FAMU-FSU College of Engineering
Department of Electrical and Computer Engineering

Final Report

EEL4911C – ECE Senior Design Project I

Solar Car

Team # 2

Student team members:

James Barge, Electrical Engineering (jb09d@fsu.edu)
Adrian Cires, Mechanical Engineering (ac06e@fsu.edu)
Keith Dalick, Mechanical Engineering (kjd07c@fsu.edu)
Nelson German, Industrial Engineering (ng09@fsu.edu)
Emiliano Pantner, Mechanical Engineering (ep07c@fsu.edu)
Rajat Pradhan, Industrial Engineering (rdp08@fsu.edu)
Zachary Prisland, Electrical Engineering (zap04@fsu.edu)
Shishir Rajbhandari, Electrical Engineering (sr07k@fsu.edu)
Amanda Roberts, Industrial Engineering (akr06@fsu.edu)

Senior Design Project Instructor:

Dr. Chris Edrington
Dr. Bruce Harvey
Dr. Zohrob Hovsapian

Submitted in partial fulfillment of the requirements for
EEL4911C – ECE Senior Design Project I

April 7, 2011

Project Executive Summary

The energy from solar radiation is the most abundant and potentially the greatest source of renewable energy. Research is constantly conducted around the globe aimed at increasing solar cell efficiency and may one day enable us to harness the full energy of the sun. The technical design project that we have undertaken is attempting to introduce senior engineering students to solve the problem of designing, building, and racing a safe and functional car that is powered via sunlight.

The objectives of the technical design project are as follows:

1. Design a composite body
2. Design Solar array configuration
3. Design suspension system
4. Design Electrical system
5. Optimize Design
6. Test Mechanical system
7. Test Electrical system

The solar car project will be designed following lean six sigma's methodology DMEDI (Define, Measure, Explore, Develop, and Implement). DMEDI is a methodology used to systematically conduct projects that require a new designed process or product. The Define phase provides a clear problem statement that charters a project with a defined scope and Outcomes. The Measure phase is the step where the team converts the needs and specifications of the project into measurable and quantifiable targets. This allows for prioritizing and quantitative reasoning for making decisions or creating alternatives. In the Explore phase the team will then create a conceptual design of the solar car based on the data collected and analyzed in the measure phase. Then in the Develop phase the team optimizes the conceptual design to capture all the needs and specifications of the solar car. Finally, the solar efficient car will be fabricated into a full scale working design.

Table of Contents

1 Introduction

1.1 Acknowledgements

The 2010-2011 Solar Car design team would like to thank all of the people and organizations helping us to design and create a cutting edge solar powered vehicle. Particularly, the team would like to thank Dr. Bruce Harvey for giving the team direction on how to approach the task of designing and building this vehicle. The team would like to thank Dr. Chris Edrington, for the technical direction for the electrical integration, High Performance Material Institute (HPMI) and Jerry Horne, for the time and know how to create the composite body.

Capital donations were made to the project form the following organizations: FAMU-FSU College of Engineering, Hexcel Corporation, SolidWorks Corporation, Wilwood Engineering Inc., IESES, HPMI, Flex Solar Cells, MSC Software Corporation, and Signs Unlimited. Without the donations provided by all of these groups it is possible that the project would not have been completed and therefore overall success of the project can be contributed in part to all of these organizations listed above.

1.2 Problem Statement

In an effort to continue the work of the 1997 FAMU-FSU solar car project a team was assigned to begin redesigning the solar car system in 2009. Using the old vehicle as a starting point and a guide towards the future efforts, the project was brought to life anew. This first year group began by stripping away many of the suboptimal and frivolous elements which were drastically reducing the efficiency of the solar car as a whole. This included the overly heavy fiberglass body and lead acid battery energy storage, which alone created an excessively heavy and inefficient vehicle. The other accomplishments of this first phase of the project included, suspension redesign, electrical system redesign, and changing the configuration from four to three wheels. Leaving the car once again as a rolling chassis, basically an electric car and the project would be continued in the following year by Phase II.

Original goals for the project included making an attempt to race in the American Solar Challenge, a long distance competition involving many schools around the world. However when design work began it became quickly apparent that the race would be nearly impossible due to one of the most constraining elements in engineering, budget. The American Solar Challenge will still be a long distance goal for the school as a whole, but for purposes of this design phase, it would not be the overall constraining factor in the success of the project. Utilizing the race regulations as a guideline for the engineering standards of the vehicle, the team could begin its design work.

The design for this project began with a new body, one made of carbon fiber. Carbon fiber technology will allow the entire car to be made from a single piece, removing the need for a steel frame, which will drastically reduce the overall weight of the car. During the design of this monocoque body, the aerodynamic performance will greatly improve as well, which as investigated later will reduce the power use of the completed vehicle.

As the weight of the vehicle is adjusted, improvements to the suspension and braking will need to be made. The overall weight goal of the vehicle, including a 180 lbs driver, is 600 lbs. This is used as the design weight for the vehicle, the weight by which all mechanical components will be selected and built.

Electrically, the goal for this portion of the project will be to expand the system, introducing many new elements, primarily targeting increasing electrical safety and allowing expansion for solar collection and storage. The largest

components from the previous phase that will be the basis for the electrical system design are the batteries, battery management system (BMS), motor controller, and motor. Similar to how the mechanical components will design to the weight of the vehicle, the electrical system will be designed to these devices.

1.3 Operating Environment

Since the car will usually be located in or around the FAMU-FSU College of Engineering, the assumed operating environment will be outside Tallahassee. This means the car will be exposed to all weather and environmental effects commonly associated with northern Florida. This includes very high heat and humidity to slightly freezing temperatures. Since the tires do not possess tread, driving in the rain would be unsafe and should be avoided whenever possible. Analysis or testing has not been performed at high speeds and it is also recommended that the driver not exceeds speeds of 40 mph.

