000-2799 psi (green- ideal to use)
Range: >2800 (red- at risk of reaching max pressure. Fluid will flow out of accumulator and into the relief valve- losing charge)
GVL.test
LT
MOVE
EN ENO
GVL.PI2_psi
2000.0
0
GVL.Accumulator_Pressure_Range
GE
GVL.PI2_psi
2000.0
LT
GVL.PI2_psi
2800.0
MOVE
1
GVL.Accumulator_Pressure_Range
GE
GVL.PI2_psi
2800.0
MOVE
2
GVL.Accumulator_Pressure_Range
17
TEST VALUE FOR ACCUMULATOR PRESSURE
MOVE
2800.0
Accumulator_Test_Value_psi
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 6 of 8


POU: main
18
TEST VALUE FOR MOTOR PRESSURE
MOVE
EN ENO
2500.0
Motor_Test_Value_psi
19
PRESSURE SENSOR 1 TEST...
At motor. Max pressure is ideally 3000 psi.
Range: 0-1999 psi (green)
Range: 2000-2799 psi (yellow)
Range: >2800 (red)
GVL.test
LT
GE
MOVE
Motor_Test_Value_psi
2000.0
2800.0
GVL.PI1_psi
GVL.Motor_Pressure_Range
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 7 of 8


POU: main
20
PRESSURE SENSOR 2 TEST...
At accumulator. Max pressure is ideally 3000 psi.
Range: 0-1999 psi (yellow- charging)
Range: 2000-2799 psi (green- ideal to use)
Range: >2800 (red- at risk of reaching max pressure. Fluid will flow out of accumulator and into the relief valve- losing charge)
GVL.test
LT
GE
MOVE
Accumulator_Test_Value_psi
2000.0
2800.0
GVL.PI2_psi
GVL.Accumulator_Pressure_Range
NFPA_Bike_Rev05.project
3/23/2024 4:50 PM
Page 8 of 8

Appendix C: Modbus

OVERVIEW		
	HMI	**PLC**
IP Address	192.168.82.1	192.168.82.247
Net Mask	255.255.255.0	255.255.255.0
Gateway		192.168.82.21
Modbus # (Unit #)	255	1
Port #		502
watchdog (ms)	5000	OFF
Master Device	YES	NO
Slave Device (Server)	NO	YES
Communication Protocol	Modbus TCP	ModbusTCP_Slave_Device
Also referred as	master / client	slave / server
Addressing	Generic Modbus (0-based) "Starts at address 0"	Start @ address "0" for all data models types.
Initiates all Communication	YES	NO
General notes	Automatically & continuously performs data polling.	Does not initiates any communication to the HMI (sits passively).

FROM PLC TO HMI

IFM PLC								EXOR HMI		
PLC Tag (Global)	PLC Tag (local "HMI Data	Tag name type	Modbus declared Variable	PLC hardware addres	Modbus Input Register	Modbus Addres	Data Flow Path	Modbus Address	Tag name type	Tag name
GVL.PI1_psi	p_h_pi1_psi	word	Application.HMI_data.p_h_pi1_psi	QW52	0.00	300000.00		300000.00	unsigned short	PLC/p_h_pi1_psi
GVL.PI2_psi	p_h_pi2_psi	word	Application.HMI_data.p_h_pi2_psi	QW53	1.00	300001.00		300001.00	unsigned short	PLC/p_h_pi2_psi
		boolean		QW54.00 (QX108.0)	2.00	300002.00		300002.00	boolean	
		boolean		QW54.01 (QX108.1)	2.10	300002.01		300002.01	boolean	
		boolean		QW54.02 (QX108.2)	2.20	300002.02		300002.02	boolean	
		boolean		QW54.03 (QX108.3)	2.30	300002.03		300002.03	boolean	
		boolean		QW54.04 (QX108.4)	2.40	300002.04		300002.04	boolean	
		boolean		QW54.05 (QX108.5)	2.50	300002.05		300002.05	boolean	
GVL.SV1_boolean	p_h_sv1_bool	boolean	Application.HMI_data.p_h_sv1_bool	QW54.06 (QX108.6)	2.60	300002.06		300002.06	boolean	PLC/p_h_sv1_bool
GVL.SV2_boolean	p_h_sv2_bool	boolean	Application.HMI_data.p_h_sv2_bool	QW54.07 (QX108.7)	2.70	300002.07		300002.07	boolean	PLC/p_h_sv2_bool
GVL.SV3_boolean	p_h_sv3_bool	boolean	Application.HMI_data.p_h_sv3_bool	QW54.08 (QX109.0)	2.80	300002.08	>>>	300002.08	boolean	PLC/p_h_sv3_bool
GVL.SV4_boolean	p_h_sv4_bool	boolean	Application.HMI_data.p_h_sv4_bool	QW54.09 (QX109.1)	2.90	300002.09		300002.09	boolean	PLC/p_h_sv4_bool
GVL.close_loop_lt	p_h_close_loop_lt	boolean	Application.HMI_data.p_h_close_loop_lt	QW54.10 (QX109.2)	2.10	300002.10		300002.10	boolean	PLC/p_h_close_loop_lt
GVL.discharge_lt	p_h_discharge_lt	boolean	Application.HMI_data.p_h_discharge_lt	QW54.11 (QX109.3)	2.11	300002.11		300002.11	boolean	PLC/p_h_discharge_lt
GVL.pedal_lt	p_h_pedal_lt	boolean	Application.HMI_data.p_h_pedal_lt	QW54.12 (QX109.4)	2.12	300002.12		300002.12	boolean	PLC/p_h_pedal_lt
GVL.regen_lt	p_h_regen_lt	boolean	Application.HMI_data.p_h_regen_lt	QW54.13 (QX109.5)	2.13	300002.13		300002.13	boolean	PLC/p_h_regen_lt
		boolean		QW54.14 (QX109.6)	2.14	300002.14		300002.14	boolean	
		boolean		QW54.15 (QX109.7)	2.15	300002.15		300002.15	boolean	
GVL.Accumulator_Pressure_Range	p_h_acc_range	word	Application.HMI_data.p_h_acc_range	QW55	3.00	300003.00		300003.00	unsigned short	PLC/p_h_acc_range
GVL.Motor_Pressure_Range	p_h_motor_range	word	Application.HMI_data.p_h_motor_range	QW56	4.00	300004.00		300004.00	unsigned short	PLC/p_h_motor_range

