Computer Controlled Vegetable Oil Fuel System

by

Joseph W. Souza

Mechanical Engineering Department

California Polytechnic State University

San Luis Obispo

2013

Statement of Confidentiality

The complete senior project report was submitted to the project advisor and sponsor. The results of this project are of a confidential nature and will not be published at this time.

Statement of Disclaimer

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

Table of Contents

List of Figures

List of Tables

Abstract

The desire to utilize alternative fuel sources is strong in many people but it rapidly diminishes as the use becomes more difficult. These alternative fuels can produce similar performance and less dependence on petroleum based fuels. Typically an operator must have knowledge of the differences between the fuels in order to maintain efficient operation of an engine. Compression ignition engines (commonly referred to as diesel engines) can be operated on fuels other than diesel. Vegetable oil is one such viable choice but has some properties that must be considered for optimum performance in engine designed for diesel fuel. Typically the operator of the engine must have a good deal of knowledge of the characteristics of the alternative fuel to avoid undesirable results. The management of an alternative fuel system is given to a microprocessor in this proof of concept. The use of an alternative fuel is greatly simplified and the chance of human error is minimized. This alternative fuel control system is capable of starting on diesel fuel, switching between fuels in operation and shutting down on diesel fuel. The system also has limited self-diagnostic capabilities which detects if a problem is occurring in the alternative oil fuel system and reverts to diesel fuel operation.

Chapter 1 Introduction

The idea for this project was one of several that wandered through my head as I attended the various undergraduate classes necessary for a degree in mechanical engineering at Cal Poly. I had looked forward to the "capstone" project for most of my time at the university. I knew I wanted the project to be based on my study concentration in mechatronics. I also wanted the project to encompass many of the subjects I had studied. There is only written mention of two ideas for a project in my senior design lab notebook. The first was to design and build a metal detecting robot. The second was to design and build an alternative fuel system with automated control. The latter is the project that is contained within this report.

Diesel engines are the most efficient reciprocating engines in common use. They have good reliability and a wide variety of application. Many tons of freight is moved daily by diesel engines. An efficient user-friendly alternative fuel system could easily find a niche in an industry that profits depend heavily on fuel prices. Fuel prices can be managed better with bio-fuels that can be grown on our own soil.

There was an attempt early on in the project to find sponsorship. I mostly needed a truck with a diesel engine to design the alternative fuel system for. PG&E had an interest in donating a retired fleet truck but the university was not accepting any more vehicle projects at that time. I also contacted Caltrans and was referred to a surplus auction site. I quickly realized, as one of the last independent student senior projects, I was going to be self-sponsored.

The main purpose of the endeavor was to design an automated system that simplified the use of an alternative fuel source in a diesel engine. The alternative fuel source would be straight vegetable oil. The oil would be unused so no filtration or other pretreatment specifications were necessary. The reason that straight vegetable oil was selected as opposed to bio-diesel is because the fuel source is available in a usable form to the consumer. At the beginning of the project bio-diesel was not commercially available. At this later date biodiesel is available but costs more because it must be refined from used vegetable oils. The specifications list that was developed for the project became increasing detailed but faced one major obstacle. The engine application was an unknown. In order to continue forward the design was made as universal as possible.

The key specification was that the fuel system had to be managed by a computer. Ideally the operator should have to do nothing to utilize an alternative fuel. The interface with the user should be easy to understand. The vehicle that the system is installed on should have minimum modifications.

The system should not in any way jeopardize the safe operation of the vehicle. The operating temperatures should be consistent with most outdoor temperatures found in the United States. Failure of the alternative fuel system should be limited to that system and not disable the vehicle. The design should be able to be commercially viable and targeted for fleet service. The initial specifications list shown in Table 1 became the core goals for the project.

Table 1. Initial Specifications List

Desired Control Function	• Engine start with ignition key only • Automatic switch between fuels • Engine shutdown with ignition key only • Regulation of tank temperature • Regulation of tank circulation • Self-diagnostic capable
Operator Control & Interface	• One toggle switch for mode • Indication of mode of operation • Indication of fuel type injected • Ignition must operate as manufactured
Vehicle Modification & Operation	• Modification to fuel supply line only • Emergency engine stop • Factory or better filtration standards • Elimination of cross contamination of fuels • Petro-diesel operation when alternative system fails • Operating environment 0°C to 60°C
Commercial Aspects	• Easy to install kit form • Cost effective vehicle modification • Maintain a minimum of 90% of cargo capacity • Fleet service targeted

The specification list eventually became more detailed. The auxiliary fuel tank needed to be about the size of a truck toolbox. The user interface should be about the size of a trailer brake control box so it could be easily attached to the dash. When the vehicle is in operation, the system must use the available 12-14V DC power. Extended parking would allow for 110V AC use but should not need this power during an eight hour work day in normal ambient conditions. The vehicle should also be able to start and stop with the key function as prior to the system installation. The computer would have direct control of the fuel into the engine and also maintain temperature in the tank. One toggle switch would switch between modes of the fuel system. Most of the specifications that went beyond the requirements to make the system work were stripped from the list and left for future improvements.

Chapter 2 Background

The idea of burning vegetable oil in a diesel engine is not new. "The diesel engine can be fed with vegetable oils and would help considerably in the development of agriculture of the countries which use it," is a quote directly from the designer Rudolph Diesel. At the time this project was began there were four companies producing similar products. The companies were Frybrid, Plant Drive, Grease Car and Greasel.

Plant Drive and Grease Car manufactured fuel tanks to facilitate the use of vegetable oil as a fuel. The fuel tanks of both companies provided some method of heating the oil in the tank. No switching system was provided. The purchaser of the tank could use their switching valves or plumb the tank directly into the fuel system. Both Plant Drive and Grease Car require the user to have knowledge of the fuel characteristics to maintain satisfactory performance of the engine. Their products provide only a simplified solution for a heated fuel tank.

Greasel offered a heated fuel tank and a switching valve in their kit. The switching valve is controlled by the user. This kit provides a more complete solution but still requires the user to have knowledge of the correct times to switch fuels.

Frybrid was the only company that offered a complete kit for operating a diesel engine on vegetable oil. The kits came in two forms. The car kit came with a fuel tank. The kit designed for trucks intended for the purchaser to have their own fuel storage method. Switching valves were provided in both kit forms. A small solid state control was used to switch the valves. Engine coolant temperature was sensed and at a set point the system would switch to vegetable oil fuel. There was no consideration for shut-down valve switching in the system. The user would still have to remember to switch the system back to diesel before shutting down in order to have good cold start characteristics in the next use.

The Frybrid system was the closest existing product to the system that was proposed to be designed. Their system was partially automated. No products could be found that featured full automation. The system that was proposed would have a computer control the switching as well as the heating of the vegetable oil.

A patent search was performed. No patents were found that pertained to an automated vegetable oil fuel system for a diesel engine. One patent application was found dated 10-28-1994 for an electronically controlled propane injection system for a diesel engine. Several patents existed for the mixing of diesel with gases or other liquid fuels. Patent 6,287,351 was for an acetylene gas fuel mix.

Patent 3,933,132 contains a diesel mixing injection system. The system that was proposed does not mix vegetable oil and diesel fuel into an intentional blend for combustion. Any mixing that may happen would occur during switching between fuels and is not held to any particular ratio.

It should be noted that California State Vehicle Emissions Standards changed in 2008 to include all 1998 and newer diesel powered trucks and cars of one ton capacity or less. The proposed system visually alters the fuel system of the vehicle it is installed on and may cause a failure of the smog test. The system is legal on all 1998 and older vehicles in the state of California. The system is legal on all model years in the other 49 states.

Chapter 3 Design Development

The conceptual design was well developed at the time the project was proposed. Dr. Maurer, the faculty advisor for the project, recommended that the overall design be reevaluated for two reasons. The first reason was to have experience creating a concept evaluation matrix. The second reason was to verify that the direction the conceptual design had already gone was supported.

There are a number of necessary items that must be included for the fuel system to be functional. For example a fuel tank is necessary for fuel storage in all iterations of the system. The concept evaluation matrix was made to weigh the importance of functionality to such criteria as ease of use, cost and fabrication time. The concept evaluation matrix can be seen in Appendix A. It was no surprise that the configuration of the fuel system that met the minimum functionality of the specification sheet was the best concept. This proved that the already well developed conceptual design was on target.

The translation of concepts to a working design was challenging at first. The vehicle for which the system was to be installed had not been procured at that time. Design work could progress in areas that were not vehicle specific. The fuel storage and fuel flow management could be developed but connections and actual piping would have to wait until the vehicle platform was known. The majority of the initial design work on the fuel system progressed in these areas. The final vehicle procurement occurred with the purchase of a used 1995 Ford F-350 powered by an International/Navistar 7.3L diesel engine. This happened half way through the second quarter of the final design class and did not allow for enough time for rest of the design to be developed. The overall design process since the end of the class has been very linear: moving from one subsystem to another. It is only natural for the summary of this design development to follow the same order.

3.1 Fuel Storage

The fuel tank is a key component to meeting the functional temperature range of 0° Fahrenheit to 140° Fahrenheit specified for the system. The initial design was a simple rectangular tank that held forty-five gallons of vegetable oil. It was intended to be mounted in the bed of the truck wrapped in an inch of foam insulation. The foam was necessary to maintain the temperature in the fuel tank so that in an eight hour work day it would not need an outside power source for heat. It was calculated that in an eight hour work day the tank would lose approximately 16° Fahrenheit with a 70° Fahrenheit temperature difference. Appendix H contains the analysis for the fuel tank heat loss. The foam

insulation was found to be easily deteriorated by sunlight. This meant that the insulation would need a protective shell. The simplest way to do this was to design the tank and its insulation to fit inside of a truck toolbox. A super deep toolbox was purchased. The toolbox inside dimension with a one inch insulation allowance on all sides then changed the capacity of the tank to thirty-two gallons. The fuel tank did not take up the whole storage space of the toolbox and a false bottom provides a volume of usable space above the fuel tank. The fuel tank had to have a heat source that could be plugged into household power if the truck were to be parked outside for lengths of time longer than eight hours in cold weather. The cheapest method of heating was an 110V AC submersion heater with a built in thermostat. This heater was required to be submersed at all times or it could be damaged. The tank design was again modified to provide a well area for the heater element to exist that could not be drained. This feature further reduced the capacity of the tank to twenty-two gallons. The nineteen gallons tank size is similar in capacity to the two diesel tanks that are already in the truck. Their capacities are nineteen and twenty-two gallons. This final iteration of the tank was fabricated from 12 gauge 5052 aluminum. The assembly drawings are found in Appendix G. Figure 1 shows the fabricated vegetable oil tank. Figure 2 shows the vegetable oil tank installed in the insulated toolbox.

Figure 1 Fabricated Vegetable Oil Tank

Figure 2 Vegetable Oil Tank Installed in Insulated Toolbox

One final modification was made to the tank during the final testing phase of the project. The original heater with the built-in thermostat was made of steel. It was threaded into a large aluminum fitting that was welded into the tank. The tank was pressurized to five pounds per square inch to check for leaks after fabrication and there were none. The pressurization was done at room temperature. The final testing phase was the first time the fuel tank was brought up to operating the temperature of 100° Fahrenheit. The difference in thermal expansion created a seep at the heater fitting. This was remedied

by removing the original heater and welding an aluminum plate over the fitting. A new heater was installed in the form of an external heated mat under the tank. This new heater is controlled by a PID heater controller.

3.2 Fluid Circuit

The vegetable oil had to be plumbed to the engine like any other fuel. The first vegetable oil fuel circuit design was quite simple and can be seen in figure 3. It had a pump, filter, heater exchanger and solenoid valves. In this design one pump pressurized the fuel and forced it through a filter and a

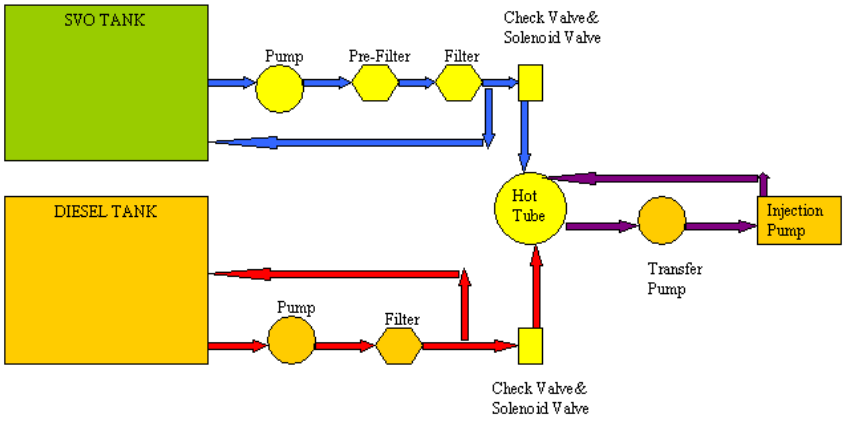


Figure 3 Initial Fluid Circuit Design

heat exchanger to an injection pump. This design would work but the closing of either solenoid valve forces the whole volume of fuel from the pump through the return of the regulator. This would cause the pump to work harder because the increase in volume through the same restriction of the regulator would raise the pressure. Again the fact that the system was not being design for a specific application limited how far development could progress.

An improved fluid circuit was designed when Ford F-350 application was known. The International/Navistar 7.3L is an electronic diesel engine in V-8 cylinder configuration. Each cylinder head has a common fuel rail that is shared by four injectors. The injectors are designed to be supply with approximately forty pounds per square inch of fuel pressure. The diesel fuel pump is a two stage unit. The first stage is a diaphragm pump that produces low pressure to circulate the fuel through the filter and provide a steady fuel supply to the second stage. The second stage is piston pump that pressurizes the fuel to the common rail pressure. The improved vegetable oil supply system mimics the

factory fuel system. Similarly there are two loops with two separate pumps. The low pressure loop operates at about seven pounds per square inch. The main components of the low pressure loop are shown in figure 4. The oil fuel from the tank in the bed enters the filter on the left. An electric heater is

Figure 4 Low Pressure Oil Fuel Loop

mounted in the fuel bowl of the first filter and aides in maintaining tank temperature. A pump pressurizes the oil fuel as it enters a finer micron filter. The pressure regulator on the right manages the pressure to the high pressure loop and the excess volume of fuel is returned to the tank.

The volume flow rates for the pumps were based on the known fuel efficiency of the vehicle. This provided the basis for some approximations for heat exchangers. It was determined that most tube and shell heat exchangers would be too large to affect the desired results. A compact heat exchanger was found to be better for the space constraints of the application. The oil fuel line sizing was made of 3/8" tubing which was one size larger than the ¼" diesel fuel lines. This is acceptable since the viscosity of room temperature diesel fuel and high temperature vegetable oil is similar.

The heated oil had to be plumbed into the fuel injector common rails. This was accomplished by the design of two special fittings. These fittings are shown in figure 5 and figure 6 on the next page. The first fitting shown on the left and was a modified version of the original factory pump outlet pipe. Originally the pump outlet pipe directed high pressure fuel to each of the cylinder head fuel rails via to small steel lines. This fitting was modified in two ways. The first modification was that the two small lines were removed and plugs installed in their location. The second modification was the addition of a large line to direct the fuel up to the valve system. The fitting on the right is a machined cylinder. It has two large lines brazed into one end to receive either high pressure oil or diesel fuel from the valve system. The other end of the cylinder has two small bent steel tubes that are routed to each of the two common injector fuel rails. Figure 7 shows the fittings installed.

Figure 7 Factory Fuel Pump Pull Off

Figure 5 Fuel Mixing Chamber and Feed Lines

Figure 6 Installed Adapter Fittings

The high pressure loop receives the fuel at the low loop pressure and a second pump raises the pressure further. Most of the components of the high pressure loop are shown in figure 8 on the next page. The vegetable oil is first fed into a compact heat exchanger mounted on the engine where it is heated by the coolant of the engine. The compact heat exchanger can be seen in the upper left corner of figure. Three solenoid valves direct the flow of the oil fuel and diesel fuel depending on the desired mode of operation. They are the silver cylinders in the figure on the next page. Two of the valves direct the fuels either into the engine or into a bypass loop. These valves are attached to the blue metal lines in the top of the image. The third valve is found at the bottom of the figure. It directs the fuel returned from the injectors. The diesel fuel system pressure is regulated by the factory pressure regulator. The oil fuel system utilizes a second separate regulator that is set to match the factory pressure

specification. This regulator also regulates diesel fuel pressure when the fuels are being switched. The complete straight vegetable oil fluid circuit diagram is found in Appendix C.

Figure 8 High Pressure Oil Fuel Loop

The initial design had specified two vane type pumps for the pressurization of the oil fuel. Both pumps were capable of the pressure and flow necessary for the system to operate properly. The initial test pressurization of the low loop showed the pump to produce five to seven pounds per square inch. This test run occurred in the afternoon when it was quite warm. The second test pressurization of the low loop is when a problem developed. This test occurred in much cooler morning hours. The pump built pressure momentarily to five pounds per square inch and then fell to two pounds per square inch. The pump was dead headed for an instant with no pressure increase or pump motor strain. The low pressure pump was removed and disassembled. The vanes of the pump were found to not be fully ejected into the eccentric housing. The viscosity of the fuel oil in the low pressure loop pump was slightly too high on cold start-up. The tank is held to 100°F but that is in the tank and approximately twenty inches from the inlet to the low pressure pump.