1.4 Intended Use(s) and Intended User(s)

The solar car will be an eco-friendly way for a single driver to traverse distances with the normal speed and efficiency of a car. The car will be equipped with all the normal lights and signals of a regular vehicle and therefore should be able to safely travel on roadways throughout a city. The vehicle's top speed will prohibit it from travelling on any interstate highways or any other roads with high speed limits.

The solar car will be used primarily for daytime driving as this is the only way to collect the solar radiation necessary to charge the batteries. The vehicle will be capable of charging the batteries from certain wall sockets so it will not be entirely restricted to driving during the day, but as stated previously, will have no way to recover energy except stopping again to charge.

This project will continue on after this portion of the design is completed in the hopes that it will be able to compete in the American Solar Challenge. This challenge is a competition that occurs bi-yearly and will give the finished product a chance to compete against other schools with similar design restrictions. To enter this competition will be the primary goal of this car as it progresses through design projects.

1.5 Assumptions and Limitations

1.5.1 Assumptions

1. Many of the electrical systems from phase one portion of the design will be useable in the design work for this phase
2. The car will be allowed to be to carry a full charge before any competition, which may be achieved through wall charging
3. There will be changing race restrictions for future races and the car will have to be left in a state where systems can be changed cheaply and simply

1.5.2 Limitations

1. Budget will be restricted and it will be necessary to seek donations wherever possible
2. The solar array will be limited to a size of 6 m^2
3. The driver's eye line must be at least 70 cm off the ground and provide 100 degrees of view to the right and the left

4. A roll cage will be protecting the driver in the event of a rollover collision
5. The electrical systems must be isolatable so that power can be immediately cut by either the driver or an onlooker from outside the car
6. The car will have to pass a series of safety and performance tests outlined in the American Solar Challenge guidelines and the finished project of this phase should have a car capable of passing all these tests

1.6 Expected End and Other Deliverables

The most important deliverable will be the completed solar car from this phase of the project. This will not be delivered until the end of the project time as will be illustrated below in the schedule. The other deliverables for the project will include several design papers which will include updates as to the current design and any modifications made from previous reports. A website will be created for the purpose of displaying information about the solar car, progress to date, and will include a section for all the papers and presentations. Finally a user manual for the safe operation of the vehicle will be completed.

2 Systems Design

2.1 Overview of the System

Due to the amount of exposure to vehicles in today's society the top level design of the solar car is fairly fundamental. There are basics that every 'car' has that will also be required in the end result of the solar car. The car will need a means of motion for not only the vehicle but also for the driver. Although motion is a good start it is almost worthless if the motion cannot be controlled and directed. The control means that the driver has to be able to slow down and speed up as desired and also has the ability to change the direction of travel. It would also be ideal for the driver to have information about this travel and control readily available (i.e. speedometer). As it is a solar car it will need a means of power generation through the sun's radiant energies. This energy will have to be stored at times because the driver may want to move the car during times when solar radiation is not available. While this is far from a summation of the goals that the solar car will need to achieve it is a basic overview of the standards to which the car will be held.

2.2 Major Components of the System

Due to the magnitude of the project it has been broken up into two sections, mechanical and electrical, mainly as a means to describe the functionality of the system or component and further dictate the primary party responsible. Each section has been further subdivided into multiple systems to allow the design work to be placed in the hands of a specific engineer on the project.

The mechanical system of the car includes the body, steering, braking, and suspension. The body of the car will be just that, the housing for all components as well as structural support for the entire vehicle. As can be inferred, the steering will be the system that will provide directional control over the vehicle with driver input. The braking system will be the means by which the driver slows the car down. Finally the suspension will be the system that will further facilitate driver control over the vehicle and provide a smoother more comfortable ride for the driver.

The electrical systems that were modified during this phase of the project include the power generation system, control system, and management system. The power generation system refers to the two methods of obtaining energy in the system, through the use of the solar panels and through the regenerative braking. The control system will refer to the control devices in the car, namely the microcontroller and dashboard interface for the driver. Finally the management system will be the safety devices in the car which will protect the driver and also the equipment.

2.3 Performance Assessment

Each of the systems will have to be evaluated to ensure not only correct performance but ideal performance under the wide variety of conditions that the car will be exposed to. This evaluation will begin during the design phases and continue through fabrication and implementation through testing.

The body of the car will be graded against three major standards: aerodynamics, strength, and weight. Due to the very low efficiency of utilizing solar energy everywhere power can be saved will be absolutely necessary. For this reason the aerodynamics of the body may be one of the most important design phases for the entire project. Through the use of CAD programs, different types of models can be tested and the overall drag coefficient for the vehicle can be calculated. The strength of the car will also be incredibly important as if the body breaks it could not only cause serious damage for the components but also to the driver of the car. However since the motor will have to propel all the weight of the car, it

will also be necessary to keep the body as light as possible. The material of the body will have to be chosen on these two factors.

The steering of the car will provide the direction control over the motion of the vehicle. Since the preliminary goal of the project was to compete in the American Solar Car Challenge (ASC), it will be used as a guideline to measure performance throughout the project. This is not the design limit for the car however and more control would be ideal unless if comes at the cost of friction or other power losses. Figure 2.3.1 displays one of the steering tests (the slalom test) that the car must undergo to qualify for the competition.