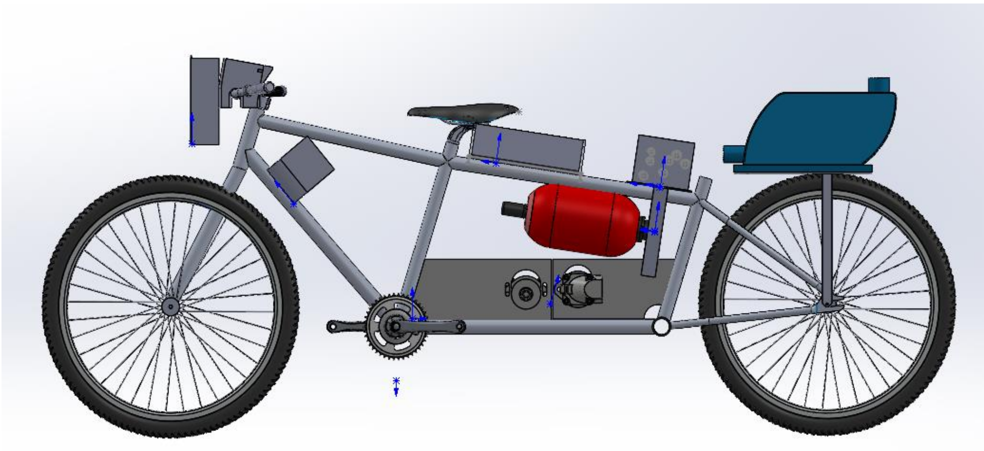
FROM HMI TO PLC

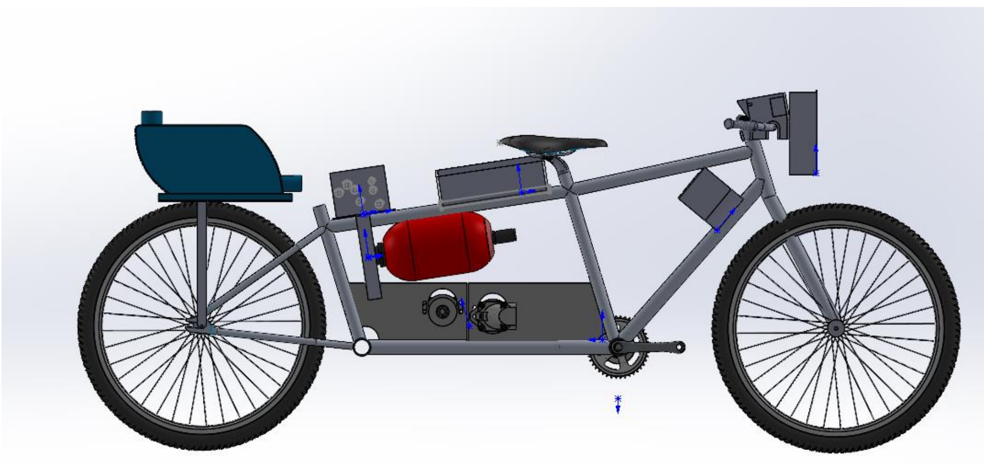
EXOR HMI				IFM Controller					
Tag name	Tag name type	Modbus Address	Data Flow Path	Modbus Address	Modbus Holding Register	PLC hardware address	Modbus declared Variable	PLC Tag (local "HMI Data"	PLC Tag (Global)
PLC/h_p_close_loop_pb	boolean	400000.00		400000.00	0.00	%IW208.00 (%IX416.0)	Application.HMI_data.h_p_close_loop_pb	h_p_close_loop_pb	GVL.close_loop_pb
PLC/h_p_discharge_pb	boolean	400000.01		400000.01	0.01	%IW208.01 (%IX416.0)	Application.HMI_data.h_p_discharge_pb	h_p_discharge_pb	GVL.discharge_pb
PLC/h_p_pedal_pb	boolean	400000.02		400000.02	0.02	%IW208.02 (%IX416.0)	Application.HMI_data.h_p_pedal_pb	h_p_pedal_pb	GVL.pedal_pb
PLC/h_p_regen_pb	boolean	400000.03		400000.03	0.03	%IW208.03 (%IX416.0)	Application.HMI_data.h_p_regen_pb	h_p_regen_pb	GVL.regen_pb
		400000.04		400000.04	0.04	%IW208.04 (%IX416.0)			
		400000.05		400000.05	0.05	%IW208.05 (%IX416.0)			
		400000.06		400000.06	0.06	%IW208.06 (%IX416.0)			
		400000.07		400000.07	0.07	%IW208.07 (%IX416.0)			
		400000.08		400000.08	0.08	%IW208.08 (%IX416.0)			
		400000.09		400000.09	0.09	%IW208.90 (%IX416.0)			
		400000.10	>>>	400000.10	0.10	%IW208.10 (%IX416.0)			
		400000.11		400000.11	0.11	%IW208.11 (%IX416.0)			
		400000.12		400000.12	0.12	%IW208.12 (%IX416.0)			
		400000.13		400000.13	0.13	%IW208.13 (%IX416.0)			
		400000.14		400000.14	0.14	%IW208.14 (%IX416.0)			
		400000.15		400000.15	0.15	%IW208.15 (%IX416.0)			
		400001.00		400001.00	1.00	%IW209			
		400002.00		400002.00	2.00	%IW210			
		400003.00		400003.00	3.00	%IW211			
		400004.00		400004.00	4.00	%IW212			