Both vanes pumps were removed from the design even though the high pressure pump had never been test run. The assumption was that both pumps being the same style would have similar performance in similar conditions. Two gear pumps were then selected and installed. Both high and low pressure loops were able to build necessary pressure in testing.

3.3 Interface Circuit Board and Microprocessor

A microcontroller was needed to manage the functions of the alternative fuel system. Two PolyBot boards using an Atmel Atmega 32 microprocessors were purchased early in the project. These

boards would have done a fair job of controlling the system but it was found to be difficult to configure my personal laptop to consistently download programs to them. The laptop that was dedicated to the programming task also failed early in the project and much of the configuration settings were lost. The lack of support and the eventual time duration the project took on lead to the specification of a new microcontroller board. The Xiphos board was the senior project of electrical engineering major Darron Baida. This board had just become available and was used for the development of this project. It proved to be a great choice because of Darron Baida's support and its ease of use.

The Xiphos board utilizes an Atmel Atmega 1281 microprocessor. It has ten digital pins that can be configured as inputs or outputs as well eight analog inputs. The board also has two motor controllers that are not necessary for this project. There is also a backlight LCD screen for the user interface. I found it to be easily programmable after the initial computer settings were made.

The Xiphos board could not directly connect to the sensors and relays that make up the fuel management system. An interface board was developed to receive the sensor information and convert it to useable data for the Xiphos board. This interface board also handled the control outputs to the real world system. The initial schematic design was hand written in Sharpie on a four foot by six wall covered with banner paper. The hand schematic was later transferred to circuit board design software. All of the components of the interface board are through-hole to ease the hand soldering process. Each component of the interface board had the pad pattern and the component symbol drawn and placed in the schematic. The traces were manually placed to allow the Xiphos board to be mounted directly on top of the interface circuit board.

The interface board design began with the design of three basic circuits. Each of these circuits were constructed on a breadboard and tested for accurate operation in conjunction with the microcontroller board.

The first circuit was designed to read an analog sensor and is diagramed in figure 9. This circuit

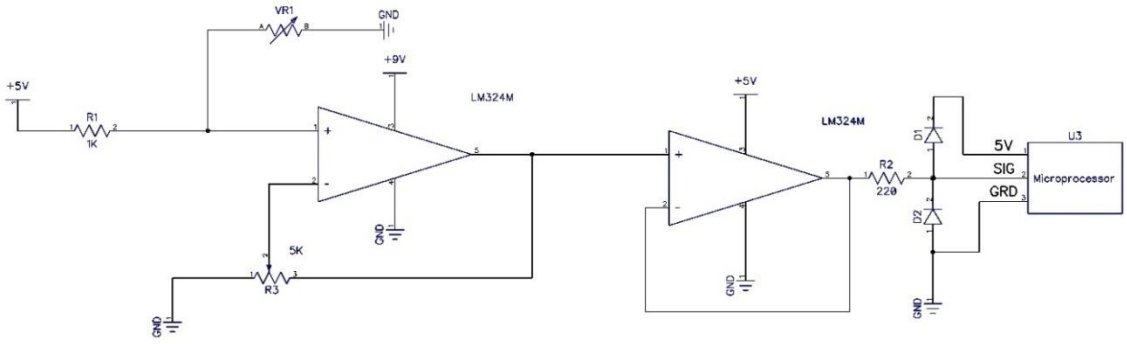


Figure 9 Analog Sensor Read Circuit

consists of two op-amps in series. The first op-amp is has the variable resistance sensor providing the input in a voltage divider arrangement. This op-amp has a variable resistor on the feedback loop to control the gain. The supply power to this op-amp is nine volts which is allows for the sensor signal out of the op-amp to be near five volts. The maximum allowable voltage into the Xiphos board was held to five volts. The second op-amp is used to protect the Xiphos board from sensor shorts or maladjustments of the gain on the first op-amp. This op-amp is arranged to be a voltage follower. It simply copies the voltage in and sends the same voltage out but its supply voltage is five volts so it will not be able to ever put out voltage higher than the maximum allowable for the Xiphos board. Four circuits are designed in this configuration. They are fuel level, tank temperature, injection temperature and injection pressure circuits. A fifth analog input is used in a slightly hybrid fashion. The supply pressure sensor is actually a switch signal read as analog. This was done because there were no more digital pins left for design but several analog pins were left.

The second circuit is designed to take in digital inputs. The circuit shown in figure 10 provides a five volt signal to the microprocessor when the switch is open and zero volts when the switch is closed.

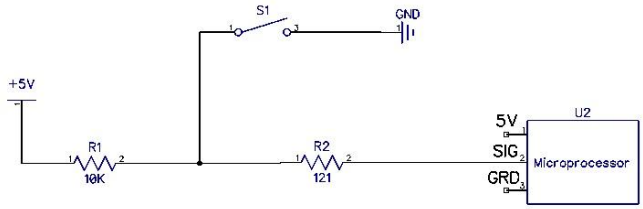


Figure 10 Digital Sensor Read Circuit

A resistor limits the current and serves as protection for the microprocessor. Three inputs use this type of circuit. Those circuits are ignition sense, emergency stop and mode switch.

The third circuit was for the safe control of a twelve volt relay by the microprocessor. The relay, represented as a device in figure 11 on the next page, has a ground signal provided by a transistor. The microprocessor switches the transistor on by applying voltage to the gate through a current limiting resistor. R2 is a pull down resistor and holds the output at zero volts when no signal is present. A diode protects the circuit from voltage spikes as the relay coil circuits are collapsed by directing the excess voltage into the twelve volt supply. A similar grounding transistor was also tied to the same microprocessor output. This secondary transistor was used for the controlled illumination of a light emitting diode on the interface board to verify the control signals. This was to aide in the diagnosis of the system in the final testing stages. Seven relays are controlled in this fashion; diesel solenoid,

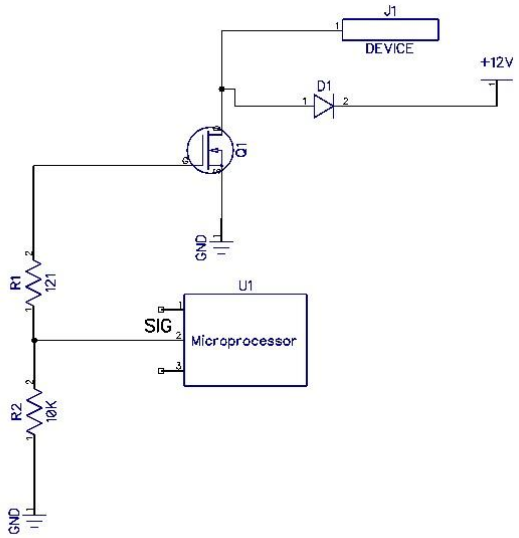


Figure 11 Relay Ground Signal Control Circuit

alternative solenoid, return solenoid, tank heater, high pressure pump, low pressure pump, and automatic shutdown relay.

Once each circuit was verified to be usable, the circuit design was then transfer to the schematic design for the interface board. The interface circuit board was polarity protected and supplies power for the Xiphos board. The full schematic for the interface circuit board can be seen in Appendix D. The printed circuit board layout can be seen in Appendix E. The final circuit board design was sent to a circuit board manufacturer and five two layer boards were made. The components were purchased and assembled. The complete interface circuit board was wired to the Xiphos board and each individual circuit was tested again for functionality. All of the circuits functioned properly.

The next step was to make a bench testing unit capable of simulating sensor inputs. Potentiometers matching the values of the sensor ranges as well as switches for the digital input were wired up. The output values were indicated by the LEDs mounted on the interface circuit board. This provided the working platform for the control code to be developed.

Later the interface circuit board and Xiphos board assembly was mounted in an enclosure in the vehicle. Figure 12 on the next page shows how the two circuit boards were mounted in the enclosure. The LCD display was remotely wired in a different box with the mode switch and emergency shutdown switch. This was done so the remote box for the display and controls could be made much smaller and mounted under the dash. Figure 13 on the next page shows the remote control box in operation.

The only major change to the circuit board design was at the handwritten stage. The fuel level sensor was initially planned to be an off the shelf automotive gauge. Complications arose from having a

Figure 12 Main Control Enclosure

Figure 13 Dash Mount Remote Control in Operation

twelve volt gauge and a five volt limited microprocessor needing to know the fuel level. The solution was to scrap the gauge and use the LCD display to show fuel level. A single sender is used with a five volt reference voltage. The fuel level information is shared between the display and the control matrix.

3.4 Control Program

The control program began with the three basic tasks that would be required to effectively control the fuel system. Those tasks were to read an analog sensor value, read a digital input and control a digital output. Each task was written and verified for correct operation individually.

The goal of the project was to produce an easy to use automated fuel system. This was accomplished by using only one switch on the control unit. The switch allows the user to choose between diesel mode and automatic mode. Diesel mode is running the vehicle on diesel fuel only. Automatic mode enables the alternative fuel system to take control of the fuel supplies. The fuel system was then examined for what it meant in terms of inputs and outputs to be operating in those two modes. The transitions between the modes of operation were also considered. A state space

diagram was made to show how the system needed to operate. See Appendix B for the state space diagram.

The control code was written in a continuous loop. In each loop the inputs are read and the code moves through a series of if statements to select the correct case of a switch command. The first control code design used only three cases for the switch statement. Those cases were diesel operation, diesel operation waiting for alternative conditions to be met and alternative fuel operation. This skeleton structure was labeled FuelControlV1.0 and was not considered functional.

FuelControlV1.1 was the next control code iteration. The sensor reading functions were added to the code with averaging. The code for the fuel level to be displayed on the lcd in the control box was created. A fourth case was added to the switch statement. The four cases were now: (A) Diesel operation, (B) Diesel operation waiting for alternative fuel conditions, (C) Alternative fuel operation and (D) Purging alternative fuel to diesel. The if statements of the continuous loop permitted the movement through the cases. The control moves freely between cases A, B, and C but once the control enters case C it is only allowed to exit through D. FuelControlV1.1 was the first control code that could be bench tested though it was not considered operational. This was because the code did not provide any digital outputs. All of the movement through the control statements was proven by printing the case that the control was in to the LCD screen.

The digital outputs were added in the FuelControlV1.2. There are seven outputs total. Three are for the fuel switching valves and two are for the pumps. One of the remaining outputs is use to control an electric heater in the low pressure fuel loop. The seventh output is used to manage an automatic shutdown relay. This relay is used to hold the ignition on when the key is turned off during the purge state and also to reboot the alternative fuel system microprocessor after each run cycle. FuelControlV1.2 was considered fully functional and was benched test. The operation was verifiable via the light emitting diodes built onto the interface circuit board.

FuelControlV1.3 had only one additional feature not found in the prior version. A self diagnostic aide was created to facilitate installation and setup of the system on an actual vehicle. This was done by displaying the unmet conditions on the LCD display during the control case that is waiting for the proper conditions of alternative fuel operation. The code associated with this feature is not intended to be in a consumer version of the product because the information provided is not necessary for the operator. This information is valuable to the designer in this prototype system. FuelControlV1.3 is the first code to be use in on- vehicle systems testing. This code was never given control of the fuel valves but it was used to verify correct operation of all of sensor reading and output functions.

The final version of the code that the automated alternative fuel system is being controlled by is FuelControl1.4. In this version a small sensing problem was fixed. The alternative fuel system controller must handle two tasks associated with the ignition. The first is to hold the ignition on during an alternative fuel purge. The second requires the ignition to be sensed for whether the key is on or off. There only to ignition feed off of the vehicle ignition switch. One is for the power to the factory engine control systems and the other is for the electrical accessories. The alternative fuel control must sense the accessory power for key position so that is can continue to hold power on to the factory control systems. The factory ignition switch cuts power to the accessories during starting so intermittently the alternative fuel control system would see ignition as off during starting and toggle the ASD control off when it should be on. The code was modified to have the ASD control reset to on any time the control entered the alternative fuel state. This would correct the ASD setting if the ignition had accidently toggled it on start-up. FuelControlV1.4 also contains the correct threshold values for the sensor inputs to maintain efficient operation of the alternative fuel system. A copy of this code is found in Appendix F.

Chapter 4 Results

The 1995 Ford F-350 purchased for this project has had the complete automated vegetable oil fuel system installed. The system meets the majority of the initial specifications. Table 2 summarizes the results of the project as related to the specifications list. The operational temperature range has not been fully tested because ambient conditions over the specified range have not been available. Engine power and mileage remain at approximately the same levels in either mode of operation.

Table 2. Specifications List Results

Specification	Result
Desired Control Function	
• Engine start with ignition key only	Normal key operation
• Automatic switch between fuels	Managed by SVO Control System
• Engine shutdown with ignition key only	Normal key operation, purge timer holds ignition on
• Regulation of tank temperature	Managed by SVO Control System
• Regulation of tank circulation	Managed by SVO Control System
• Self-diagnostic capable	Displays out of operational range data on display
Operator Control & Interface	
• One toggle switch for mode	Mode switch
• Indication of mode of operation	Backlit "Auto" Light
• Indication of fuel type injected	Backlit "Diesel" and "Oil" Light
• Ignition must operate as manufactured	Yes
Vehicle Modification & Operation	
• Modification to fuel supply line only	Both fuel supply and return are modified
• Emergency engine stop	Stop Button removes microprocessor control of ignition
• Factory or better filtration standards	Better- 10 micron & 2 micron filters in series
• Elimination of cross contamination of fuels	Check valves for backflow- purge fuel to vegetable oil tank
• Petro-diesel operation when alternative system fails	Any parameter out of range switches to petro-diesel
• Operating environment 0°C to 60°C	Operated 10°C to 40°C to date
Commercial Aspects	
• Easy to install kit form	Kit form- Requires removal of Turbocharger, somewhat difficult
• Cost effective vehicle modification	Yes, with reasonable vegetable oil source
• Maintain a minimum of 90% of cargo capacity	85% of cargo volume, 100% of eight foot bed length, 95% of load weight
• Fleet service targeted	Diesel smog regulation limits use in California

The truck still easily starts and runs as a diesel. The automated system does not affect the standard operation of the truck even with power disconnected to the system. It has run for a month as only a diesel with the automated system installed with no problems. When the automated system is used it demonstrates seamless switching between diesel fuel and vegetable oil. The only hint that the change has occurred is that the exhaust odor changes slightly. The operator of the vehicle has to only choose which mode of operation he/she desires; the system handles the rest. The concept of an automated alternative fuel control system has been effectively proven by this operational prototype.

Chapter 5 Conclusions and Recommendations

This project has been an incredible journey through the challenges of engineering a complete working prototype. It is a good example of what I believe a capstone project should involve and demonstrates engineering knowledge in a variety of disciplines. The scope, in hindsight, was too large for any individual to complete within the original time constraints. The four subsections of the project described in the prior pages could each have been a project of their own. Regardless, I have been dedicated to completing this task and not wavered from the goal I proposed many years ago.

There are several recommendations I would offer those who might be considering an engine/vehicle project such as this. The first recommendation is to take good stock of your automotive knowledge. I am an ASE certified master mechanic in light duty, heavy duty and alternative fuel vehicle repair and still found this project extremely challenging. The second recommendation is to carefully consider the number of systems that must be designed to create a working prototype. Often in mechatronics projects we are used to wiring components directly to microcontrollers but that is not the case with a vehicle platform. An intermediate set of circuitry is often necessary to work between multiple voltage levels. I completely neglected this in my initial planning stages. The third and final recommendation is to consider the vehicle application carefully. I would recommend a different engine configuration for a proof of concept on a compression ignition engine. The International/Navistar 7.3L has a V-8 cylinder configuration with the majority of the fuel system in the valley pan under the turbocharger. To access the fuel system completely you must remove the turbocharger. This is a lot of extra work for a proof of concept only concerned with the fuel system. An inline six-cylinder configuration would have been easier to work around and would have provided more packaging space for the system.