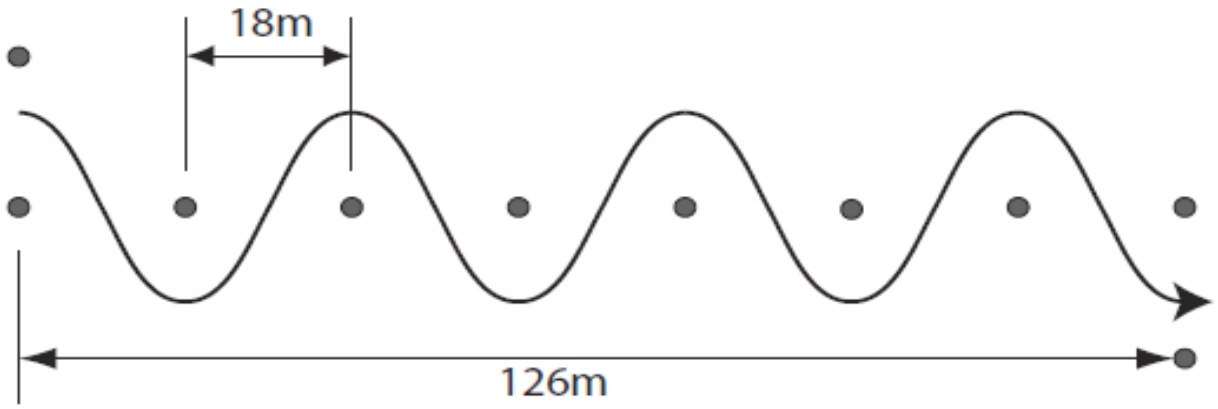


Figure 2.3.1 – Slalom Test Diagram

Similar with the braking system, the ASC has stipulated requirements on quickly the car must decelerate (4.72 m/s^2), but that is in no way a limit. Therefore the design of the braking system will be to meet the needs of the competition and if it is possible to efficiently surpass these bounds than the system will be designed as such.

The suspension of the car will provide some impact protection for various road conditions that the car will encounter during operation. This protection will keep the body from unnecessary damage and also provide the driver a more comfortable ride. Once the final weight and size of the vehicle is determined there will be a maximum displacement for the suspension to be designed around.

The power generation system will also be limited by the ASC regulations, allowing a maximum of 6 m^2 of surface area for the solar arrays. The performance of these solar cells will be measured by the fill factor, the ratio of theoretical power to actual power, solar efficiency, and thermal efficiency. Along with the solar cells, power point trackers (PPT) will need to be utilized as well. PPTs will be chosen based on the power rating and the overall efficiency with the solar cells utilized.

The control system's primary component will be the microcontroller. The microcontroller's performance will be determined by comparing the needs of the project to the specifications of the microcontroller. When a few microcontrollers are found that will meet the needs of the project, with a little extra for possible design changes, then the search for the most cost effective one will take place.

The management system's performance will be based upon how well the state of charge information is monitored, both through the BMS and also the information displayed for the driver. It will also be important to verify that the

information seen by the driver will correspond to the information that will be seen by the BMS, primarily as a means of checking that both systems are working properly.

2.4 Design Process

The solar car team has been provided with three industrial engineers to act as subcontractors for the project, allowing for a wider variety of design input. The industrial engineers were able to greatly facilitate the design process through their methodologies as shown in Figure 2.4.1 and Figure 2.4.2.