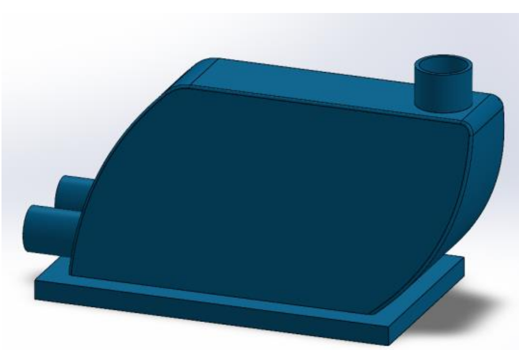
Generic Modbus TCP Address Register Convention			
1st digit starting with:	**Data Model Types**	**Current Address Range (decimal)**	**Editability**
"0"	Coil (Discrete Ouput)	000000-065535	Read-Write
"1"	Discrete Input	100000-165535	Read-Only
"3"	Input Register	300000-365535	Read-Only
"4"	Holding Register	400000-465535	Read-Write
NOTES:			
Each address register is a 16-bits wide ("word" or "short integer")	Example: Address 300009 is the 10th register of the Input Register section		
Within each address register, the bit on the right side is BIT 0. This allows the addressing of Boolean values.	Example: Address 300015:9 is the 10th bit inside the 16th register of the Input Register section (xxxx xx1x xxxx xxxx)		
"For each of the primary tables, the protocol allows individual selection of 65536 data items, and the operations of read or write of those items are designed to span multiple consecutive data items up to a data size limit which is dependent on the transaction function code."	https://www.modbus.org/specs.php		
When performing various Modbus functions, the starting address can be anywhere from 0x000 -0xFFFF (0-65,535). This translates to 65,536 possible addresses.	reference document: Modbus_Application_Protocol_V1_1b3.pdf page 6/50 thru 7/50.		
Not applicable for this discussion, older Modbus versions only allows 10,000 possible addresses.			

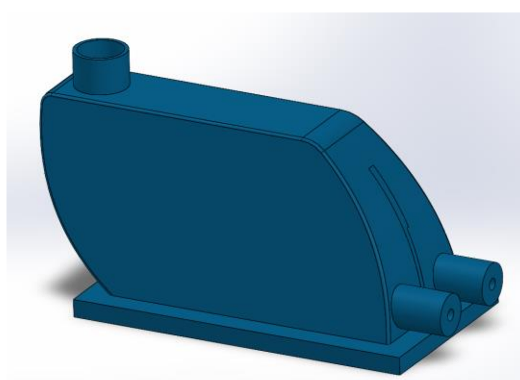
Basic Modbus Functions		
Function Code	**Function**	**Size**
0x01 (01 decimal)	Read Coil	8-bits
0x02 (02 decimal)	Read discrete input	8-bits
0x03 (03 decimal)	Read holding registers	16-bits
0x04 (04 decimal)	Read input registers	16-bits
0x05 (05 decimal)	Write coils	8-bits
0x06 (06 decimal)	Write holding registers	16-bits