Appendix A

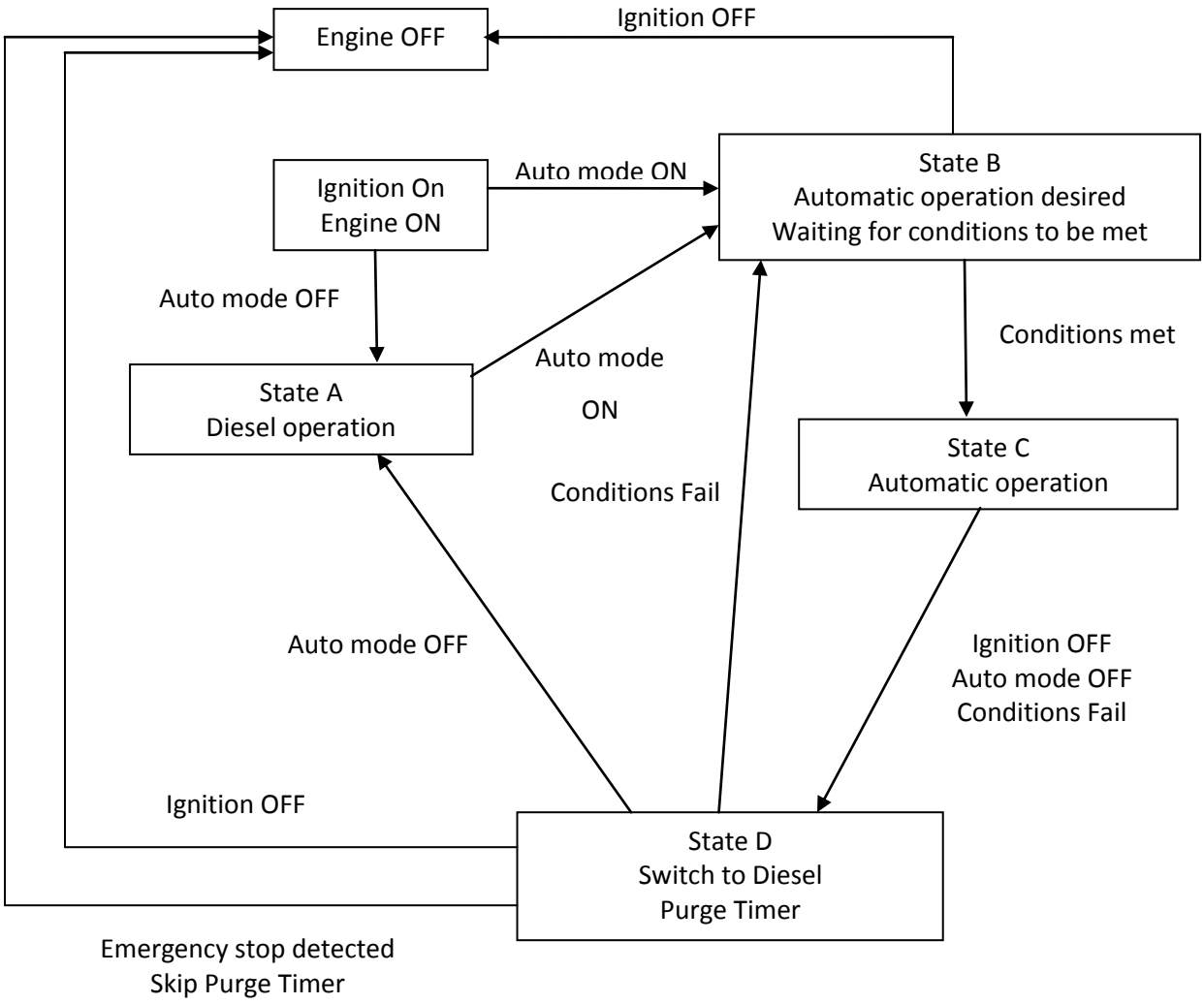
SVO Concept Evaluation Matrix

Concept / Criteria	*Single Tank	*Dual Tank *Indication Only *Manual Switching	*Dual Tank *Tank Heating *Auto Switching	*Dual Tank *Tank Heating *Auto Switching *Preheated Oil Injection	*Dual Tank *Tank Heating *Auto Switching *Auto After Run Purge *Preheated Oil Injection	*Dual Tank *Tank Heating *Auto Switching *Auto After Run Purge *Preheated Oil Injection *Separate Injection System
	1	2	3	4	5	6
A. Ease of Use	+	-	D	S	+	+
B. Power	-	-		+	+	+
C. Cold Weather Feasibility	-	-	A	+	+	+
D. Emissions	-	-		+	+	+
E. Cost	+	+	T	-	-	-
F. Fabrication	+	S		-	-	-
G. Assembly	+	S	U	S	S	-
Sum +	4	1		3	4	4
Sum -	3	4	M	2	2	3
Sum Total	1	-3		1	2	1

Appendix B

State Space Diagram

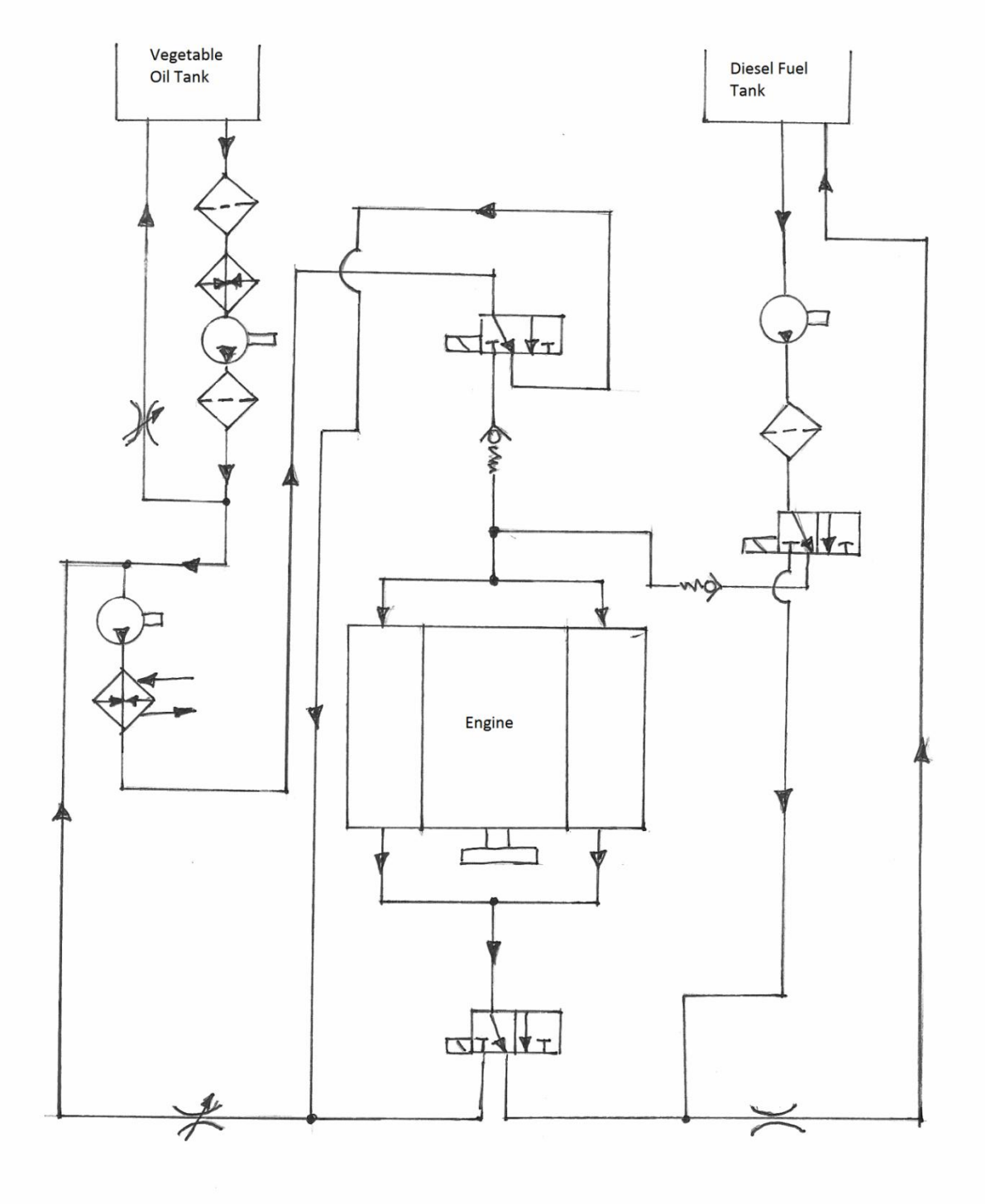
SVO Control


Engine OFF
Ignition OFF
Ignition On
Engine ON
Auto mode ON
State B
Automatic operation desired
Waiting for conditions to be met
Auto mode OFF
Conditions met
Auto mode
ON
State A
Diesel operation
State C
Automatic operation
Conditions Fail
Auto mode OFF
Ignition OFF
Auto mode OFF
Conditions Fail
Ignition OFF
State D
Switch to Diesel
Purge Timer
Emergency stop detected
Skip Purge Timer