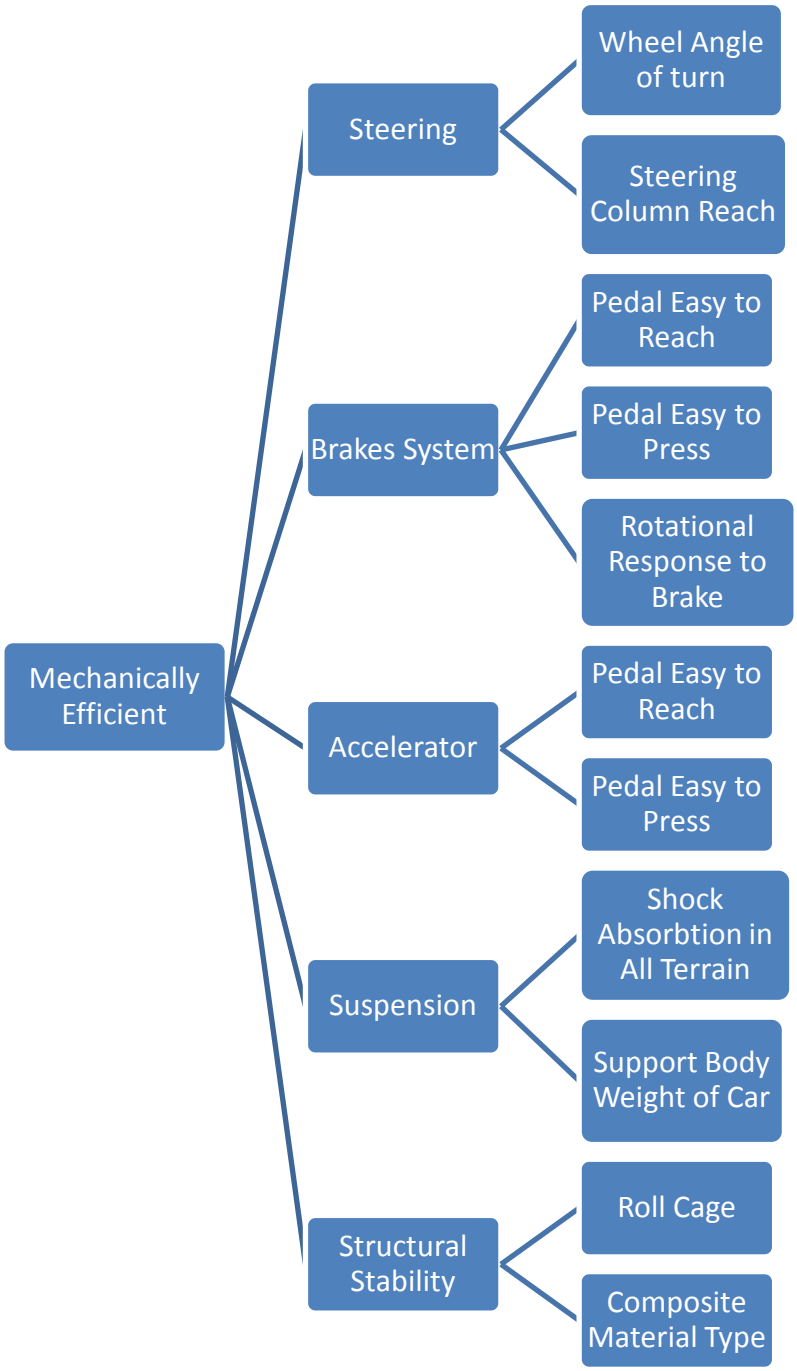


Figure 2.4.1 – Mechanical System

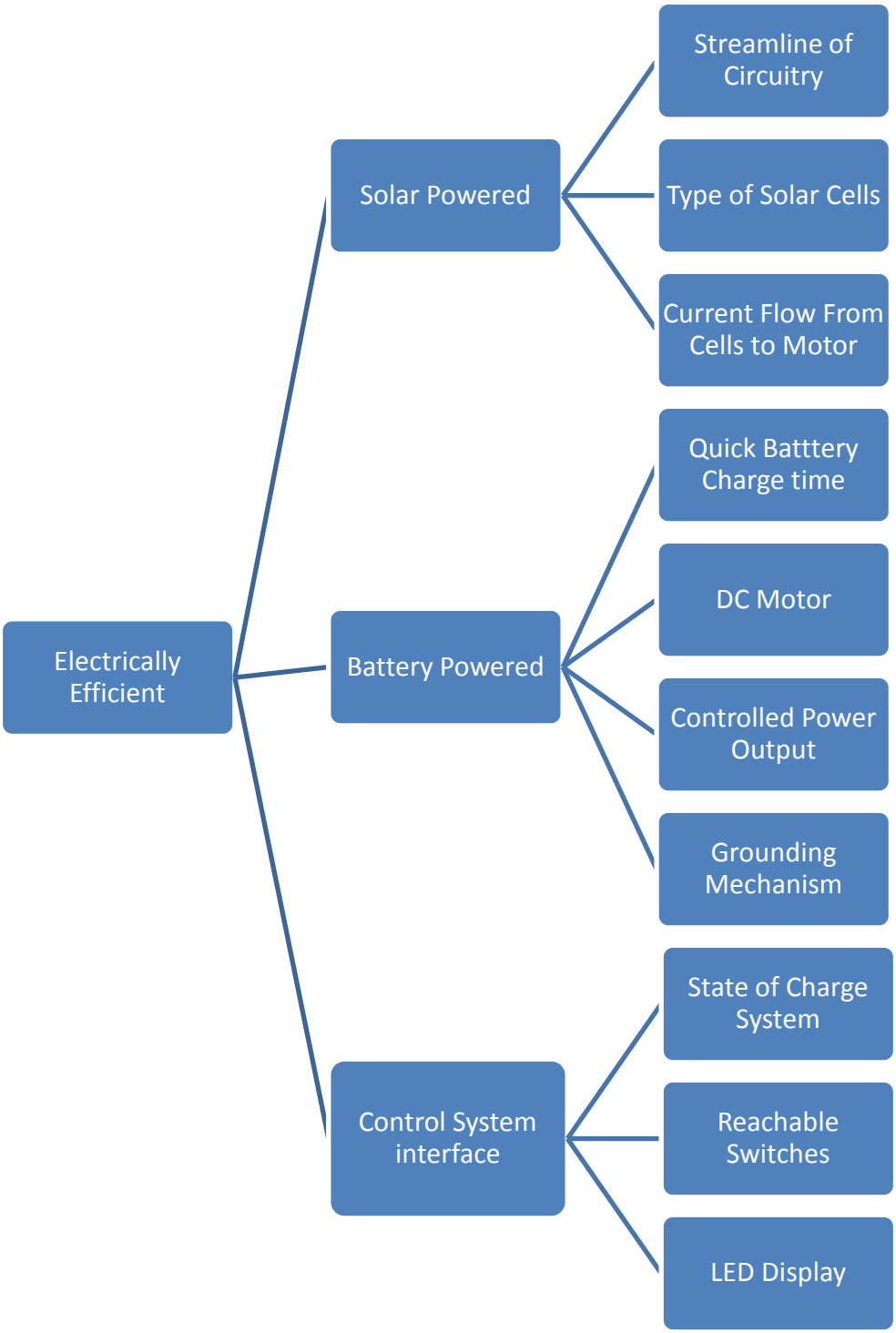


Figure 2.4.2 – Electrical System

These figures helped ensure that all factors were taken into consideration while going through the other design stages. It was this process that helped to create the major component areas for our project.

For the power generation system the most important decision was the type of solar cells that would be utilized in this design project. The options that were considered were crystalline and amorphous silicon. The major differences between the two types of cells are discussed more thoroughly below in Section 3.5.1. While budget will not allow for the

purchase of a maximum power point tracking device (MPPT), it will be discussed briefly as the integral component between any solar power source and any other electrical system. Finally some changes will be made to the regenerative braking in order to increase utility and functionality of the driver.

The control system had to make a decision about which microcontroller would meet all the needs of the design project. It was important to keep in mind that if the capabilities of the microcontroller greatly exceeded the needs of the project it would most likely cost more money as well. The microcontroller chosen for the project is the Dragon12 Development Board, which contains more than enough I/O pins for the entirety of the project. The programmable memory on this board is rather large which should surpass the simple application for the dashboard displays.