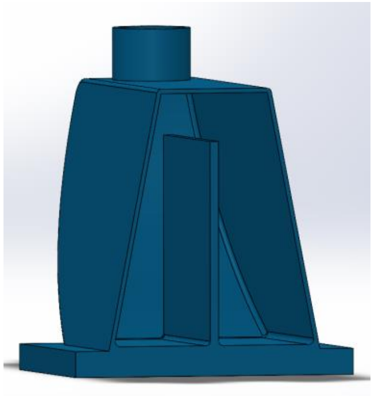
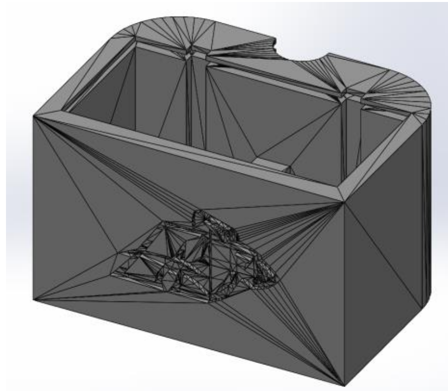
Appendix D: CAD Models:

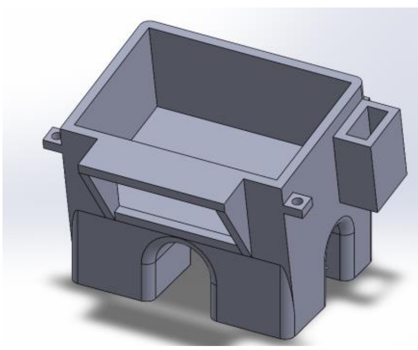
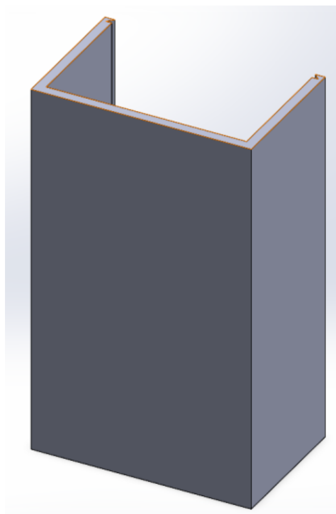
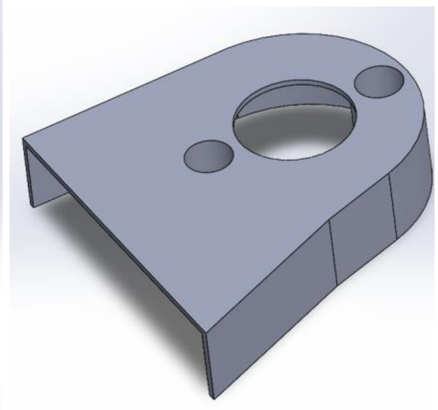