Appendix C

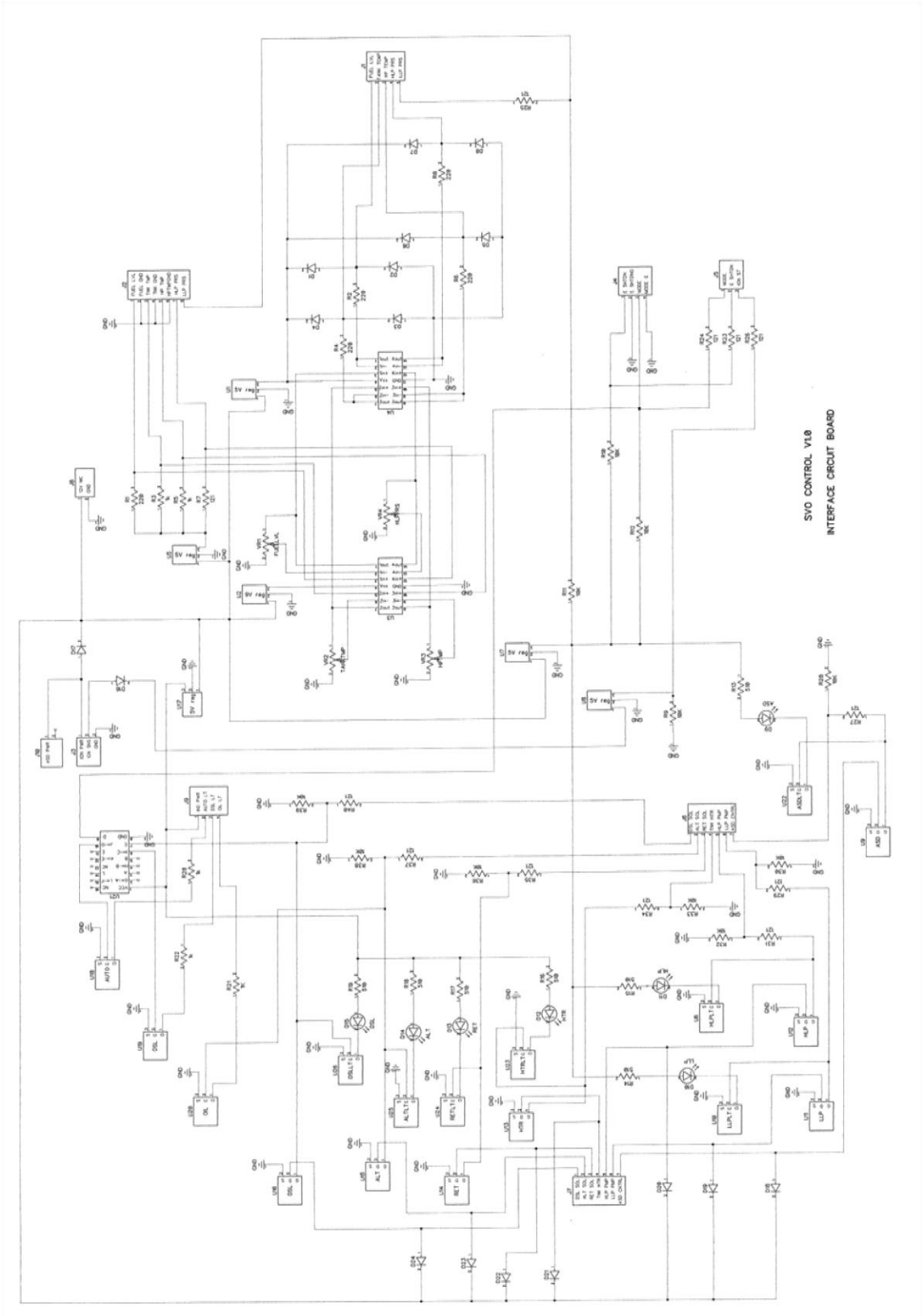
Fluid Circuit Schematic

Fluid Circuit Schematic



Appendix D

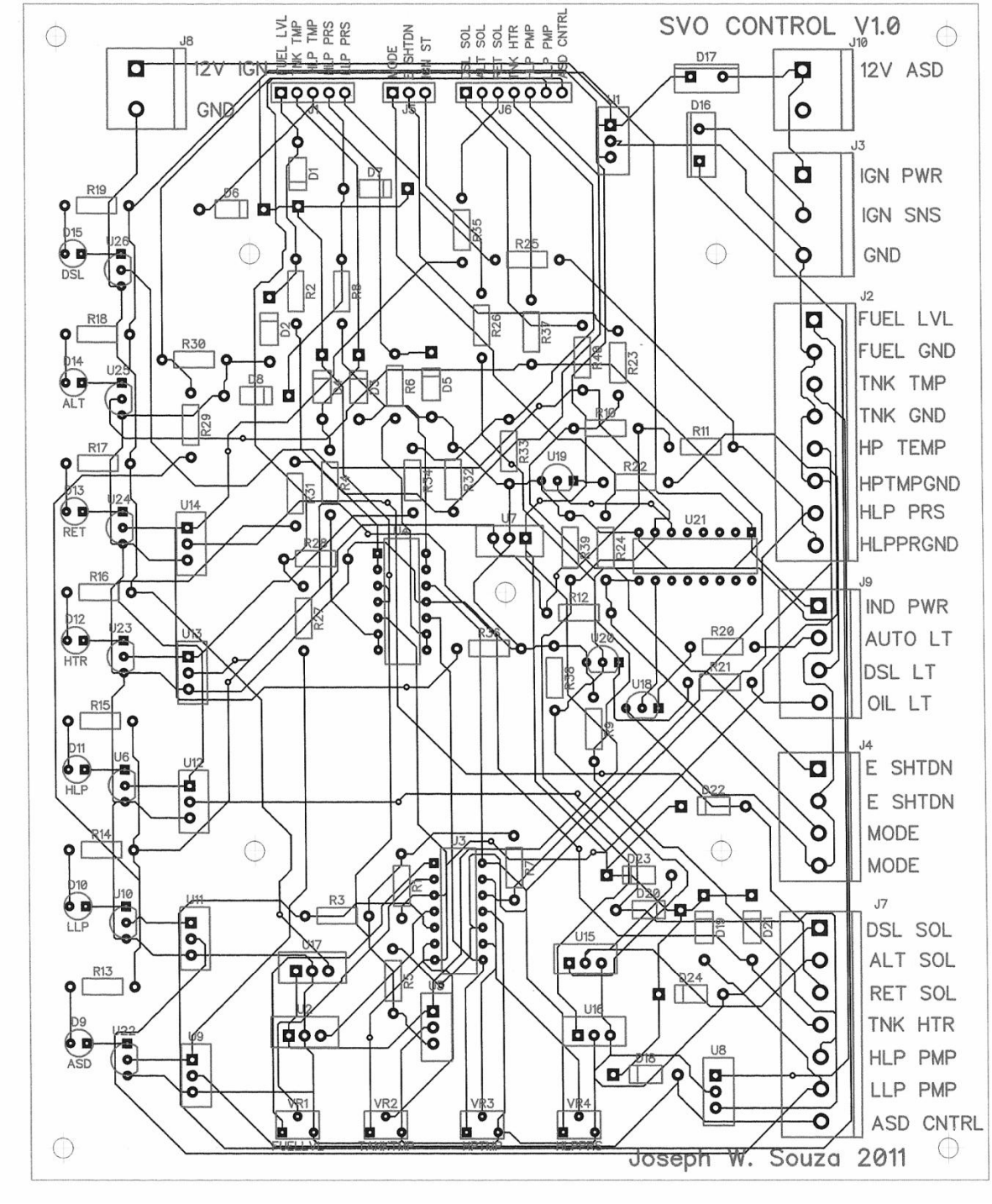
Interface Circuit Board Schematic


SVO CONTROL V1.0
INTERFACE CIRCUIT BOARD

Appendix E

Interface Circuit Board Layout

SVO CONTROL V1.0 INTERFACE PCB



Appendix F

Fuel Control V1.4 Control Code

```
//Fuel Test

#include "globals.h"


/*Joseph W Souza
 SVO Fuel System
 3-7-2012
 Program:        FuelControlV1.4
 Input:          User controls and system Status.
 Output:         Switch statement control for desired operation. Screen displays
                          information for user.
 Description:    This program is the fifth interation of the program.  It is fully
                          functional and has an extra self diagnostic feature to aid in
                          the preliminary set up of the system.  Thresholding values have been set
                          to better represent the actual operating state of the actual system.

 Revision History:
                 4/16/12          Joseph W. Souza          FuelControlV1.1
                                                           Addition of GETFUELLEVEL function
                                                           Addition of FUELDISPLAY function
                                                           Fuel Display divisions can only be modified
                                                                   from within the function.

                 4/18/12          Joseph W. Souza          Addition of GETTANKTEMP function
                                                           Addition of GETHPLOOPTMP function
                                                           Addition of GETHIGHLOOPPRES function
                                                           Addition of GETLOWLOOPPRES function
                                                           Addition of CHECKSTATUS function

                 5/12/12          Joseph W. Souza          Addition of RUNPUMPS function
                                                           Addition of OFFPUMPS function
                                                           Addition of HEATERCONTROLON function
                                                           Addition of HEATERCONTROLOFF function
                                                           Addition of DIESELCONFIG function
                                                           Addition of ALTFUELCONFIG function
                                                           Addition of PURGEFUELCONFIG function

                 5/13/12          Joseph W. Souza          Addition of INTRO function

                 5/16/12          Joseph W. Souza          Addition of WAITCONDITION function

                 4/28/13          JOSEPH W. Souza          Reset threshold value based on steady state operation

                 5/3/13           JOSEPH W. SOUZA          Addition initializing output states

                 5/4/13           JOSEPH W. SOUZA          Addition of two functions for commanding ASD on & off
                                                                   Function is commanded from within state.
                                                                   Fix for ign sense intermittently being caught off on a
                                                                   long crank cycle.

          */


#include "globals.h"

void SHOWINTRO(int FLetterDelay);

int GETFUELLEVEL(int FSampleSize);
int GETTANKTEMP(int FSampleSize);
int GETHPLOOPTMP(int FSampleSize);
int GETHIGHLOOPPRES(int FSampleSize);
int GETLOWLOOPPRES(int FSampleSize);

char CHECKSTATUS(int FFuelLvlAvgST, int FTnkTmpAvgST, int FHpTmpAvgST,  int FHlpPrsAvgST, int FLlpPrsAvgST, int FFuelLvlLowST,
                          int FTnkTmpLowST, int FHpTmpLowST, int FHlpPrsLowST, int FLlpPrsLowST);
```

```
void WAITCONDITIONS(int WFuelLvlAvg, int WTnkTmpAvg, int WHpTmpAvg, int WHlpPrsAvg, int WLlpPrsAvg, int WFuelLvlLow,
                    int WTnkTmpLow, int WHpTmpLow, int WHlpPrsLow, int WLlpPrsLow, int WWaitDsplyTime);

void FUELDISPLAY(int FFuelLvlAvg, int FFuelDsplyDur);

void RUNPUMPS(void);
void OFFPUMPS(void);
void ASDOFF(void);
void ASDON(void);
void HEATERCONTROLON(int FTnkTmpAvgHC, int FTnkHtrLowLimit, int FTnkHtrHighLimit);
void HEATERCONTROLOFF(void);
void DIESELCONFIG(void);
void ALTFUELCONFIG(int FValveDelay);
void PURGEFUELCONFIG(int FPurgeTime);

int main()
{
        int FuelLvlAvg;                                                         /*Analog inputs*/
        int TnkTmpAvg;
        int HpTmpAvg;
        int HlpPrsAvg;
        int LlpPrsAvg;

        int Mode;                       /*mode 1 =diesel 0=auto*/               /*Digital inputs*/
        int EShtdn;                     /*0=standard 1=momentary*/
        int IgnSt;

        int DslSol;                                                             /*Digital outputs*/
        int AltSol;
        int RetSol;
        int TnkHtr;
        int HlpPmp;
        int LlpPmp;
        int AsdCntrl;

        char action;                            /*'A' for diesel, 'B' for alt. waiting, 'C' for alt. running*/
        char status;                            /*'w' for waiting or 'r' for ready*/


        initialize();

        int WaitDsplyTime=2000;                 /*Delay in milliseconds between WaitCondition items*/
        int LetterDelay=160;                    /*Delay in milliseconds between letters at intro*/
        int ValveDelay=20;                      /*Slight delay on return valve switching in milliseconds.*/
        int PurgeTime=180;                      /*Fuel System purge time in seconds.*/
        int FuelDsplyDur=2000;                  /*Fuel display on screen time duration in milliseconds.*/
        int SampleSize=25;                      /*Sample size for average value. Watch overflow when changed.*/

        int FuelLvlLow=729;                     /*This is where the thresolding values are set.*/
        int TnkTmpLow=818;                      /*Set to 85 degF = 818*/
        int HpTmpLow=257;                       /*Set to 170 degF = 257*/
        int HlpPrsLow=526;                      /*Set to 38 psi = 526*/
        int LlpPrsLow=500;                      /*This is a switch sent to analog 42=Closed 1020=0pen*/
        int TnkHtrLowLimit=800;                 /*Heater on at 87 degF = 800*/
        int TnkHtrHighLimit=733;                /*Heater off at 98 degF = 733*/


        /*This is setting up digital I/O. pins*/

        digitalDirection(0, INPUT);                     /*Mode input 0=diesel 1=auto*/
        digitalDirection(1, INPUT);                     /*Emergency shutdown 0=momentary 1=standard*/
        digitalDirection(2, INPUT);                     /*Ignition state 0=off 1=on*/

        digitalDirection(3, OUTPUT);                    /*Diesel solenoid 0=normally open*/
        digitalDirection(4, OUTPUT);                    /*Alt solenoid 0=normally closed*/
```

```
digitalDirection(5, OUTPUT);                                    /*Return solenoid 0= diesel return*/
digitalDirection(6, OUTPUT);                                    /*Heater 0=off*/
digitalDirection(7, OUTPUT);                                    /*High pressure pump 0=off*/
digitalDirection(8, OUTPUT);                                    /*Low pressure pump 0=off*/
digitalDirection(9, OUTPUT);                                    /*Auto shutdown relay 0=off*/

/*This initializes the outputs to the correct states.*/

digitalOutput(3, 0);                                            /*Diesel solenoid- Set to normally open(0)*/
digitalOutput(4, 0);                                            /*Alt solenoid- Set to normally closed(0)*/
digitalOutput(5, 0);                                            /*Return solenoid- Set to normally diesel return(0)*/
digitalOutput(6, 0);                                            /*Heater- Set to off(0)*/
digitalOutput(7, 0);                                            /*High pressure pump- Set to off(0)*/
digitalOutput(8, 0);                                            /*Low pressure pump- Set to off(0)*/
digitalOutput(9, 0);                                            /*Auto shutdown relay- Set to off(0)*/

action='A';                                                                     /*Set initial state to 'A'*/

SHOWINTRO(LetterDelay);

while (1)
{
        Mode=digitalInput(0);                                                   /*Reading user inputs*/
        EShtdn=digitalInput(1);
        IgnSt=digitalInput(2);


        FuelLvlAvg=GETFUELLEVEL(SampleSize);                            /*Initial Function Calls*/
        TnkTmpAvg=GETTANKTEMP(SampleSize);
        HpTmpAvg=GETHPLOOPTMP(SampleSize);
        HlpPrsAvg=GETHIGHLOOPPRES(SampleSize);
        LlpPrsAvg=GETLOWLOOPPRES(SampleSize);

        status=CHECKSTATUS(FuelLvlAvg, TnkTmpAvg, HpTmpAvg, HlpPrsAvg, LlpPrsAvg, FuelLvlLow,
                        TnkTmpLow, HpTmpLow, HlpPrsLow, LlpPrsLow);

        if (Mode==1 && IgnSt==1 && action!='C')                                 /*diesel mode operation*/
                action='A';

        else if (Mode==1 && IgnSt==0  && action!='C')                           /*diesel mode shutdown*/
                action='D';

        else
        {
                if (Mode==0 && IgnSt==1 && status=='w' && action!='C')          /*alternate mode desired*/
                        action='B';

                else if (Mode==0 && IgnSt==0 && action!='C')                    /*alt mode no wait shutdown*/
                        action='D';

                else
                {
                        if (Mode==0  && IgnSt==1 && status=='w' && action=='C')                 /*conditions fail*/
                                {
                                        PURGEFUELCONFIG(PurgeTime);
                                        action='B';
                                }

                        else if (Mode==1 && IgnSt==1 && action=='C')            /*mode switch to diesel*/
                                {
                                        PURGEFUELCONFIG(PurgeTime);
                                        action='A';
                                }

                        else if (Mode==0 && IgnSt==0 && action=='C')            /*alt mode purge shutdown*/
                                {
```

```
                                                PURGEFUELCONFIG(PurgeTime);
                                                action='D';
                                        }
                                else
                                        action='C';
                                /*alt mode operation*/
                        }
                }

                switch(action)
                {
                case 'A':
                        ASDOFF();
                        OFFPUMPS();
                        HEATERCONTROLOFF();

                        DIESELCONFIG();

                        FUELDISPLAY(FuelLvlAvg, FuelDsplyDur);
                        /*printString("case a");
                        printChar(status);
                        print_u08(EShtdn);*/
                        break;

                case 'B':
                        ASDON();
                        RUNPUMPS();

                        DIESELCONFIG();

                        HEATERCONTROLON(TnkTmpAvg, TnkHtrLowLimit, TnkHtrHighLimit);
                        FUELDISPLAY(FuelLvlAvg, FuelDsplyDur);
                        /*printString("case b");
                        printChar(status);*/

                        WAITCONDITIONS(FuelLvlAvg, TnkTmpAvg, HpTmpAvg, HlpPrsAvg,
                                                LlpPrsAvg, FuelLvlLow, TnkTmpLow, HpTmpLow,
                                                HlpPrsLow, LlpPrsLow, WaitDsplyTime);

                        break;

                case 'C':
                        ASDON();
                        RUNPUMPS();

                        ALTFUELCONFIG(ValveDelay);

                        HEATERCONTROLON(TnkTmpAvg, TnkHtrLowLimit, TnkHtrHighLimit);
                        FUELDISPLAY(FuelLvlAvg, FuelDsplyDur);
                        /*printString("case c");
                        printChar(status);
                        print_u08(EShtdn);*/
                        break;

                case 'D':
                        OFFPUMPS();
                        HEATERCONTROLOFF();
                        FUELDISPLAY(FuelLvlAvg, FuelDsplyDur);
                        ASDOFF();
                        /*printString("case d");*/

                }
        }
}
/*****************************************************************************************************/
void SHOWINTRO(int FLetterDelay)
```

```
{
	clearScreen();
	delayMs(100);

	printString("S");                                   /*SVO Control V1.3*/
	delayMs(FLetterDelay);
	printString("V");
	delayMs(FLetterDelay);
	printString("O");
	delayMs(FLetterDelay);
	printString(" ");
	delayMs(FLetterDelay);
	printString("C");
	delayMs(FLetterDelay);
	printString("o");
	delayMs(FLetterDelay);
	printString("n");
	delayMs(FLetterDelay);
	printString("t");
	delayMs(FLetterDelay);
	printString("r");
	delayMs(FLetterDelay);
	printString("o");
	delayMs(FLetterDelay);
	printString("l");
	delayMs(FLetterDelay);
	printString(" ");
	delayMs(FLetterDelay);
	printString("V");
	delayMs(FLetterDelay);
	printString("1");
	delayMs(FLetterDelay);
	printString(".");
	delayMs(FLetterDelay);
	printString("4");

	lowerLine();
	delayMs(500);
	printString(" Joseph W. Souza");
	delayMs(2000);

	clearScreen();
	printString("SVO Control V1.4");
	lowerLine();
	printString(" Copyright 2012");
	delayMs(2000);
}
/*************************************************************************************************************/
int GETFUELLEVEL(int FSampleSize)
{
	int FFuelLvl=0;                                                         /*initializing*/
	int FFuelLvlAvg;

	int i;                                                                  /*"for" loop variable*/

	for (i=0;i<FSampleSize;i++)
		{
			FFuelLvl=FFuelLvl+(analog10(0));

			delayMs(5);
		}

	FFuelLvlAvg=FFuelLvl/FSampleSize;

	return FFuelLvlAvg;
}
```

```
/*****************************************************************************************************/
int GETTANKTEMP(int FSampleSize)
{
	int FTnkTmp=0;                                                          /*initializing*/
	int FTnkTmpAvg;

	int i;                                                                  /*"for" loop variable*/

	for (i=0;i<FSampleSize;i++)
		{
			FTnkTmp=FTnkTmp+(analog10(1));

			delayMs(5);
		}

	FTnkTmpAvg=FTnkTmp/FSampleSize;

	return FTnkTmpAvg;
}

/*****************************************************************************************************/
int GETHPLOOPTMP(int FSampleSize)
{
	int FHpTmp=0;                                                           /*initializing*/
	int FHpTmpAvg;

	int i;                                                                  /*"for" loop variable*/

	for (i=0;i<FSampleSize;i++)
		{
			FHpTmp=FHpTmp+(analog10(2));

			delayMs(5);
		}

	FHpTmpAvg=FHpTmp/FSampleSize;

	return FHpTmpAvg;
}
/*****************************************************************************************************/
int GETHIGHLOOPPRES(int FSampleSize)
{
	int FHlpPrs=0;                                                          /*initializing*/
	int FHlpPrsAvg;

	int i;                                                                  /*"for" loop variable*/

	for (i=0;i<FSampleSize;i++)
		{
			FHlpPrs=FHlpPrs+(analog10(3));

			delayMs(5);
		}

	FHlpPrsAvg=(FHlpPrs/FSampleSize);

	return FHlpPrsAvg;
}
/*****************************************************************************************************/
int GETLOWLOOPPRES(int FSampleSize)
{
	int FLlpPrs=0;                                                          /*initializing*/
	int FLlpPrsAvg;
```

```
	int i;                                                                                                  /*"for"
loop variable*/

	for (i=0;i<FSampleSize;i++)
		{
			FLlpPrs=FLlpPrs+(analog10(4));

			delayMs(5);
		}

	FLlpPrsAvg=FLlpPrs/FSampleSize;

	return FLlpPrsAvg;
}
/*****************************************************************************************************************/
char CHECKSTATUS(int FFuelLvlAvgST, int FTnkTmpAvgST, int FHpTmpAvgST,
                                        int FHlpPrsAvgST, int FLlpPrsAvgST, int FFuelLvlLowST,
                                        int FTnkTmpLowST, int FHpTmpLowST, int FHlpPrsLowST,
                                        int FLlpPrsLowST)
{
	char Fstatus;

	if (FFuelLvlAvgST < FFuelLvlLowST &&                    /*Avg value is greater than low value when too low*/
		FTnkTmpAvgST < FTnkTmpLowST &&                                  /*Avg value is greater than low value when
too cold*/
		FHpTmpAvgST < FHpTmpLowST  &&                                   /*Avg value is greater than low value when
too cold*/
		FHlpPrsAvgST < FHlpPrsLowST &&                                  /*Avg value is greater than low value when
too low*/
		FLlpPrsAvgST > FLlpPrsLowST)                            /*Avg value is less than low value when too low*/

			Fstatus='r';

	else
			Fstatus='w';

	return Fstatus;
}
/*****************************************************************************************************************/
void WAITCONDITIONS(int WFuelLvlAvg, int WTnkTmpAvg, int WHpTmpAvg,
                                                int WHlpPrsAvg, int WLlpPrsAvg, int WFuelLvlLow,
                                                int WTnkTmpLow, int WHpTmpLow, int WHlpPrsLow,
                                                int WLlpPrsLow, int WWaitDsplyTime)
{
	int FuelLevelPrcnt;
	int AdjFuelLevelPrcnt;
	int TnkTmpDegF;
	int InjectTempDegF;
	int InjectPresPsi;

	if (WFuelLvlAvg > WFuelLvlLow)
		{
			clearScreen();
			printString("   Fuel level");
	/*Fuel Level Sensor*/

			FuelLevelPrcnt=(s08)((790-WFuelLvlAvg)/5.60);

			lowerLine();
			printString("     ");

			if (FuelLevelPrcnt > 100)
				AdjFuelLevelPrcnt = 100;

			else if(FuelLevelPrcnt < 0)
				AdjFuelLevelPrcnt = 0;
```

```
                        else
                                AdjFuelLevelPrcnt = (u08)FuelLevelPrcnt;

                        printPlain_u08(AdjFuelLevelPrcnt);

                        printString("%");

                        delayMs(WWaitDsplyTime);
                }

if (WTnkTmpAvg > WTnkTmpLow)
                {
                        clearScreen();
                        printString("Tank Temperature");                                    /*Tank Temperature Sensor*/

                        TnkTmpDegF=(s08)((-0.15133 * WTnkTmpAvg) + 208.9301);

                        lowerLine();
                        printString("    ");

                        printPlain_u08(TnkTmpDegF);

                        printString(" deg F");

                        delayMs(WWaitDsplyTime);
                }

if (WHpTmpAvg > WHpTmpLow)
                {
                        clearScreen();
                        printString(" Injection Temp");                                    /*Injection Loop Temperature*/

                        InjectTempDegF=(s08)((-0.15133 * WHpTmpAvg) + 208.9301);

                        lowerLine();
                        printString("    ");

                        printPlain_u08(InjectTempDegF);

                        printString(" deg F");

                        delayMs(WWaitDsplyTime);
                }

if (WHlpPrsAvg > WHlpPrsLow)
                {
                        clearScreen();
                        printString(" Injection Pres");                                    /*Injection Loop Pressure*/

                        InjectPresPsi=(s08)((-0.1211 * WHlpPrsAvg) + 101.7269);

                        lowerLine();
                        printString("     ");

                        printPlain_u08(InjectPresPsi);

                        printString(" psi");

                        delayMs(WWaitDsplyTime);
                }

if (WLlpPrsAvg < WLlpPrsLow)
                {
                        clearScreen();
                        printString(" Feed Pressure");                                     /*Feed Loop Pressure*/
```

```
                        lowerLine();

                        if (WLlpPrsAvg > 500)
                                printString("    GOOD");

                        else
                                printString("    LOW");

                        delayMs(WWaitDsplyTime);
                }
}



/*****************************************************************************************************************/
void RUNPUMPS(void)
{
        digitalOutput(7,1);
        digitalOutput(8,1);
}
/*****************************************************************************************************************/
void OFFPUMPS(void)
{
        digitalOutput(7,0);
        digitalOutput(8,0);
}

/*****************************************************************************************************************/
void ASDOFF(void)
{
        digitalOutput(9,0);
}
/*****************************************************************************************************************/
void ASDON(void)
{
        digitalOutput(9,1);
}
/*****************************************************************************************************************/
void HEATERCONTROLON(int FTnkTmpAvgHC, int FTnkHtrLowLimit, int FTnkHtrHighLimit)
{
        if (FTnkTmpAvgHC>FTnkHtrLowLimit)                       /*Setting heater to on*/
                digitalOutput(6,1);

        else if (FTnkTmpAvgHC<FTnkHtrHighLimit)         /*Setting heater to off*/
                digitalOutput(6,0);
}
/*****************************************************************************************************************/
void HEATERCONTROLOFF(void)
{
        digitalOutput(6,0);                                                             /*Setting heater to off*/
}
/*****************************************************************************************************************/
void DIESELCONFIG(void)
{
        digitalOutput(3, 0);                                                            /*Diesel solenoid 0=normally open*/
        digitalOutput(4, 0);                                                            /*Alt solenoid 0=normally closed*/
        digitalOutput(5, 0);                                                            /*Return solenoid 0= diesel return*/
}
/*****************************************************************************************************************/
void ALTFUELCONFIG(int FValveDelay)
{
        digitalOutput(5, 1);                                                            /*Return solenoid 0= diesel return*/
        delayMs(FValveDelay);
        digitalOutput(3, 1);                                                            /*Diesel solenoid 0=normally open*/
```

```
	digitalOutput(4, 1);                                                    /*Alt solenoid 0=normally closed*/
}
/*****************************************************************************************************************/
void PURGEFUELCONFIG(int FPurgeTime)
{
	int count;
	int EShtdn;

	clearScreen();
	printString("   PURGE TIME");

	digitalOutput(3, 0);                                                    /*Diesel solenoid 0=normally open*/
	digitalOutput(4, 0);                                                    /*Alt solenoid 0=normally closed*/
	digitalOutput(5, 1);                                                    /*Return solenoid 0= diesel return*/

	for(count=FPurgeTime;count>=0;)
		{
			EShtdn=digitalInput(1);
			lowerLine();
			printString("    ");
			print_u08(count);
			printString(" SECS");
			delayMs(1000);
			count--;

			if (EShtdn==0 || count==0)
				{
					count=0;
					digitalOutput(5,0);
				}
		}
}
/*****************************************************************************************************************/
void FUELDISPLAY(int FFuelLvlAvg, int FFuelDsplyDur)
{
	int i;                                                                  /*"for" loop variables*/
	int u;

	clearScreen();

	printString("   FUEL LEVEL");                                   /*upper line of fuel display*/


	lowerLine();

		if (FFuelLvlAvg <= 265)                                                                 /*100% full level*/
			{
				printString("F");

				for (i=0; i<14; i++)
				{
					printChar(BLACK_SQUARE);
				}

				printString("E");
			}


		else if (FFuelLvlAvg <= 290 && FFuelLvlAvg > 265)
			{
				printString("F");
				printChar(" ");

				for (i=0; i<13; i++)
				{
					printChar(BLACK_SQUARE);
```

```
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 330 && FFuelLvlAvg > 290)
        {
                printString("F");

                for (i=0; i<2; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<12; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 370 && FFuelLvlAvg > 330)
        {
                printString("F");

                for (i=0; i<3; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<11; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 410 && FFuelLvlAvg > 370)
        {
                printString("F");

                for (i=0; i<4; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<10; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 450 && FFuelLvlAvg > 410)
        {
                printString("F");

                for (i=0; i<5; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<9; u++)
```