Finally the management system of the car will describe all the power management throughout the car such as relays, fuses, and battery management system (BMS). All of these devices will interact with the various elements of the control system in order to create a safe and reliable system. Safety will be important for not only the driver but also the equipment itself. To this end, the AWG (American Wire Gauge) standard was referenced innumerable times throughout the design and actual fabrication to ensure that a seemingly innocuous mistake was not made in choosing correct wire sizes.

As this phase of the project comes to a close all the final design decisions have been made. These decisions were all made keeping in mind that the project will continue to be passed to other students throughout the years and it will be important that the system be left in a state such that it can be easily expanded. Any short sightedness in this aspect may prevent the successful completion of future endeavors.

2.5 Overall Risk Assessment

There are two major risks to the overall success of the project. They include a budget risk which is almost inevitable, but like most engineering projects there is never as much money as the individuals working on the project would like. The other major risk is having enough time to complete the project. While there is still over two months left in the project which would seem like plenty of time there is always difficulty with other coursework and schedules which will divide the time of each member of the team. The time risk will be all about each member managing their individual schedule to try to maximize the amount of time that they can spend working on the project. Besides these two major risks, which are risks to almost any project in any field (not just engineering), there are wide variety of technical and safety risks involved with the project.

The biggest safety risk will be working with the batteries. As the batteries have a huge amount of energy stored in them an accidental short circuit is very dangerous if not lethal. When performing any of the work with the batteries or wiring a member will have to be very careful and should wear rubber gloves if possible. The other safety risks will involve when the car is actually in operation. The driver will have to be able to get out of the car quickly in the event of an electrical fire. This risk can be overcome by designing the body of the car to separate as easily as possible. The other risk will be operating the vehicle on roads. Despite the length and width dimensions of the car being quite large, the short height of the car may make it difficult for other motorists to see the solar car. To prevent an unnecessary risk a chase vehicle will be needed whenever driving the car on the street.

The technical risks for this project all involve designing a system or integration incorrectly. Due to the magnitude of the project there is a higher potential of error. These risks will be combated by having other project members, with similar

technical skills, review any design work produced. During the fabrication phase it will also be necessary to have multiple team members present to verify that a design is being implemented correctly.

3 Design of Major Components

3.1 Body

The design for the body of the solar car has many factors when designing an efficient body with very little frictional losses. When considering the design, the team has to keep in mind a few very important factors. The first is the safety of the driver, which must meet the race regulations. The race regulations state that the driver must be encapsulated in a roll cage for rollover protection. Dimensioning the vehicle to fit the roll-cage has to be considered. The next factor that has to be considered is the overall shape of the vehicle to keep air resistance at a minimum. Frictional loss from air resistance can be a huge variable when driving at speeds reaching 70 mph. The final factor that must be considered is the weight of the vehicle. The race states that the driver must weigh 80kg, making this the minimum the total car can weigh. When considering rolling frictional loss in the tires, the main variable is the downward force between the tires and the road, also referred to as the overall weight of the vehicle. This design must be drawn in SolidWorks CAD software to be analyzed for structural integrity.

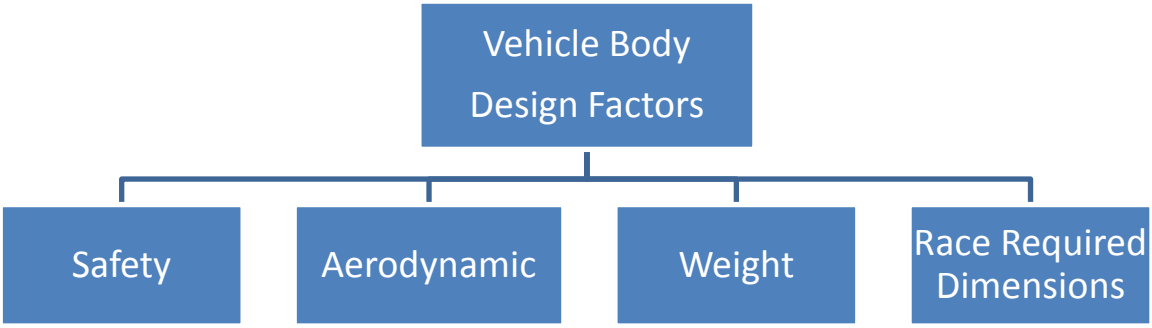


Figure 3.1.1---Vehicle Body Design Factors

3.1.1 Safety

According to the race regulations, the solar car must have a roll-cage to protect the driver in a crash situation. To achieve the maximum amount of safety, a roll cage must be designed in conjunction with the body to provide for the most effective design. When considering the roll-cage, the team had to decide between a cage that has double bars over the drivers' helmet or a design using one bar over the drivers' helmet and one bar over the lap of the driver. The factors for deciding between the two designs are the effectiveness, whether it is easy to escape the car and the weight comparison. The team chose to use the design of the double bars over the helmet of the driver because it would be much easier to escape from the car in the case of an accident or fire. The bars will be made from chromoly tubing because it is lightweight and very strong.