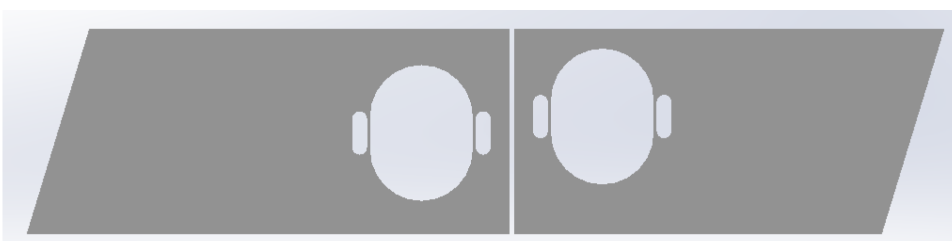




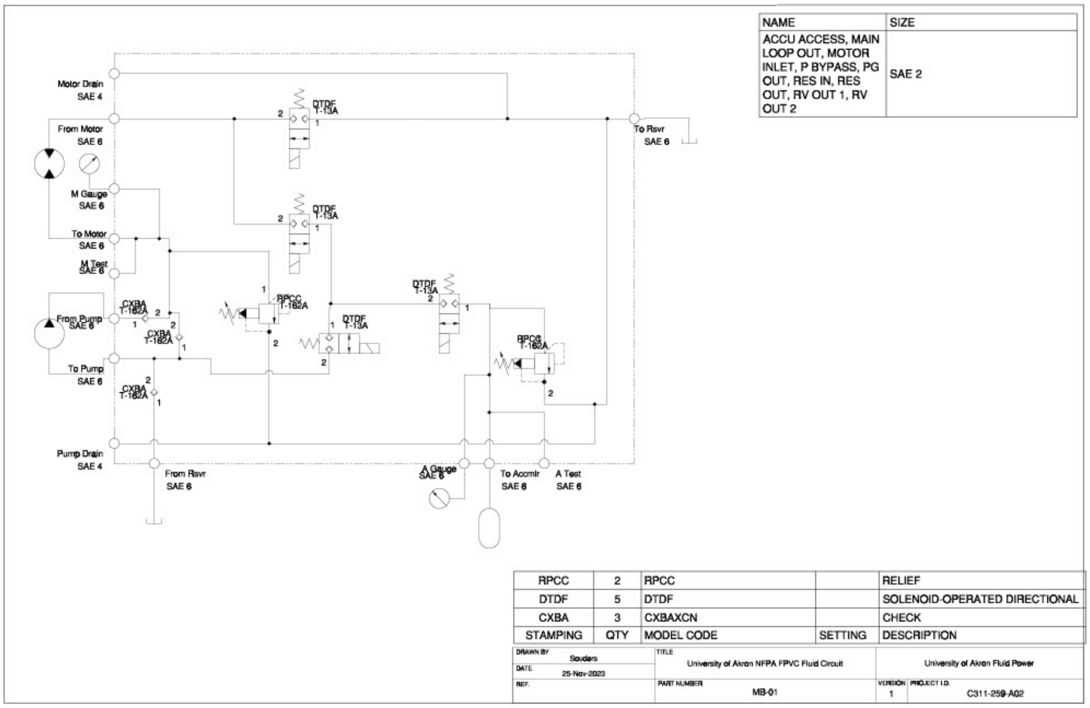