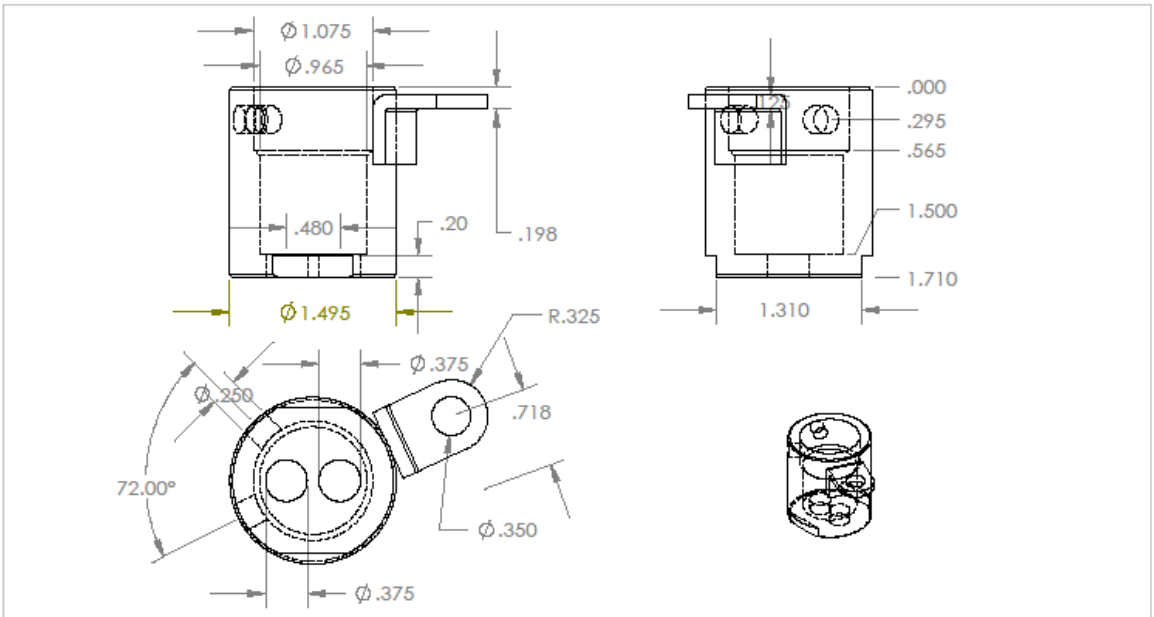
```
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 490 && FFuelLvlAvg > 450)
        {
                printString("F");

                for (i=0; i<6; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<8; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 530 && FFuelLvlAvg > 490)
        {
                printString("F");

                for (i=0; i<7; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<7; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 570 && FFuelLvlAvg > 530)
        {
                printString("F");

                for (i=0; i<8; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<6; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 610 && FFuelLvlAvg > 570)
        {
                printString("F");

                for (i=0; i<9; i++)
                {
                        printChar(" ");
                }
```

```
                for (u=0; u<5; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 650 && FFuelLvlAvg > 610)
        {
                printString("F");

                for (i=0; i<10; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<4; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 690 && FFuelLvlAvg > 650)
        {
                printString("F");

                for (i=0; i<11; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<3; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 730 && FFuelLvlAvg > 690)
        {
                printString("F");

                for (i=0; i<12; i++)
                {
                        printChar(" ");
                }

                for (u=0; u<2; u++)
                {
                        printChar(BLACK_SQUARE);
                }

                printString("E");
        }

else if (FFuelLvlAvg <= 770 && FFuelLvlAvg > 730)
        {
                printString("F");

                for (i=0; i<13; i++)
                {
```

```
                        printChar(" ");
                }

                printChar(BLACK_SQUARE);
                printString("E");
        }

    else                                                                    /* 0% full level*/
        {
                printString("F");

                for (i=0; i<14; i++)
                {
                        printChar(" ");
                }

                printString("E");
        }

    delayMs(FFuelDsplyDur);                                 /*Duration in milliseconds display is on screen*/

}
```

Appendix G

Mechanical Drawings



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN	J.Souza	5/16/06
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL Steel	COMMENTS:		
		FINISH			
NEXT ASSY	USED ON				
APPLICATION		DO NOT SCALE DRAWING			

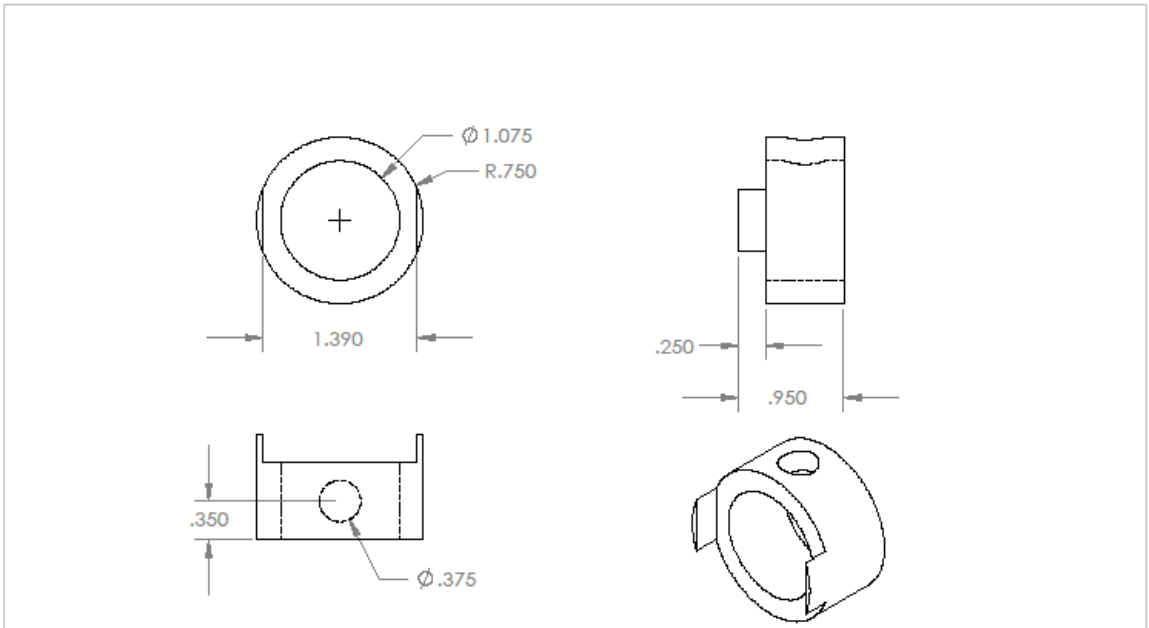
SVO Fuel System

TITLE:

Fuel Combiner

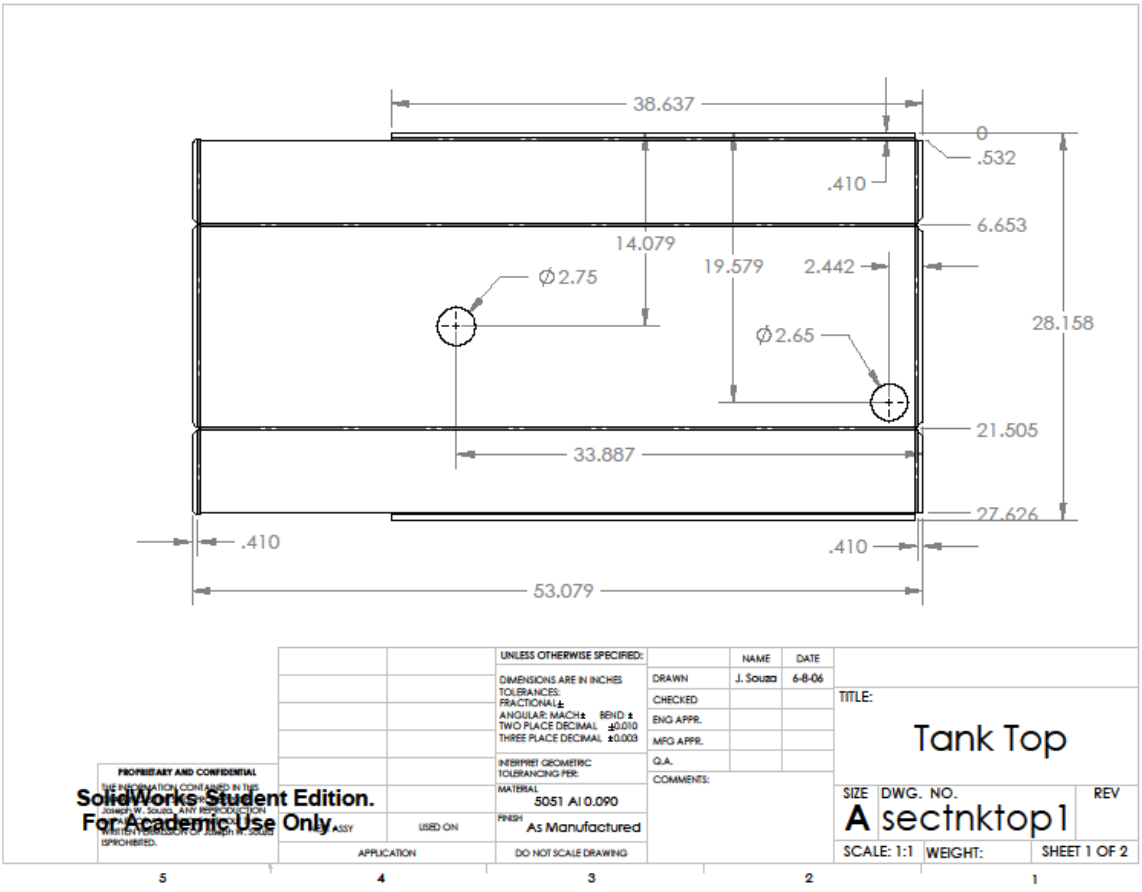
SIZE	DWG. NO.	REV
A	Inlet 1	
SCALE: 2:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

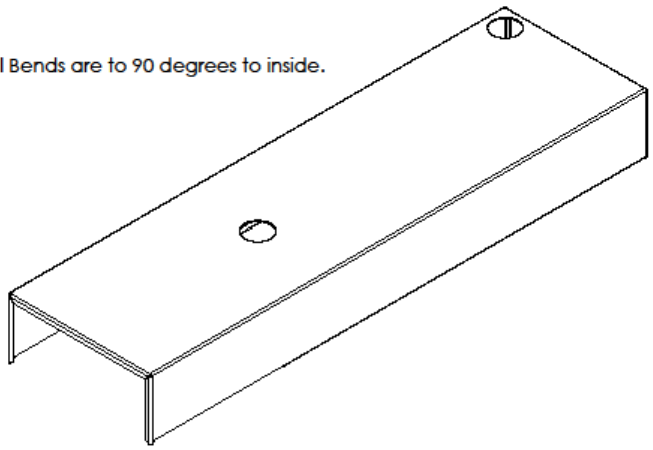


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE	SVO Fuel System
		DIMENSIONS ARE IN INCHES	DRAWN	J. Souza	5/17/06	TITLE:
		TOLERANCES: FRACTIONAL ±	CHECKED			Diesel Fuel Boss
		ANGULAR: MACH ± BEND ±	ENG APPR.			
		TWO PLACE DECIMAL ±	MFG APPR.			
		THREE PLACE DECIMAL ±	Q.A.			
		INTERPRET GEOMETRIC TOLERANCING PER:	COMMENTS:			SIZE A DWG. NO. Fuel Ftng 2 REV
		MATERIAL Steel				
	USED ON	FINISH				SCALE: 1:1 WEIGHT: SHEET 1 OF 1
	NEXT ASSY	DO NOT SCALE DRAWING				
	APPLICATION					

5 4 3 2 1



All Bends are to 90 degrees to inside.



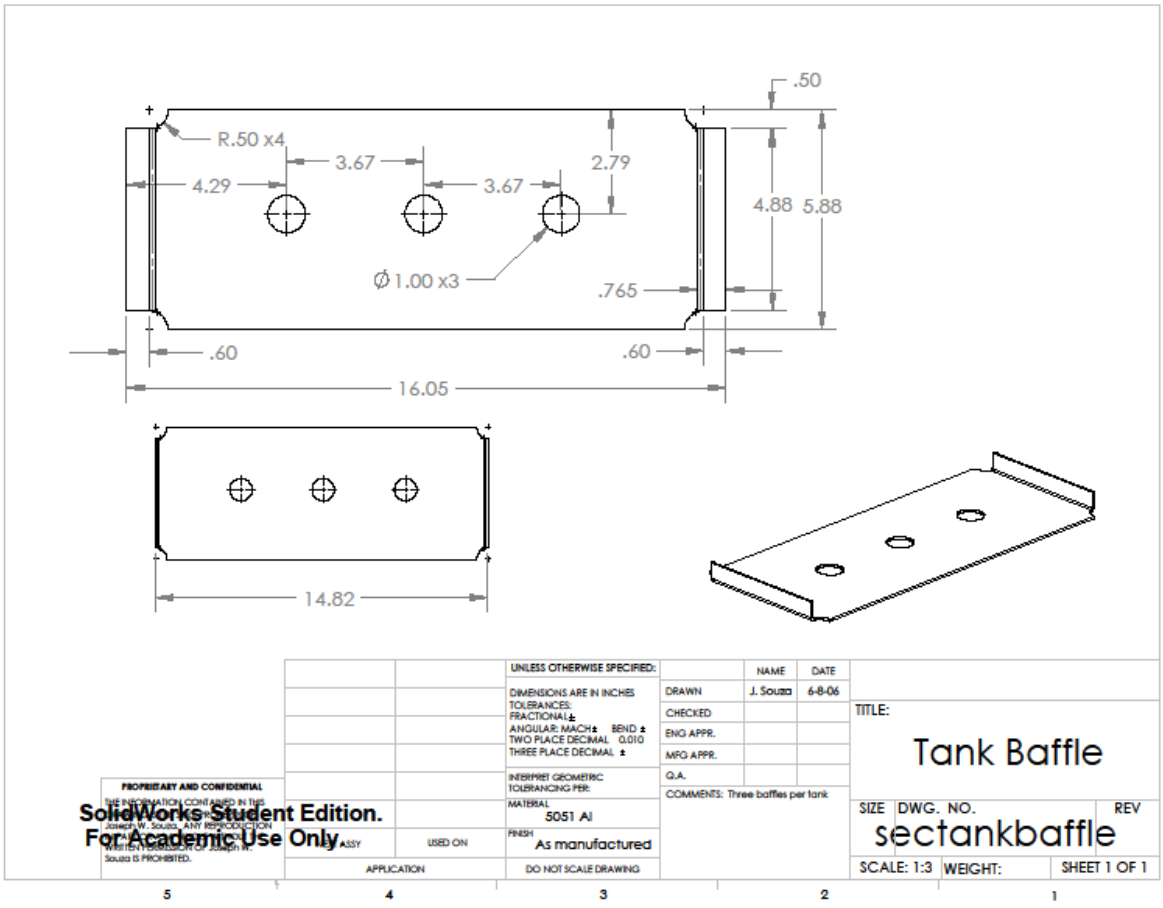
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±0.010 THREE PLACE DECIMAL ±0.003	DRAWN	J. Souza	6-8-06
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL 5051 Al 0.090	COMMENTS:		
		FINISH As Manufactured			
NEXT ASSY	USED ON	DO NOT SCALE DRAWING			
APPLICATION					