3.1.2 Body Shape

The body of the Solar Car must have as little air drag as possible. This makes for a more streamline design reducing the force it takes to cut through the air. Through extensive research, the proposed design takes the shape of a water droplet falling through the air which is commonly known as the basic most aerodynamic object. At the bow of the vehicle, the

design mimics the parabolic shape of the bottom end of a water droplet. This is so there is no separation between the air and the body. When separation of the air and the body occurs, there is a pocket of low pressure air that acts against the direction the vehicle is moving. At the aft of the vehicle, the upper and lower halves of the body converge to a single line. This, again, is to reduce the separation of air from the body reducing the chance for the low pressure air pocket to be generated. When calculating the drag an object produces, the two variables that can be controlled are the cross sectional area (A) and the drag coefficient (C_D) of the vehicle, as seen in the *Equation 3.1.1*.

$$F_D = \frac{1}{2}\,\rho\,V^2\,C_D\,A \qquad \textit{Equation 3.1.1}$$

Other Solar Car teams have tried using different radical shapes, as seen in Figure 3.1.2, but the standard aerofoil shape, Figure 3.1.3 has been proven to work the best in the solar car application.

Figure 3.1.2 – Radical Design from American Solar Challenge 2010 (Änderung, 2009)

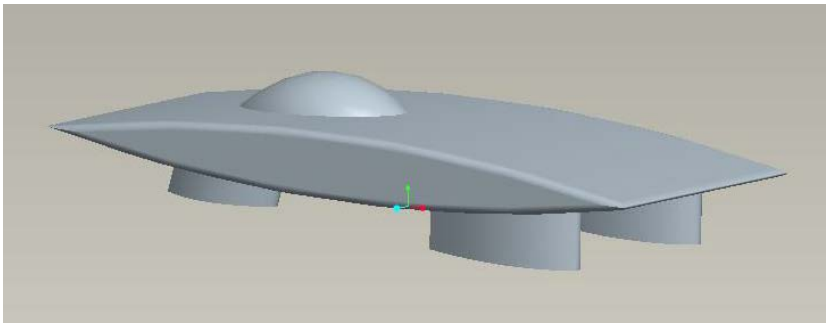


Figure 3.1.3 – Proposed Aerofoil Design

3.1.3 Body Weight

The overall goal of the car is to make the body and its components as light as possible. This is because frictional loss between the tires and the road and the frictional loss in the wheel bearings is a function of the weight. Weight is also known as the Normal force (N_f) exerted to hold the car above the ground. When determining the rolling friction lost between the road and the tires, *Equation 3.1.2* is used: where C_{rr} is the coefficient of rolling friction for Michelin solar car/eco-marathon tires.

$$F_r = C_{rr}\,N_f \qquad \textit{Equation 3.1.2}$$

For reduction of overall weight of the body, there are two major components that have to be considered, first of which is the design of the frame. Previously, solar cars were made using space age aluminum framing and covered with a fiberglass shell as seen in Figure 3.1.5. The use of a frame adds considerable weight making it less desirable.

Figure 3.1.4---Aluminum Frame with Outer Shell (Cyber, 1999)

The design chosen uses the idea of a monocoque construction which utilizes the shell of the body as the load bearing structure as seen in Figure 3.1.6. It eliminates the aluminum tubing frame making for a much lighter design.

Figure 3.1.5---Monocoque Body (Kruschandl, 2005)

The monocoque body can be made of either fiber glass or carbon fiber fabric. It is desirable to use the carbon fiber fabric because it is 40% lighter than fiber glass and much stronger. The only drawback is that it is considerably more expensive. The proposed design, Figure 3.1.7, is to use carbon fiber to make the solar car as light as possible to reduce the frictional losses.

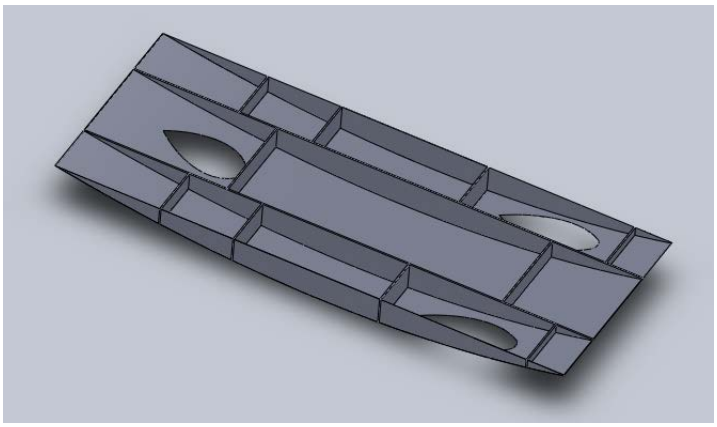


Figure 3.1.6---Proposed Monocoque Bottom Half

3.2 Steering

The steering system from the previous year's solar car will be salvaged and implemented into the current solar car design.

The major components of a rack and pinion steering system are the steering wheel, steering column, rack and pinion gear, and the tie rods. This is depicted in a block diagram in Figure 3.2.1.