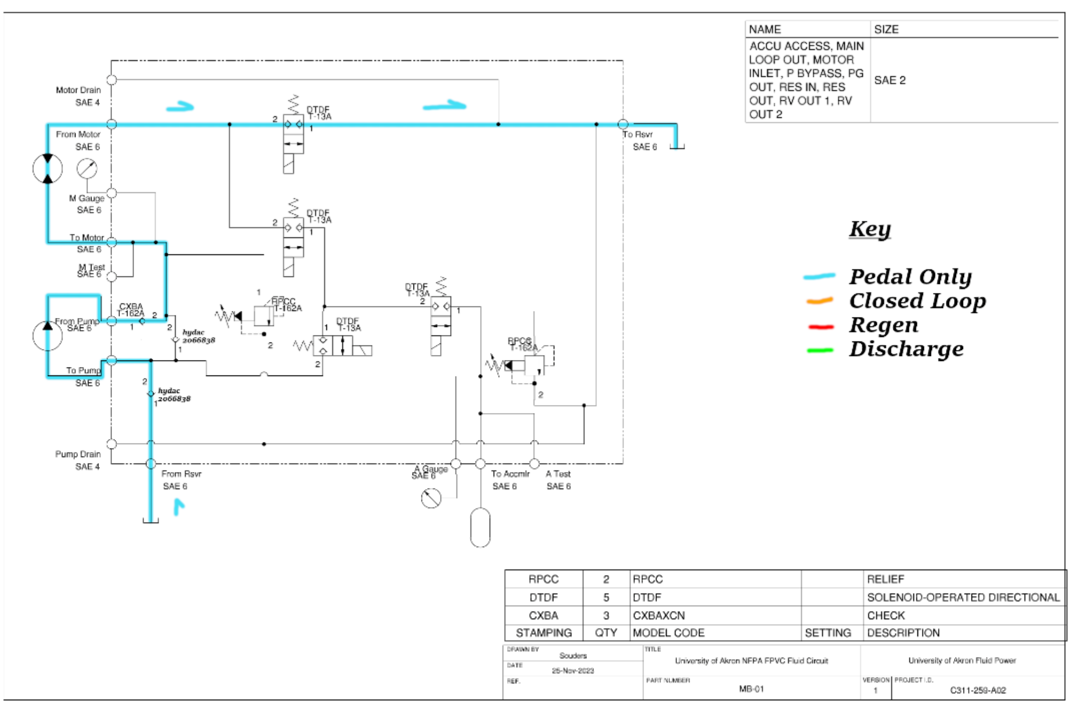



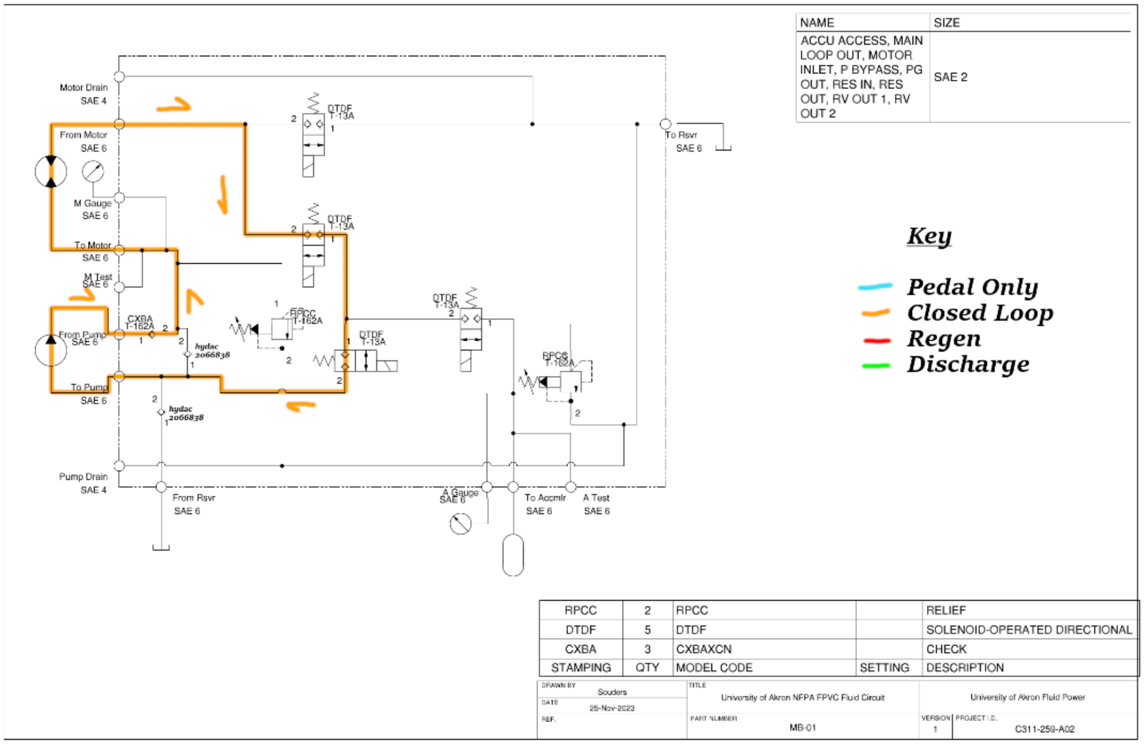