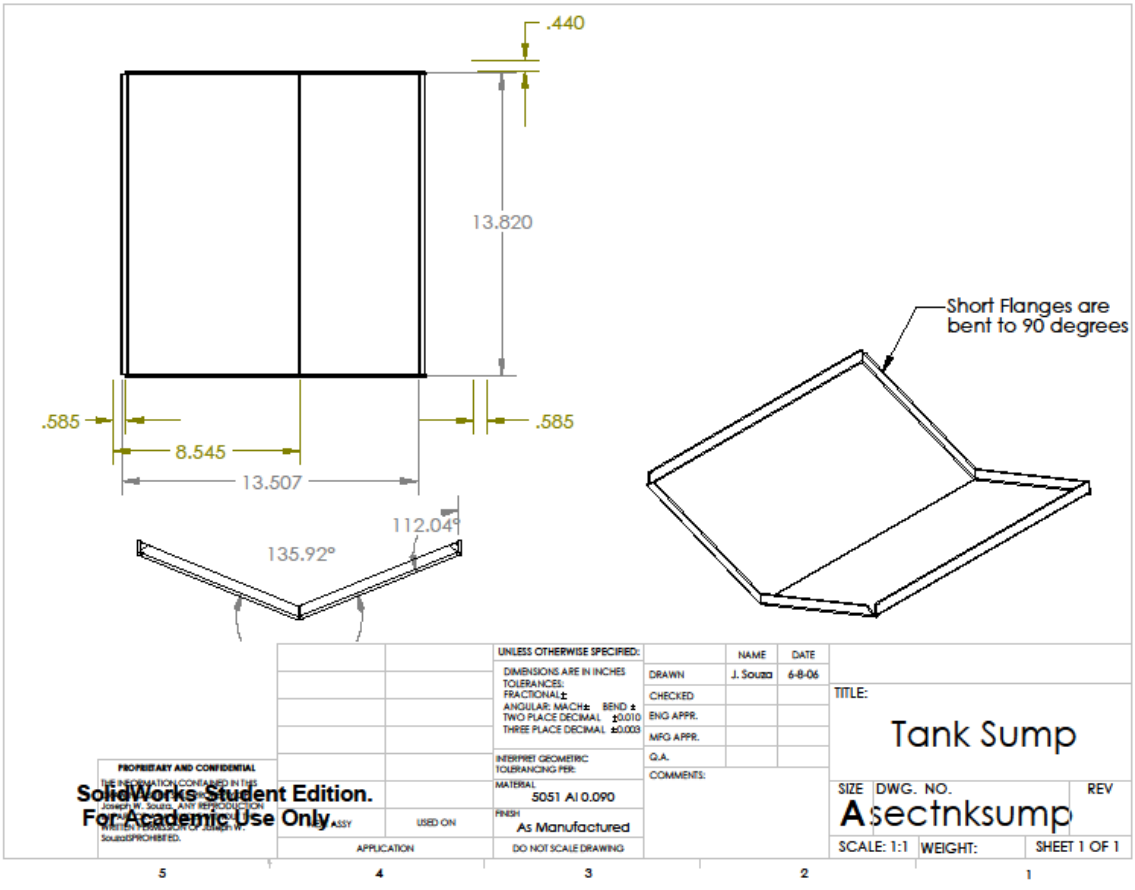
TITLE:

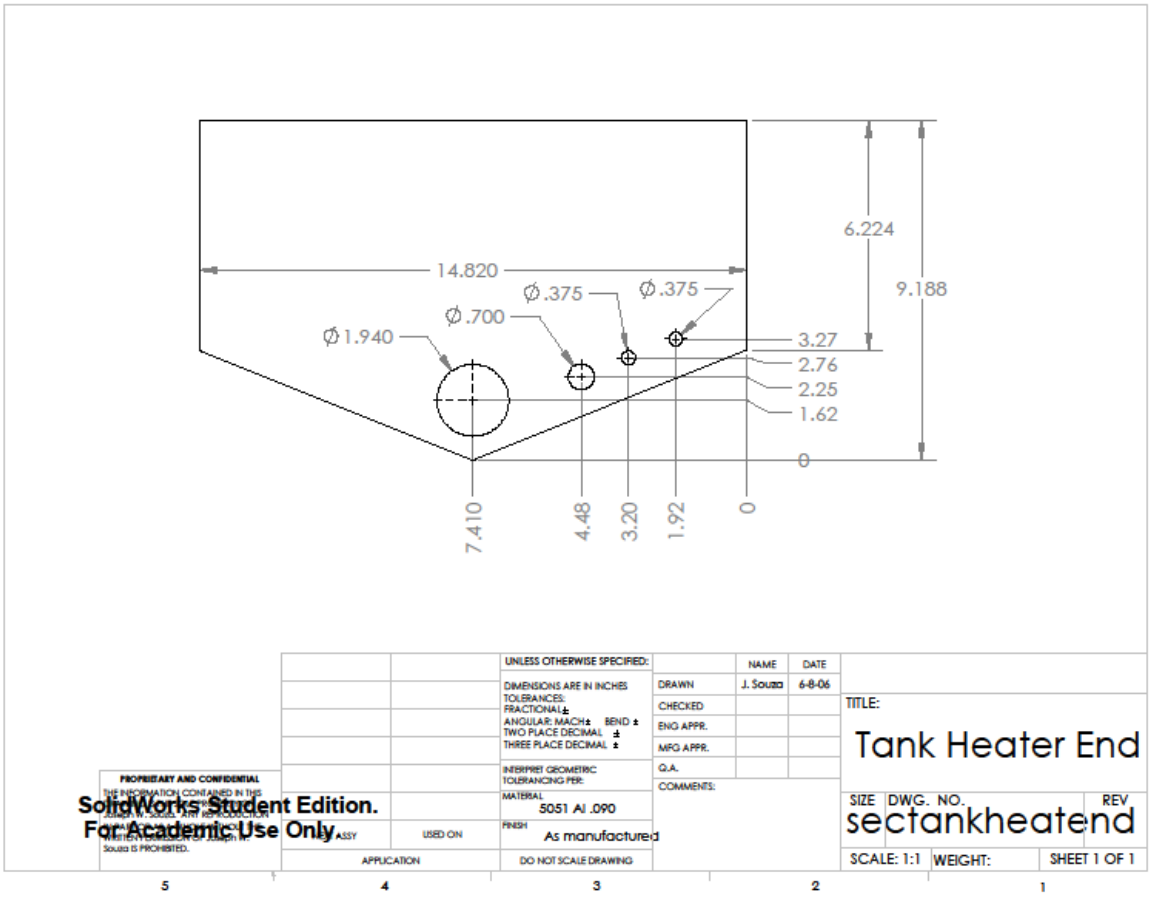
Tank Top

SIZE	DWG. NO.	REV
A	sectnktop	
SCALE: 1:1	WEIGHT:	SHEET 2 OF 2

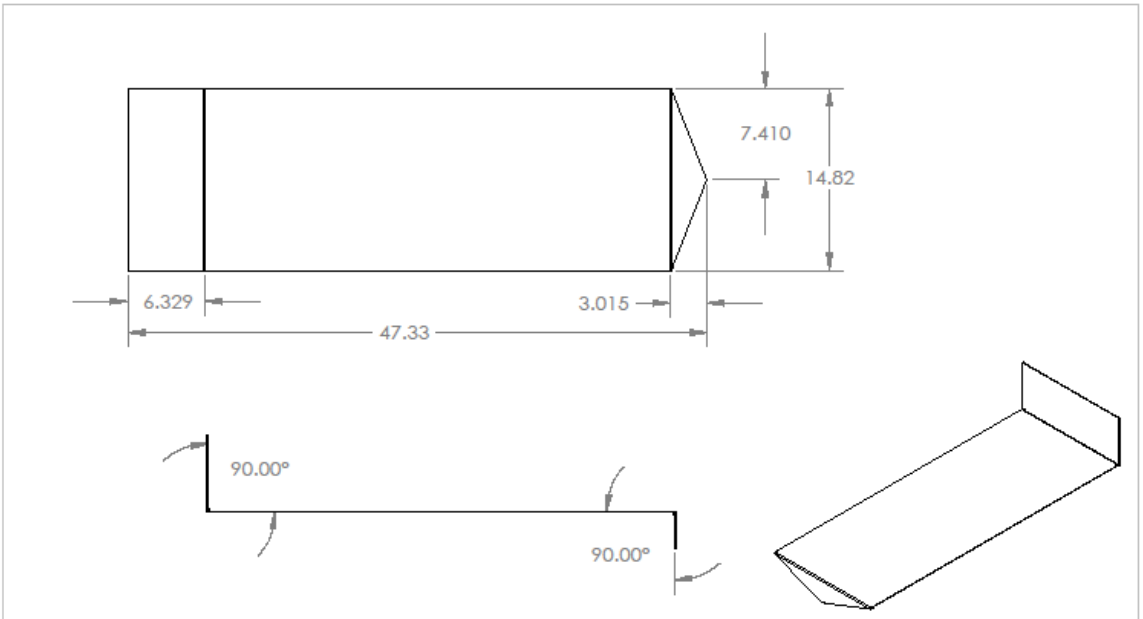
5 4 3 2 1




.440
13.820
.585
8.545
13.507
.585
112.04°
135.92°
Short Flanges are
bent to 90 degrees
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±0.010
THREE PLACE DECIMAL ±0.003
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
5051 Al 0.090
FINISH
As Manufactured
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
J. Souza
6-8-06
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
Tank Sump
SIZE
DWG. NO.
REV
Asectnksump
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS
Joseph W. Souza. ANY REPRODUCTION
WITHOUT PERMISSION OF Joseph W.
Souza IS PROHIBITED.
SolidWorks Student Edition.
For Academic Use Only
ASSY
USED ON
APPLICATION
5
4
3
2
1



UNLESS OTHERWISE SPECIFIED:		NAME	DATE		
DIMENSIONS ARE IN INCHES	DRAWN	J. Souza	6-8-06		
TOLERANCES: FRACTIONAL ±	CHECKED			TITLE:	
ANGULAR: MACH ± BEND ±	ENG APPR.			Tank Heater End	
TWO PLACE DECIMAL ±	MFG APPR.				
THREE PLACE DECIMAL ±	Q.A.				
INTERPRET GEOMETRIC TOLERANCING PER:	COMMENTS:				
MATERIAL 5051 Al .090				SIZE DWG. NO.	REV
FINISH As manufactured				sectankheatend	
DO NOT SCALE DRAWING				SCALE: 1:1 WEIGHT:	SHEET 1 OF 1



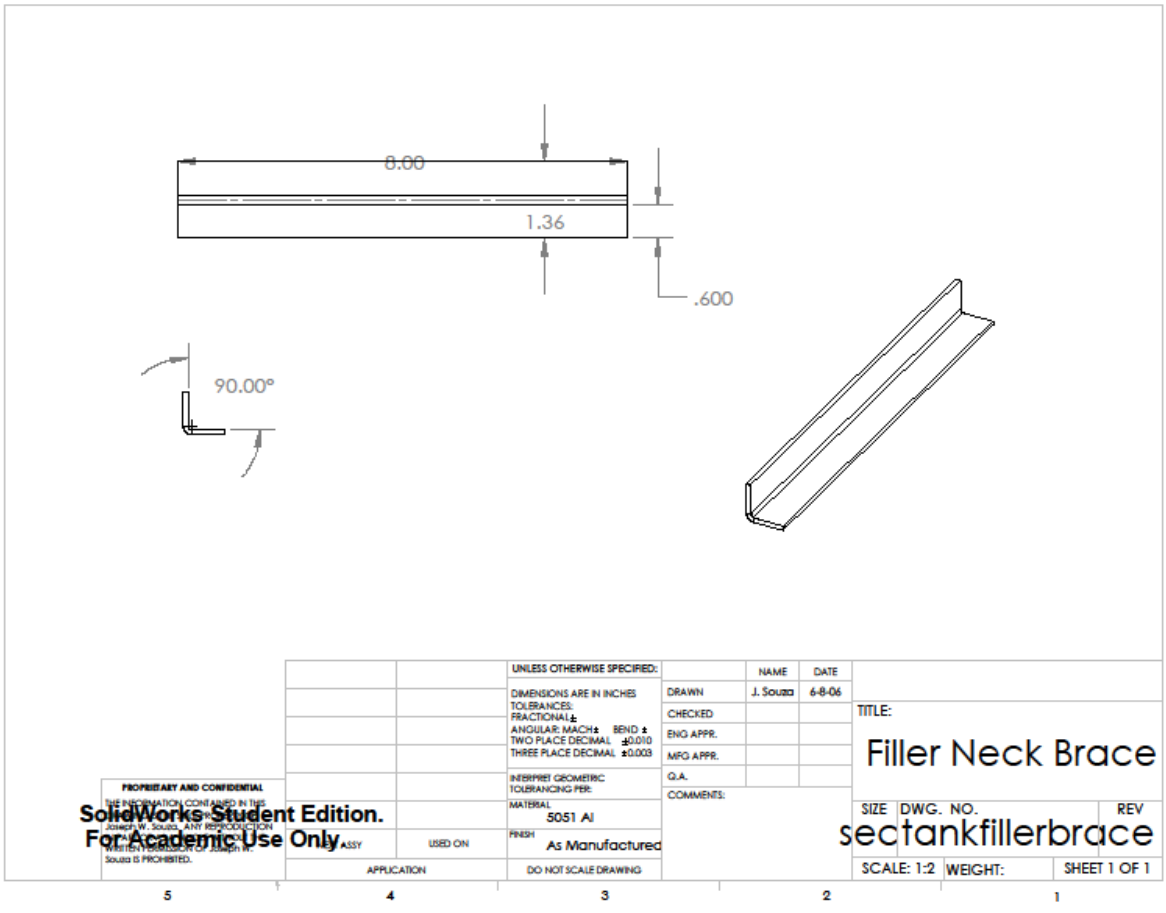
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES	DRAWN	J Souza	6-8-06
		TOLERANCES: FRACTIONAL ±	CHECKED		
		ANGULAR: MACH ± BEND ±	ENG APPR.		
		TWO PLACE DECIMAL ±0.010	MFG APPR.		
		THREE PLACE DECIMAL ±0.003	Q.A.		
		INTERPRET GEOMETRIC TOLERANCING PER:	COMMENTS:		
		MATERIAL 5051 Al			
	USED ON	FINISH As Manufactured			
	APPLICATION	DO NOT SCALE DRAWING			

TITLE: Tank Bottom

SIZE | DWG. NO. | REV: sectankbottom

SCALE: 1:9 | WEIGHT: | SHEET 1 OF 1

5 4 3 2 1



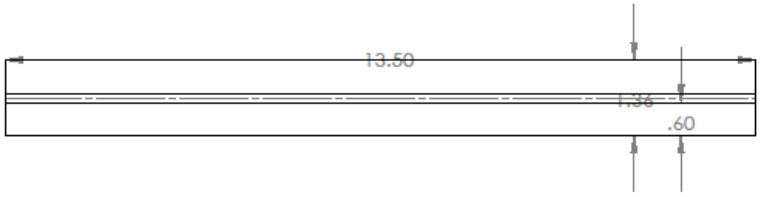
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES	DRAWN	J. Souza	6-8-06
		TOLERANCES: FRACTIONAL ±	CHECKED		
		ANGULAR: MACH ± BEND ±	ENG APPR.		
		TWO PLACE DECIMAL ±0.010	MFG APPR.		
		THREE PLACE DECIMAL ±0.003	Q.A.		
		INTERPRET GEOMETRIC TOLERANCING PER:	COMMENTS:		
		MATERIAL 5051 Al			
		FINISH As Manufactured			
NEXT ASSY	USED ON				
APPLICATION		DO NOT SCALE DRAWING			