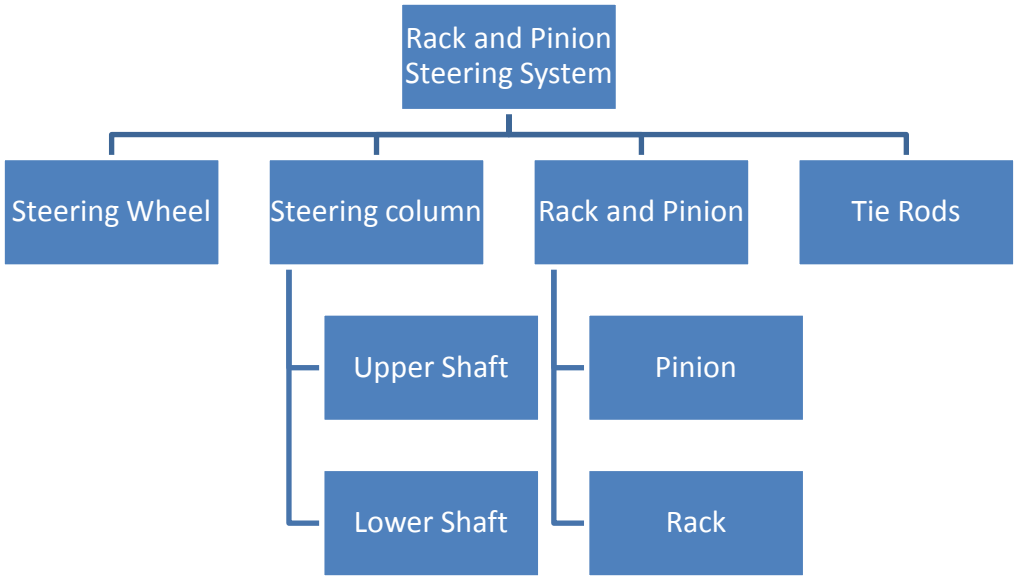


Figure 3.2.1 – Steering System Block Diagram

3.2.1 Steering Wheel

The steering wheel is the input device for the steering system. As the driver turns the wheel it rotates the steering column in order to turn the wheels in the desired direction.

3.2.2 Steering Column

The Steering column consists of two parts, the upper and lower shaft. The upper shaft is attached to the steering wheel so as the steering wheel is turned, the upper shaft rotates proportionally to the wheel. The lower shaft is positioned parallel to the road and is connected to the upper shaft using a universal joint. Shown in Figure 3..1 is an example of a steering shaft assembly.

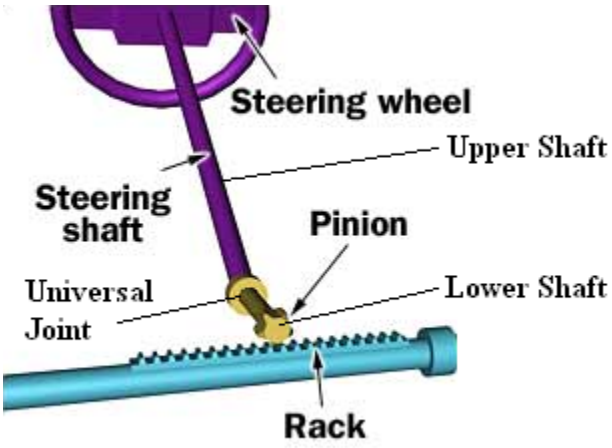


Figure 3.1.2.1-- Steering shaft. (Nice, How Car Steering Works, 2001)

Figure 3.1.2.2: Steering Column Mounted in Car

Figure 3.1.2.2 shows the steering column mounted onto a bracket.

3.2.3 Rack and Pinion

The rack and pinion, as shown in Figure 3.2.2.1 is the main component of the steering system. It consists of a pinion gear, which is attached to the lower shaft that is in mesh with a rack. The rotational motion of the pinion is converted to a linear motion by the rack gear. As the rack moves linearly it moves the tie rods which turn the wheels.

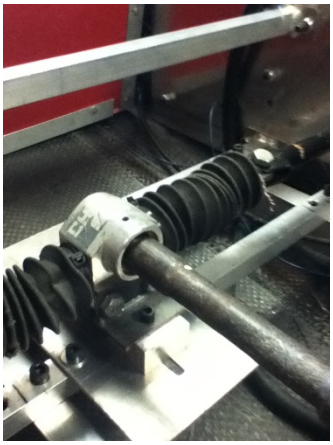


Figure 3.2.2 – Rack and Pinion Gear (Rack and Pinion)

3.2.4 Tie Rods

The tie rod, shown in Figure 3.2.3, in the steering mechanism is a thin, slender rod which connects the rack shaft to the steering arm of the system. This part of the system undergoes tensile loadings under the forces applied to it by rack. This force is transmitted by the tie rod to the steering knuckle or steering arm which connects to the wheel of the vehicle, moving the wheel in the desired direction.

Figure 3.2.3 – Tie Rods

The tie rod translates the force applied by the driver through the rack and pinion mechanism to the wheel's steering arms and are designed to withstand axial loading and stresses

3.3 Braking

When designing the braking system for the solar car, the rules and regulations of the American Solar Challenge were used as a design standard to start the design process. The solar car features a hydraulic disc braking system which will be implemented on the front two wheels of the solar car. None of the parts from the previous year's car were able to be salvaged due to the light weight of the new design. The new hydraulic braking system will feature a balanced, co-reactive, dual braking system. This is a safety precaution in the event that one system fails; the vehicle can still be stopped. In order to achieve this, each wheel of the vehicle will have its own master cylinder to supply brake fluid to

their respective brake calipers. Shown in Figure 3.3.1 is the block diagram for the hydraulic disc braking system. The main components of the system are the pedal system, master cylinder, caliper, and brake rotor.