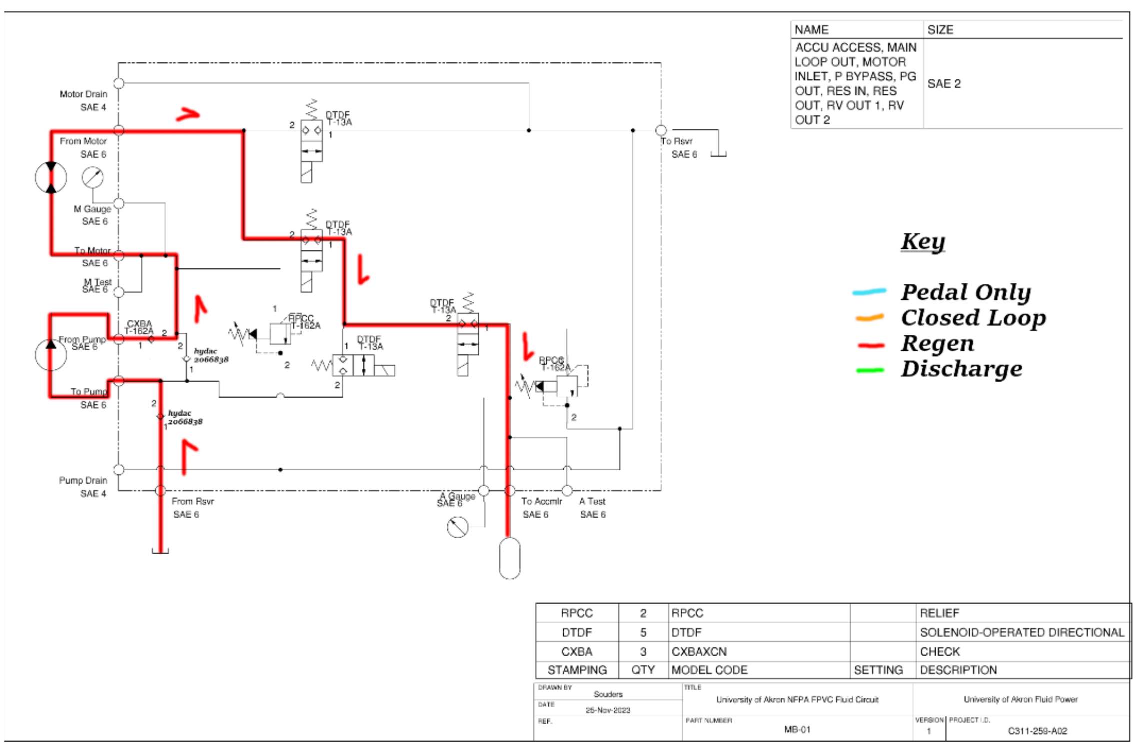


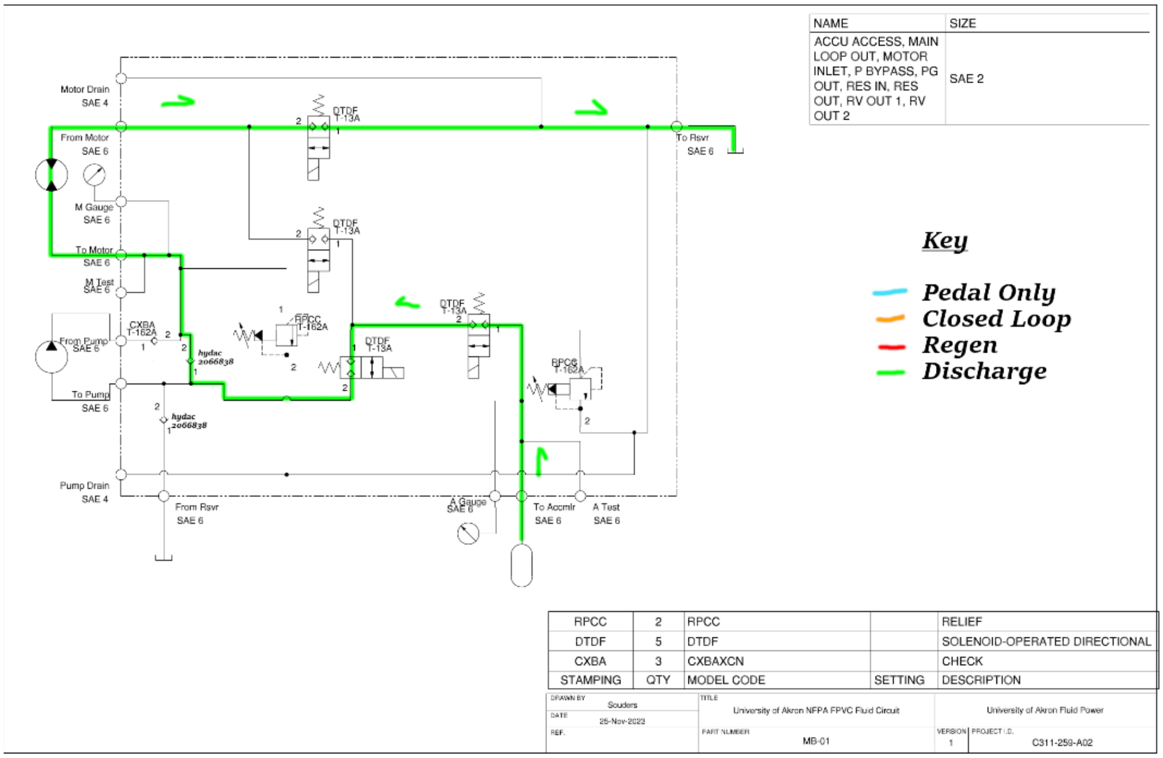
Appendix E: Hydraulic Circuit:

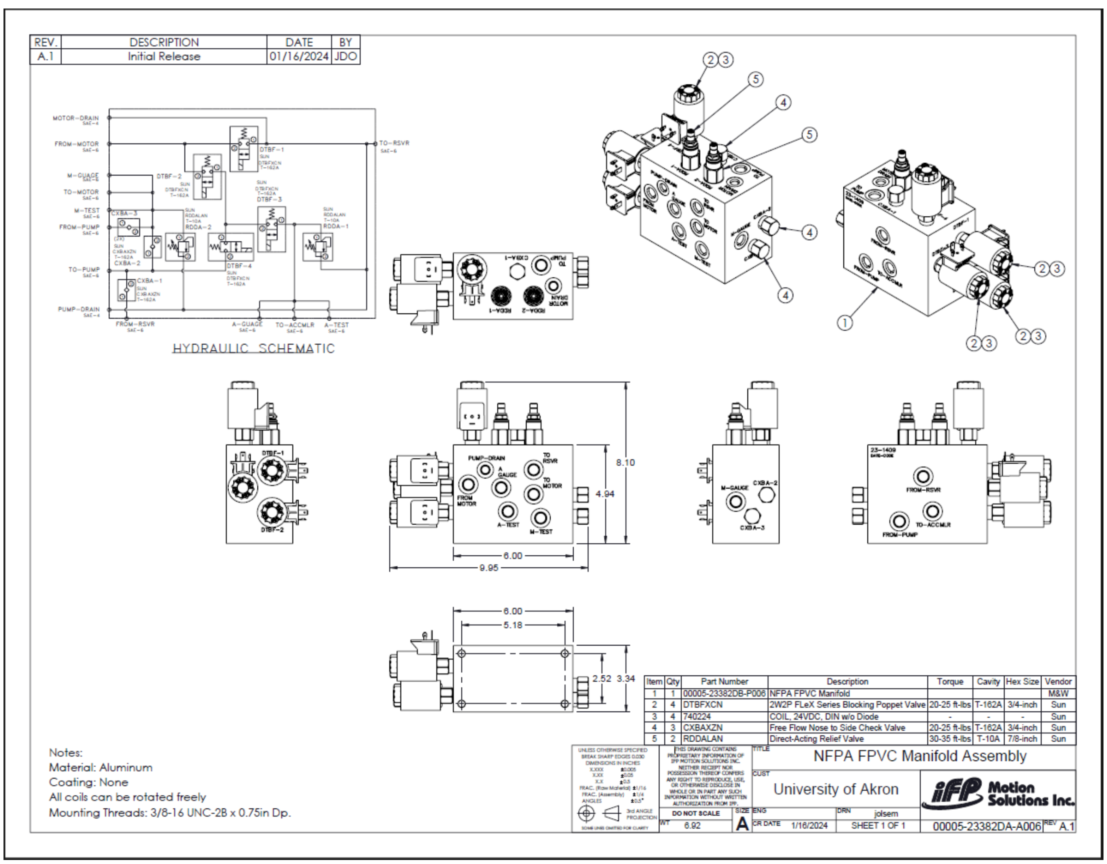





NAME | SIZE
ACCU ACCESS, MAIN LOOP OUT, MOTOR INLET, P BYPASS, PG OUT, RES IN, RES OUT, RV OUT 1, RV OUT 2 | SAE 2
Motor Drain SAE 4
From Motor SAE 6
M Gauge SAE 6
To Motor SAE 6
M Test SAE 6
From Pump SAE 6
To Pump SAE 6
Pump Drain SAE 4
From Rsvr SAE 6
A Gauge SAE 6
To Accmlr SAE 6
A Test SAE 6
To Rsvr SAE 6
DTDF T-13A
CXBA T-162A
RPCC T-162A
hydac 2066838
Key
Pedal Only
Closed Loop
Regen
Discharge
RPCC | 2 | RPCC | | RELIEF
DTDF | 5 | DTDF | | SOLENOID-OPERATED DIRECTIONAL
CXBA | 3 | CXBAXCN | | CHECK
STAMPING | QTY | MODEL CODE | SETTING | DESCRIPTION
DRAWN BY Souders
DATE 25-Nov-2023
TITLE University of Akron NFPA FPVC Fluid Circuit
University of Akron Fluid Power
PART NUMBER MB-01
VERSION 1
PROJECT I.D. C311-259-A02


NAME | SIZE
ACCU ACCESS, MAIN LOOP OUT, MOTOR INLET, P BYPASS, PG OUT, RES IN, RES OUT, RV OUT 1, RV OUT 2 | SAE 2
Motor Drain SAE 4
From Motor SAE 6
M Gauge SAE 6
To Motor SAE 6
M Test SAE 6
From Pump SAE 6
To Pump SAE 6
Pump Drain SAE 4
From Rsvr SAE 6
A Gauge SAE 6
To Accmlr SAE 6
A Test SAE 6
To Rsvr SAE 6
DTDF T-13A
CXBA T-162A
RPCC T-162A
hydac 2066838
Key
Pedal Only
Closed Loop
Regen
Discharge
RPCC | 2 | RPCC | | RELIEF
DTDF | 5 | DTDF | | SOLENOID-OPERATED DIRECTIONAL
CXBA | 3 | CXBAXCN | | CHECK
STAMPING | QTY | MODEL CODE | SETTING | DESCRIPTION
DRAWN BY Souders
DATE 25-Nov-2023
TITLE University of Akron NFPA FPVC Fluid Circuit
University of Akron Fluid Power
PART NUMBER MB-01
VERSION 1
PROJECT I.D. C311-259-A02


NAME | SIZE
ACCU ACCESS, MAIN LOOP OUT, MOTOR INLET, P BYPASS, PG OUT, RES IN, RES OUT, RV OUT 1, RV OUT 2 | SAE 2
Key
Pedal Only
Closed Loop
Regen
Discharge
Motor Drain SAE 4
From Motor SAE 6
M Gauge SAE 6
To Motor SAE 6
M Test SAE 6
From Pump SAE 6
To Pump SAE 6
Pump Drain SAE 4
From Rsvr SAE 6
A Gauge SAE 6
To Accmlr SAE 6
A Test SAE 6
To Rsvr SAE 6
DTDF T-13A
CXBA T-162A
RPCC T-162A
hydac 2066838
RPCC | 2 | RPCC | | RELIEF
DTDF | 5 | DTDF | | SOLENOID-OPERATED DIRECTIONAL
CXBA | 3 | CXBAXCN | | CHECK
STAMPING | QTY | MODEL CODE | SETTING | DESCRIPTION
DRAWN BY Souders
DATE 25-Nov-2023
TITLE University of Akron NFPA FPVC Fluid Circuit
University of Akron Fluid Power
PART NUMBER MB-01
VERSION 1
PROJECT I.D. C311-259-A02


REV. | DESCRIPTION | DATE | BY
A.1 | Initial Release | 01/16/2024 | JDO
HYDRAULIC SCHEMATIC
6.00
9.95
8.10
4.94
5.18
2.52
3.34
Item | Qty | Part Number | Description | Torque | Cavity | Hex Size | Vendor
1 | 1 | 00005-23382DB-P006 | NFPA FPVC Manifold | | | | M&W
2 | 4 | DTBFXCN | 2W2P FLeX Series Blocking Poppet Valve | 20-25 ft-lbs | T-162A | 3/4-inch | Sun
3 | 4 | 740224 | COIL, 24VDC, DIN w/o Diode | - | - | - | Sun
4 | 3 | CXBAXZN | Free Flow Nose to Side Check Valve | 20-25 ft-lbs | T-162A | 3/4-inch | Sun
5 | 2 | RDDALAN | Direct-Acting Relief Valve | 30-35 ft-lbs | T-10A | 7/8-inch | Sun
Notes:
Material: Aluminum
Coating: None
All coils can be rotated freely
Mounting Threads: 3/8-16 UNC-2B x 0.75in Dp.
NFPA FPVC Manifold Assembly
University of Akron
iFP Motion Solutions Inc.
DO NOT SCALE
A
1/16/2024
jolsem
SHEET 1 OF 1
00005-23382DA-A006
REV A.1