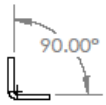
TITLE: Filler Neck Brace

SIZE | DWG. NO. | REV: sectankfillerbrace

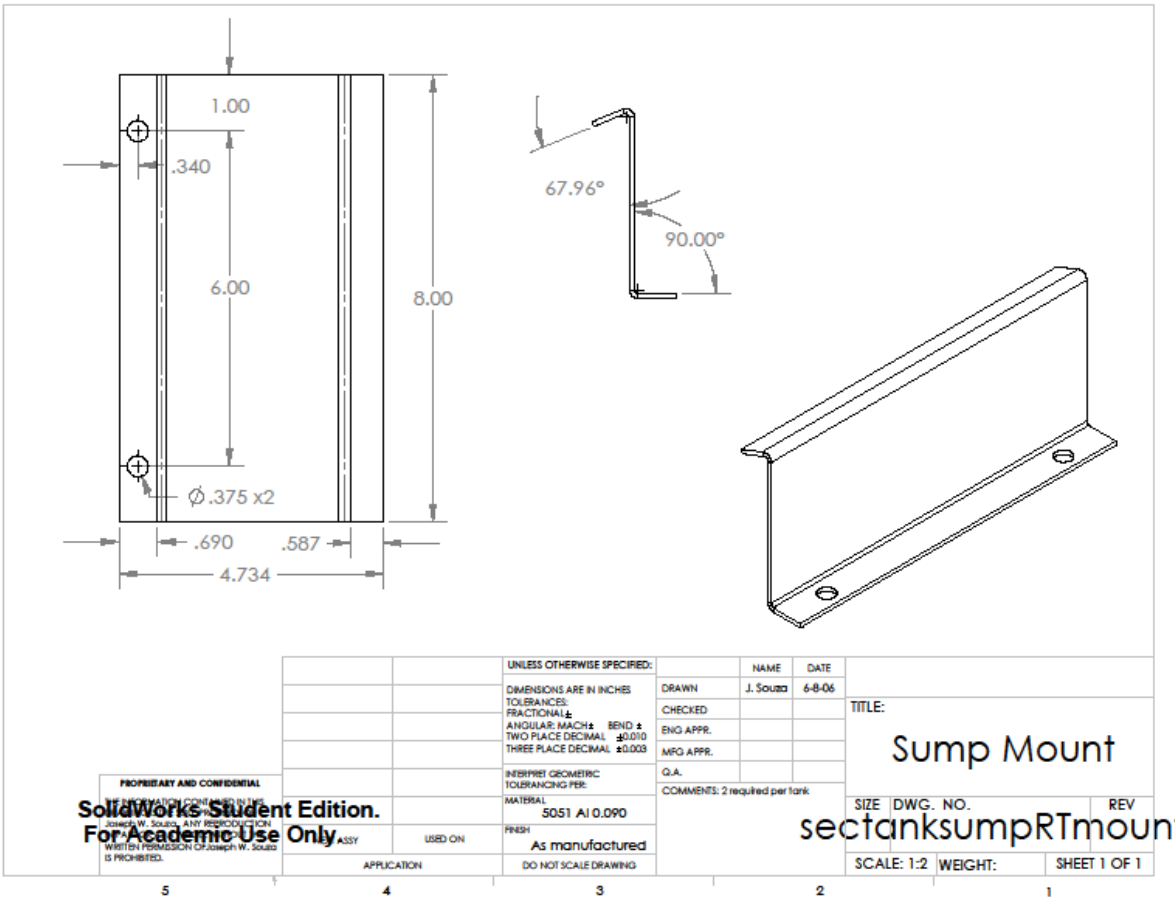
SCALE: 1:2 | WEIGHT: | SHEET 1 OF 1

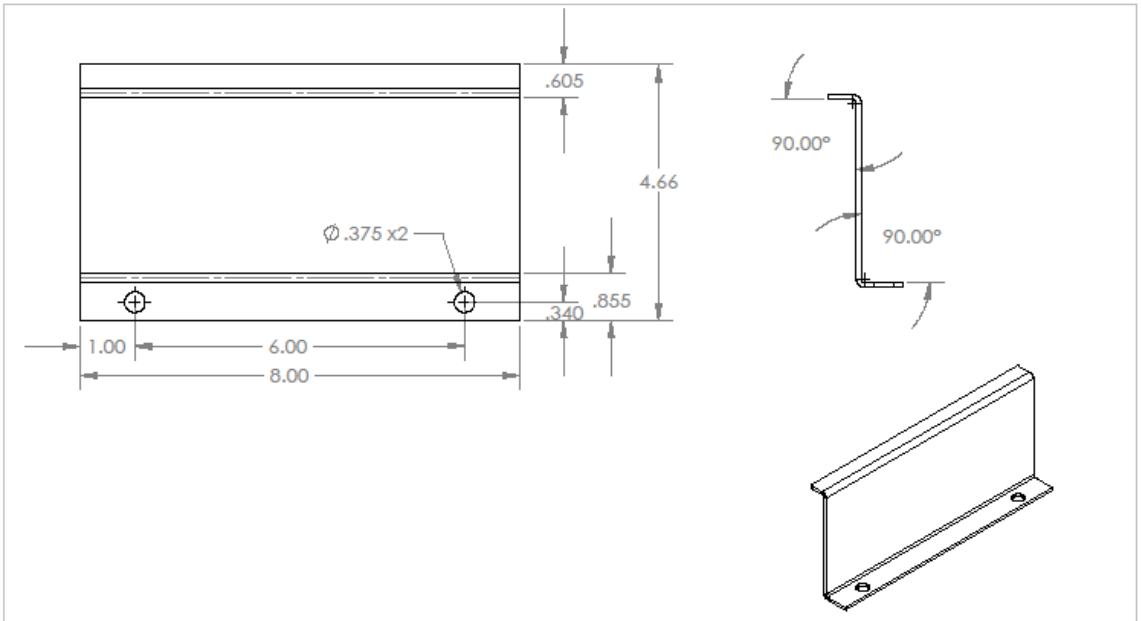
5 4 3 2 1





UNLESS OTHERWISE SPECIFIED:		NAME	DATE		
DIMENSIONS ARE IN INCHES	DRAWN	J. Souza	6-8-06		
TOLERANCES: FRACTIONAL ±	CHECKED			TITLE: Tank Heater Brace	
ANGULAR: MACH ± BEND ±	ENG APPR.				
TWO PLACE DECIMAL ±0.010	MFG APPR.				
THREE PLACE DECIMAL ±	Q.A.				
INTERPRET GEOMETRIC TOLERANCING PER:	COMMENTS:				
MATERIAL 5051 Al 0.090				SIZE DWG. NO. sectankheaterbrace	REV
FINISH As Manufactured				SCALE: 1:1 WEIGHT:	SHEET 1 OF 1
DO NOT SCALE DRAWING					

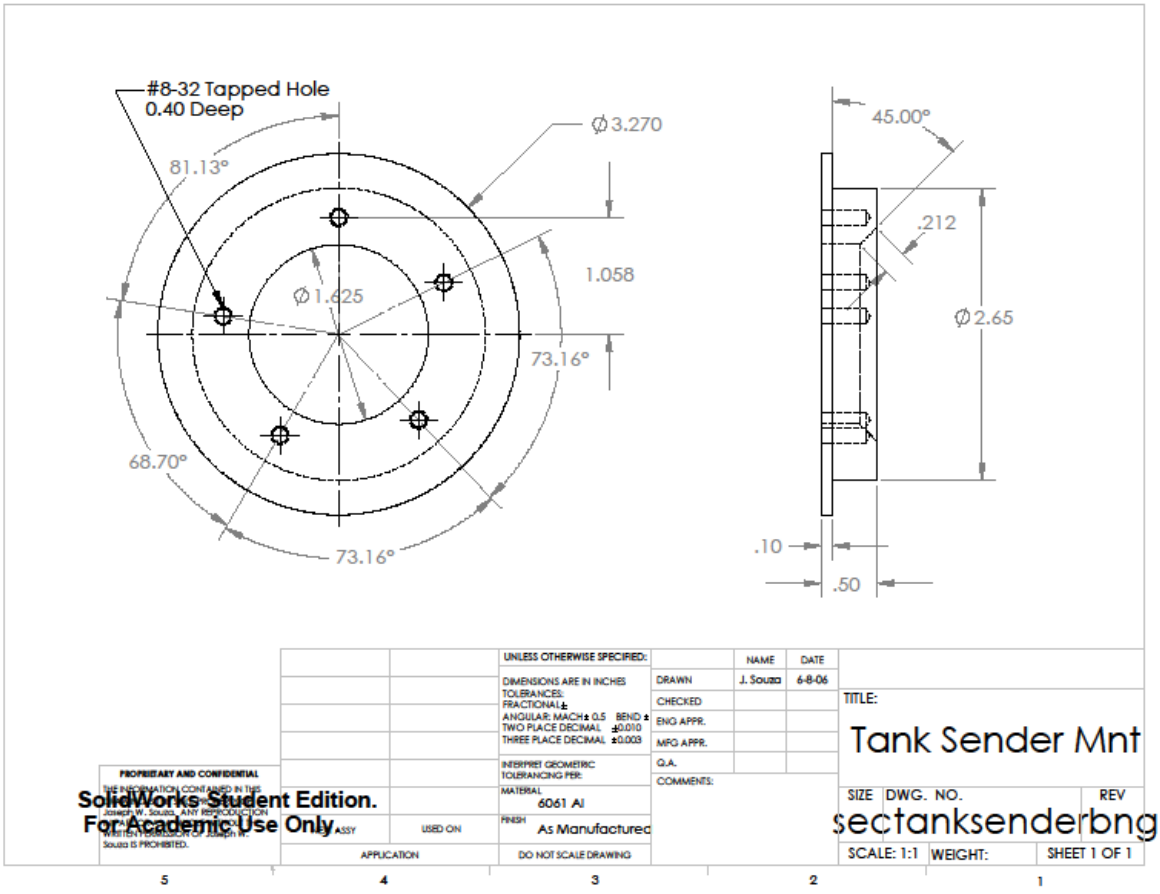


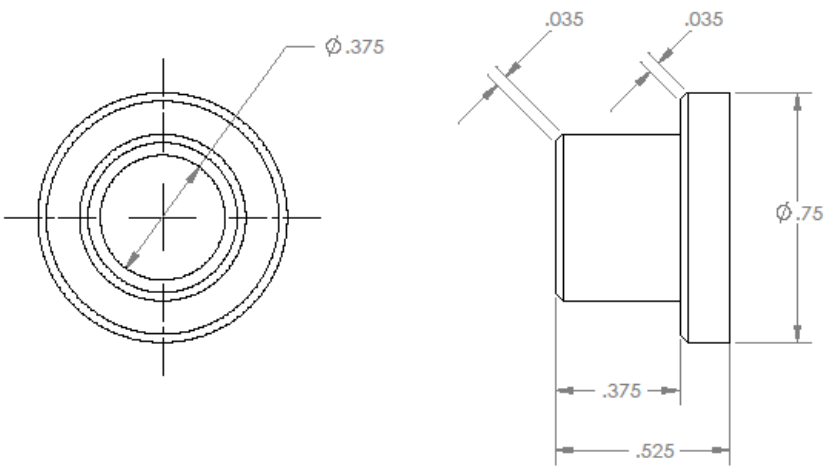


UNLESS OTHERWISE SPECIFIED:		NAME	DATE		
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±0.010 THREE PLACE DECIMAL ±0.003	DRAWN	J. Souza	6-8-06	TITLE: Back tank Mount	
	CHECKED				
	ENG APPR.				
	MFG APPR.				
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.				
MATERIAL 5051 Al	COMMENTS:			SIZE DWG. NO. sectankBKmount	REV
FINISH As Manufactured					
DO NOT SCALE DRAWING				SCALE: 1:4 WEIGHT:	SHEET 1 OF 1

NEXT ASSY | USED ON | APPLICATION

5 4 3 2 1





		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES	DRAWN	J. Souza	6-8-06
		TOLERANCES: FRACTIONAL ±	CHECKED		
		ANGULAR: MACH ± BEND ±	ENG APPR.		
		TWO PLACE DECIMAL ±0.010	MFG APPR.		
		THREE PLACE DECIMAL ±0.003	Q.A.		
		INTERPRET GEOMETRIC TOLERANCING PER:	COMMENTS:		
		MATERIAL 6061 Al			
NEXT ASSY	USED ON	FINISH As Manufactured			
APPLICATION		DO NOT SCALE DRAWING			

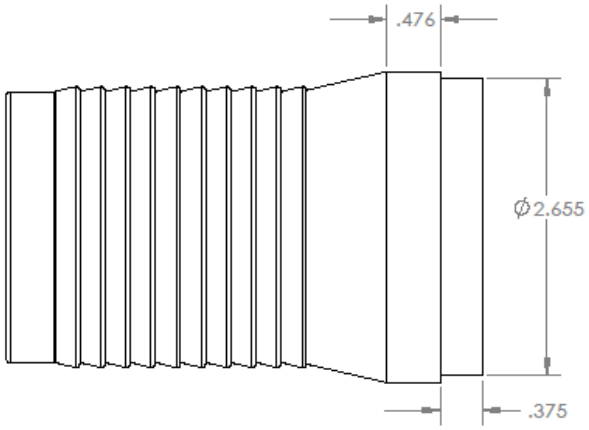
TITLE: Tank Pickup Boss

SIZE | DWG. NO. | REV

sectankfuelpickupboss

SCALE: 4:1 | WEIGHT: | SHEET 1 OF 1

5 4 3 2 1



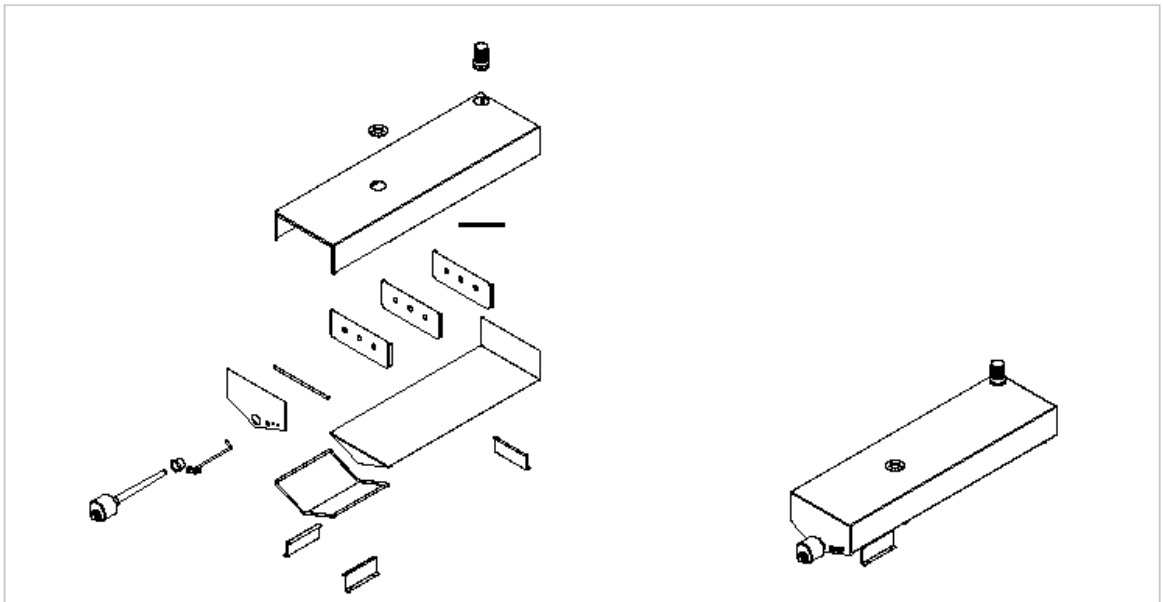
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES	DRAWN	J. Souza	6-8-06
		TOLERANCES: FRACTIONAL ±	CHECKED		
		ANGULAR: MACH ± BEND ±	ENG APPR.		
		TWO PLACE DECIMAL ±0.010	MFG APPR.		
		THREE PLACE DECIMAL ±0.003	Q.A.		
		INTERPRET GEOMETRIC TOLERANCING PER:	COMMENTS: This is a modified production part		
		MATERIAL 6061 Al			
NEXT ASSY	USED ON	FINISH As Manufactured			
APPLICATION		DO NOT SCALE DRAWING			

TITLE: Filler Neck

SIZE | DWG. NO. | REV: sectankfillerbung

SCALE: 1:1 | WEIGHT: | SHEET 1 OF 1

5 4 3 2 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES	DRAWN	J.Souza	5-22-06
		TOLERANCES: FRACTIONAL ±	CHECKED		
		ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

SVO Fuel System

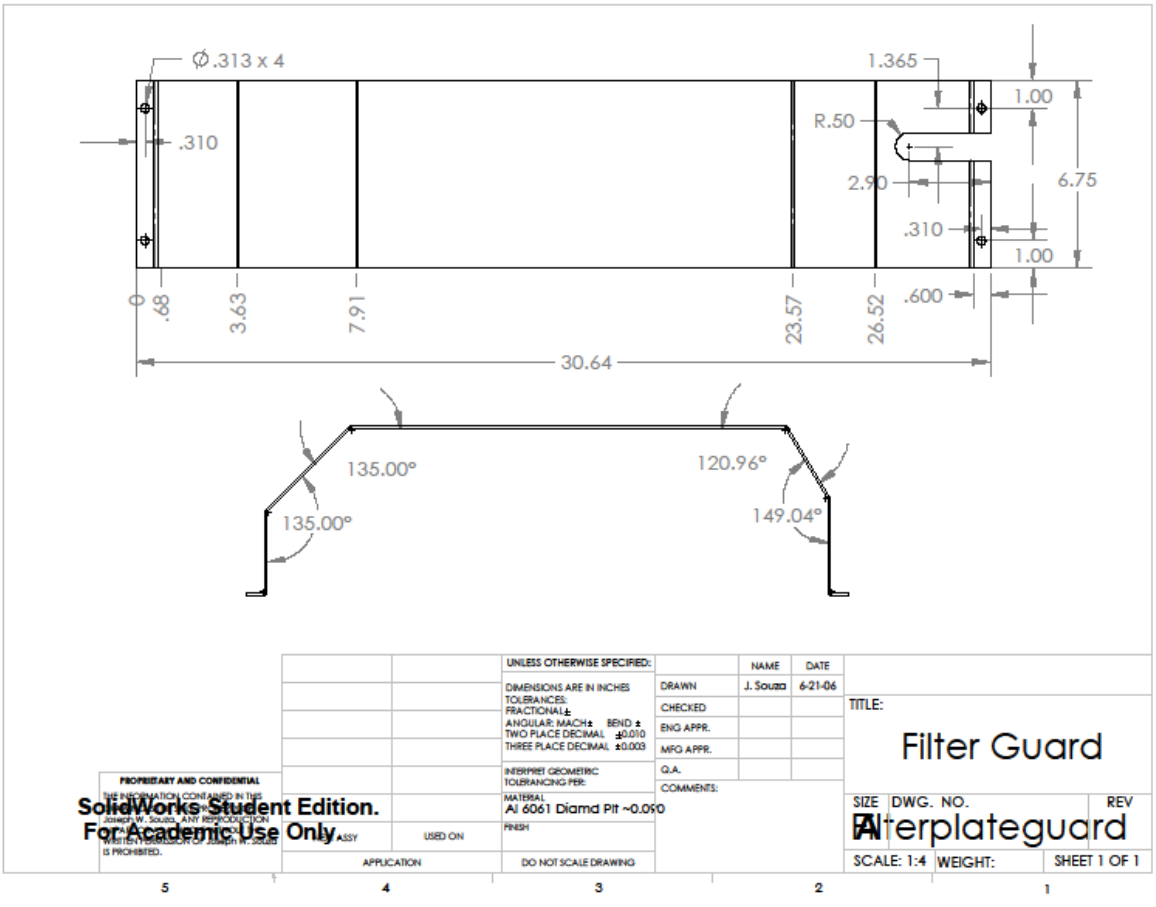
TITLE:

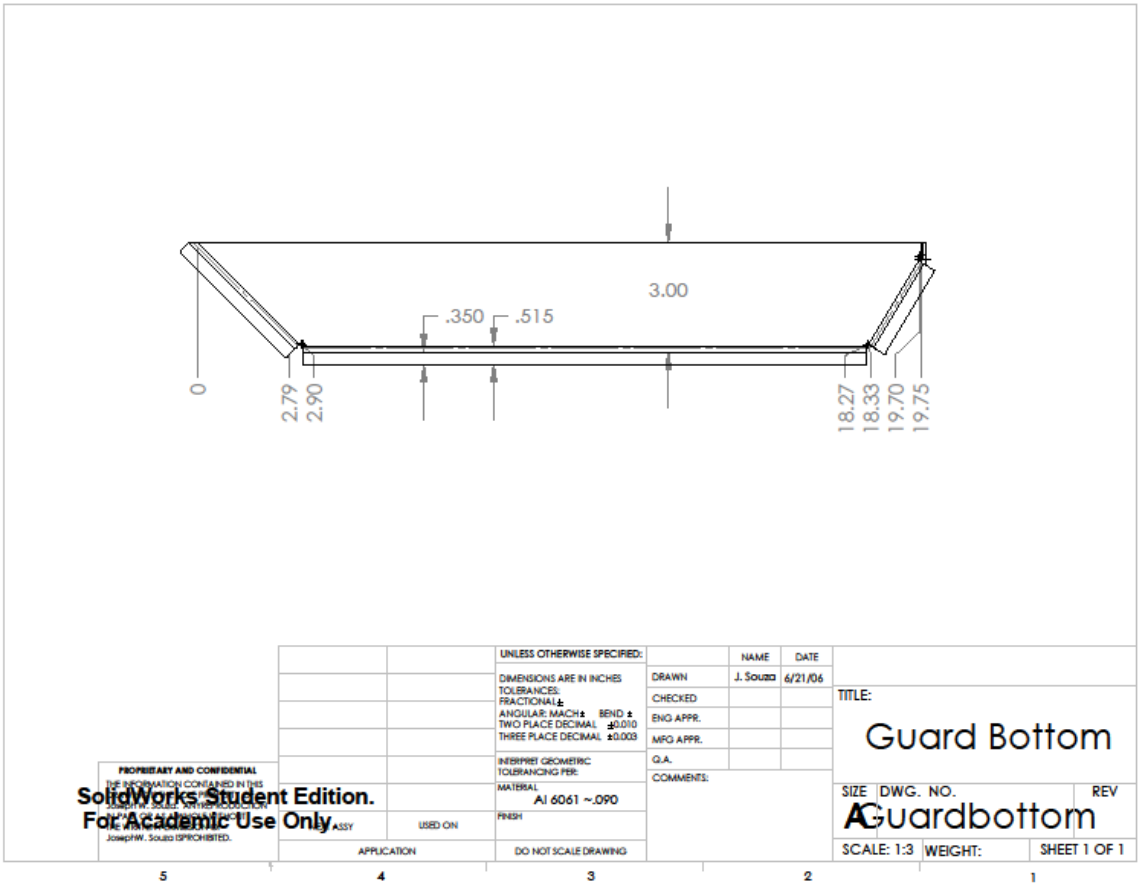
Tank Assembly

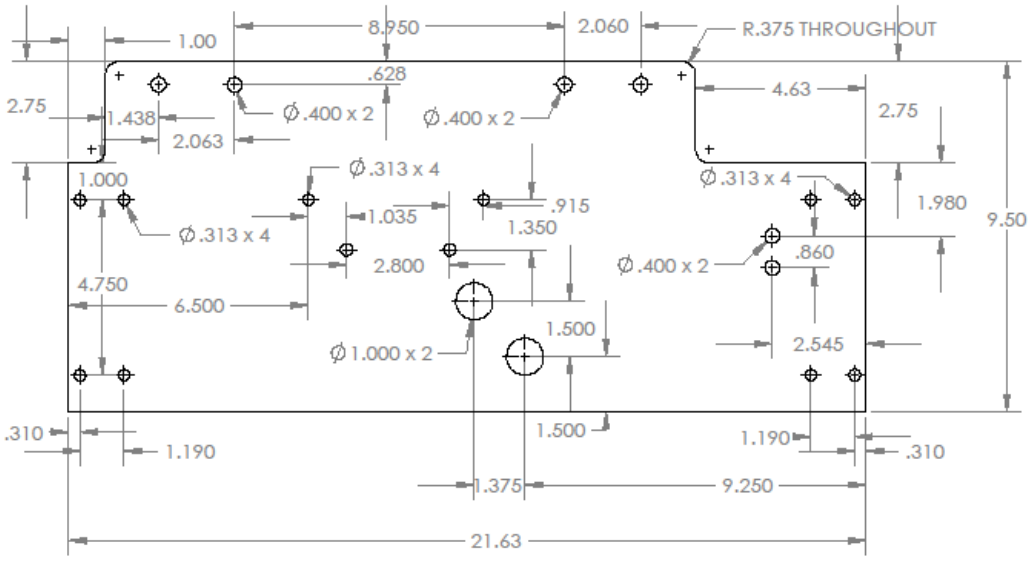
SIZE	DWG. NO.	REV
A	Sectank1	2

SCALE: 1: 20 | WEIGHT: | SHEET 1 OF 1

5 4 3 2 1







		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±0.02 THREE PLACE DECIMAL ±0.005	DRAWN	J. Souza	5/25/06
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL 0.100 in Steel	COMMENTS:		
	NEXT ASSY	USED ON	FINISH Painted (Black)		
	APPLICATION		DO NOT SCALE DRAWING		

TITLE:

Filter Plate

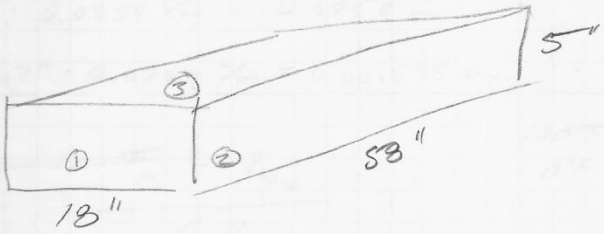
SIZE	DWG. NO.	REV
A	Filterplate	
SCALE: 1:3	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

Appendix H

Supporting Analysis

TANK HEAT LOSS



TANK VOLUME

$18'' \times 58'' \times 5'' \cdot \frac{.0043\ gal}{1\ in^3} = 22.59\ gallons = 0.0855\ m^3$

TANK SURFACE AREA

$5'' \times 0.0254\ ^{IN}/_{m} = 0.127\ m$
$18'' \times 0.0254\ ^{IN}/_{m} = .4572\ m$
$58'' \times 0.0254\ ^{IN}/_{m} = 1.4732\ m$

TANK SURFACE AREA

SIDE (1) $0.127_m \times 0.4572\ m = 0.058\ m^2$

SIDE (2) $0.127\ m \times 1.4732\ m = 0.187\ m^2$

SIDE (3) $0.4572 \times 1.4732\ m = 0.674\ m^2$

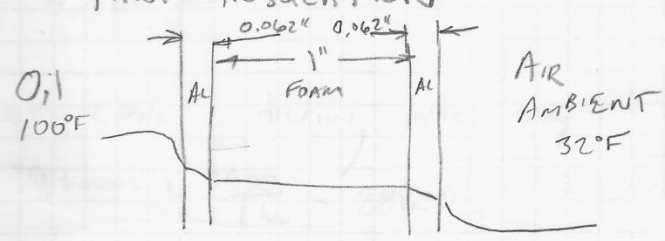
TOTAL SURFACE AREA

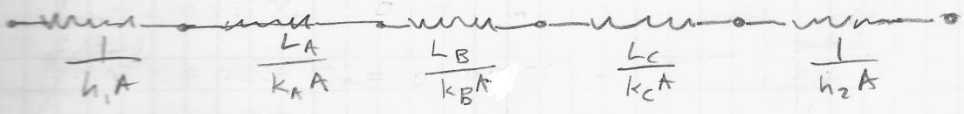
$2(0.058\ m^2) + 2(0.187\ m^2) + 2(0.674\ m^2) = S$

$S = 0.116\ m^2 + 0.374\ m^2 + 1.348\ m^2$

$S = 1.838\ m^2$

TANK INSULATION





$1'' \cdot 0.0254\ ''/m = 0.0254\ m$

$0.062'' \cdot 0.0254\ ''/m = 0.001575\ m$

$100°F = 310.928°K$

$0°F = 273.15\ °K$

$$q = \frac{T_{\infty_1} - T_{\infty_2}}{\Sigma R_t}$$

$h_1 = 1000\ \frac{w}{m^2 k}$

$h_2 = 2.0\ \frac{w}{m^2 k}$

$k_A = 239\ \frac{w}{m^2 \frac{k}{m}}$

$k_B = 0.0231\ \frac{w}{m^2 \frac{k}{m}}$

$k_C = 239\ \frac{w}{m^2 \frac{k}{m}}$

$$\Sigma R_t = \frac{1}{h_1 A} + \frac{L_A}{k_A A} + \frac{L_B}{k_B A} + \frac{L_C}{k_C A} + \frac{1}{h_2 A}$$

$$\Sigma R_t = \frac{1}{A}\left(\frac{1}{h_1} + \frac{L_A}{k_A} + \frac{L_B}{k_B} + \frac{L_C}{k_C} + \frac{1}{h_2}\right)$$

$$\Sigma R_t = \frac{1}{1.838 m^2}\left(\frac{1}{1000\ \frac{w}{m^2 k}} + \frac{0.001575 m}{239\ \frac{w}{m^2 k/m}} + \frac{0.0254 m}{0.0231\ \frac{w}{m^2 \frac{k}{m}}} + \frac{0.001575 m}{239\ \frac{w}{m^2 \frac{k}{m}}} + \frac{1}{2.0\ \frac{w}{m^2 k}}\right)$$

$$\Sigma R_t = \frac{1}{1.838 m^2}\left(.001\ \frac{m^2 k}{w} + 6.59 \times 10^{-6}\ \frac{m^2 k}{w} + 1.0996\ \frac{m^2 k}{w} + 6.59 \times 10^{-6}\ \frac{m^2 k}{w} + 0.5\ \frac{m^2 k}{w}\right)$$

$$\Sigma R_t = .5441\left(1.6006\ \frac{m^2 k}{w}\right)$$

$$\Sigma R_t = 0.8708\ \frac{k}{w}$$

$$q = \frac{310.928°K - 273.15°K}{0.8708\ \frac{k}{w}}$$

$$q = \frac{37.778 k}{0.8708\ \frac{k}{w}}$$

$$q = 43.38\ w$$

HEAT LOSS during work day

$8\ hours \cdot \frac{3600 s}{1 hr} = 28800\ s$

$q = 43.38\ w = 43.38\ \frac{J}{s}$

$Day_{Loss} = 43.38\ \frac{J}{s} \times 28800 s$ w/ 70°F Temp diff

$Day\ Loss = 1249344\ J$

$Mass = V\rho$ $\rho_{oil} = sg\ \rho_{H_2O}$ $\rho_{H_2O} = 970.87\ kg/m^3$

$sg_{oil} = 0.9$

$\rho_{oil} = 0.9(970.87\ kg/m^3)$

$\rho_{oil} = 873.783\ kg/m^3$

$V = 0.08551\ m^3$ see page 1

$m = V\rho_{oil}$ $Cp = 1.909\ \frac{kJ}{kg\,k}$

$m = 0.08551 m^3 (873.983\ kg/m^3)$

$m = 74.736\ kg$

$HE = m\,cp\,(\Delta T)$

$HE =$

$1249\ KJ = 74.736\ kg\left(1.909\ \frac{kJ}{kg\,k}\right)(\Delta T)$

$1249\ kJ = \left(142.671\ \frac{kJ}{k}\right)(\Delta T)$

$\Delta T = 8.75\ °K$

$T_F = 310.928\ k - 8.75\ k$

$T_F = 84.25\ °F$

Appendix I

Bill of Materials

Bill Of Materials

Major components only

Quantity	Description	Part#	Vendor	Cost
1	Oil Pressure Switch	OP6075	Napa Echlin	15.00
3	Solenoid Valve	25X00440GH	Peter Paul Electronics	225.00
2	Filter Housing	215R12	Racor	120.00
1	12V Bowl Heater	RK222350-01	Racor	45.00
1	215R 10 mic. Filter Element	R15T	Racor	25.00
1	215R 2 mic. Filter Element	R15S	Racor	25.00
1	Compact Heater Exchanger	LL520G12	Lytron	225.00
1	Gear Pump	GP-311-12	Reverso	420.00
1	Gear Pump	GP-212-12	Reverso	350.00
1	Xiphos Board	Xiphos	Cal Poly EE	125.00
2	Temperature Sender	TS6696	Napa Echlin	25.00
1	Fuel Level Sender	501-1742	Napa Echlin	40.00
1	Fuel Pressure Regulator 0-Psi	MAA-4309	Malory	99.95
1	Fuel Pressure Regulator 0-Psi	13109	Aeromotive	152.95
1	115V PID Heater Controller	STC-1000	Elitech	39.00
1	115V Blanket Heater	SRP12241	BriskHeat	215.00
1	Oil Pressure Sender	VDO-360003	VDO Gauges	41.00
			Total	**2187.90**

Appendix J

Original Gantt Chart

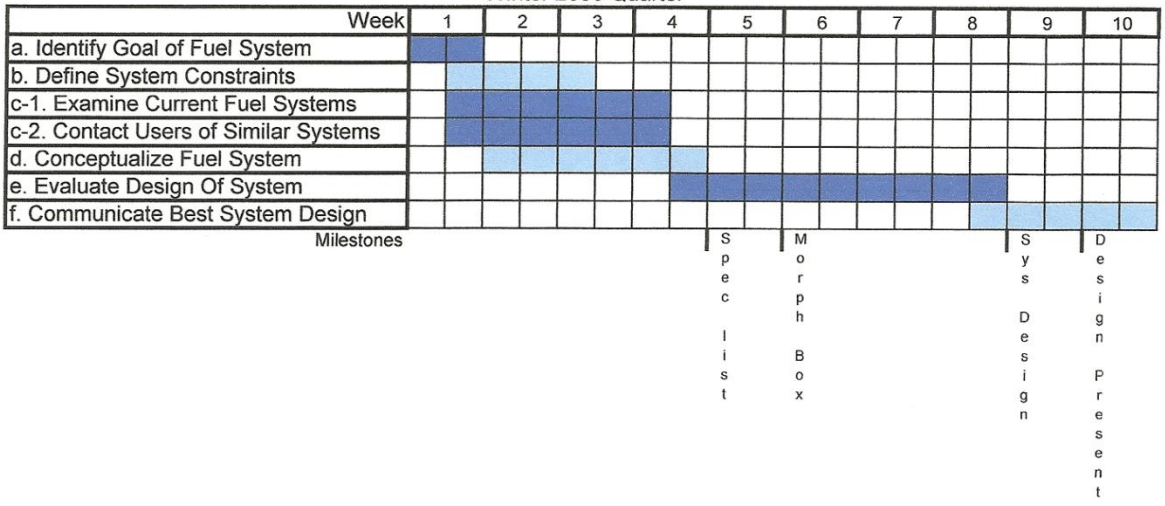
SVO Diesel Fuel System

Gantt Chart

1/15/2006

Joseph W. Souza

Winter 2006 Quarter



SVO Diesel Fuel System

Gantt Chart

1/15/2006

Joseph W. Souza

Spring 2006 Quarter