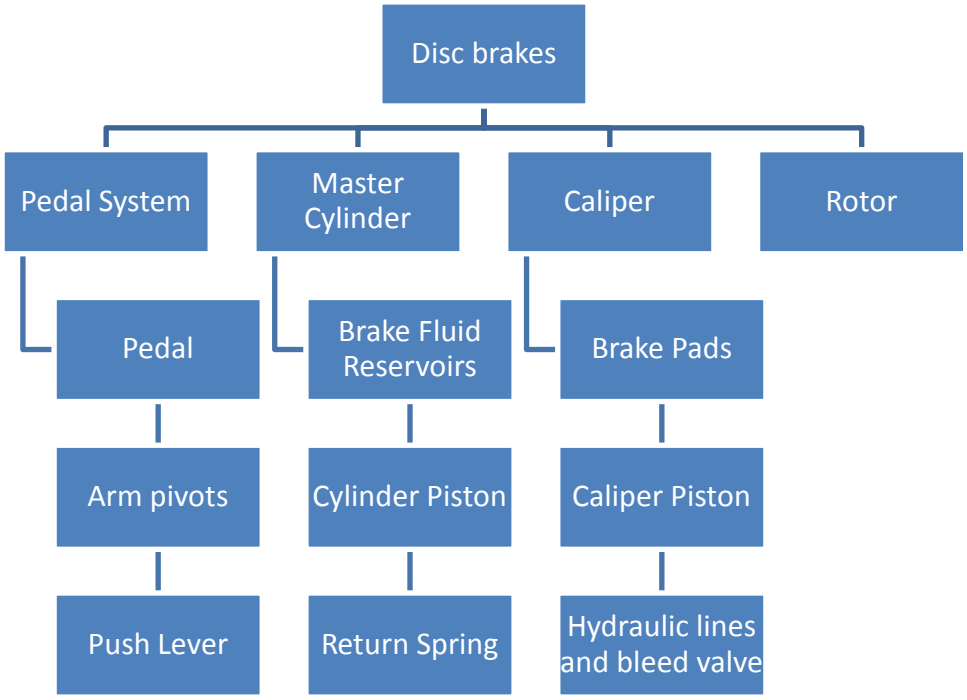


Figure 3.3.1– Hydraulic disc braking system block diagram

3.3.1 Pedal System

To initialize the braking system of the car, a brake pedal is installed in the car so when pressure is applied to the pedal it rotates the arm pivot around a point to activate the push lever, which is connected to the master cylinder and is responsible for applying pressure to the pistons in the master cylinder to drive the hydraulic fluid. Shown in Figure 3.3.2 is the force diagram of the pedal system.

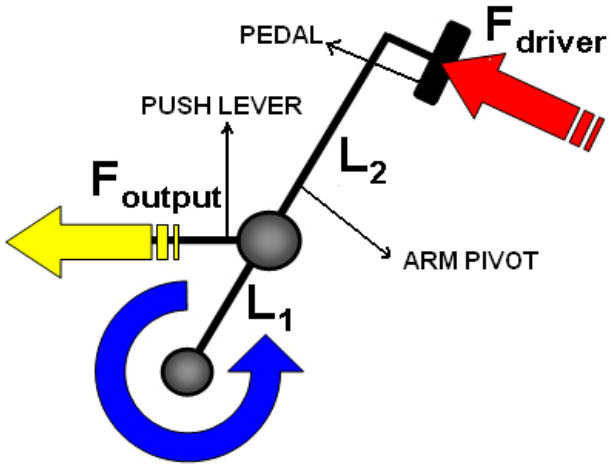


Figure 3.3.2 – Brake pedal system (Brake pedal setup)

The previous year's pedal assembly was not compatible with the new braking components that were installed, so a new pedal cluster was designed and installed into the car. The new pedal cluster is shown in Figure 3.3.3.

Figure 3.3.3 – Solar car's New Pedal Cluster

3.3.2 Master Cylinder

The master cylinder is a crucial component of a disc braking system. The master cylinder is a control device that converts the pressure from the push lever into the hydraulic pressure needed to stop the vehicle. The master cylinder is comprised of the main cylindrical body, which encases two pistons and two return springs, and a reservoir for the brake fluid. When the brake pedal is pressed it moves the primary piston. As the primary piston moves, hydraulic pressure builds in the cylinder and pushes a second piston. The built pressure from these pistons gets transferred into the brake lines which go to the respective brake caliper systems. When pressure is taken off the brake pedal, the return springs return springs bring both pistons back to their respective rest states relieving pressure in the master cylinder. Shown in Figure 3.3.4 is a schematic of a master cylinder.

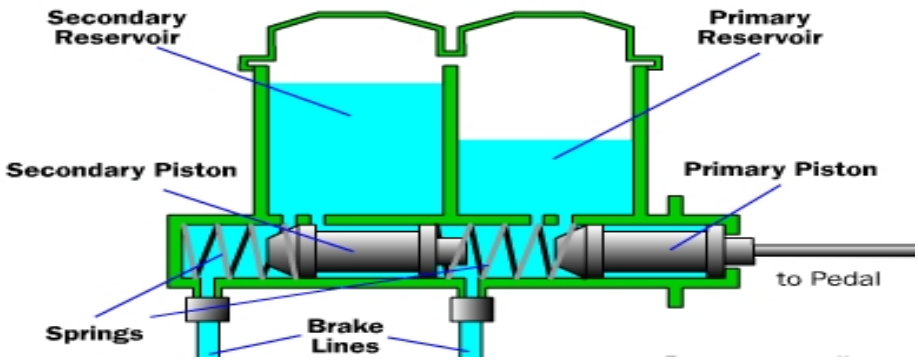


Figure 3.3.4 – Master Cylinder Schematic (Master Cylinder System)

Wilwood Engineering sponsored the solar car this year and has given the team various braking components to utilize in the cars design, one of these being the master cylinder. Due to the vehicles light weight, go kart master cylinders were used to supply hydraulic fluid to the brake caliper. These master cylinders are shown in Figure 3.3.5.

Figure 3.3.5 – Wilwood Go Kart Master Cylinder

3.3.3 Caliper

The actual device that applies the frictional force on to the rotor to stop the vehicle is the brake caliper. The brake caliper is an assembly that contains brake pads, caliper piston. The caliper fits over the brake rotor like a clamp. Inside the caliper there are frictional pads placed on both inside faces of the caliper. When pressure is applied to the brake pedal, brake fluid is sent from the master cylinder to the brake caliper causing hydraulic pressure on the caliper system. This hydraulic pressure on the piston forces the brake pads against the motor, which in turn stops the vehicle. Figure 3.3.6 shows a brake caliper assembly mounted on a rotor.

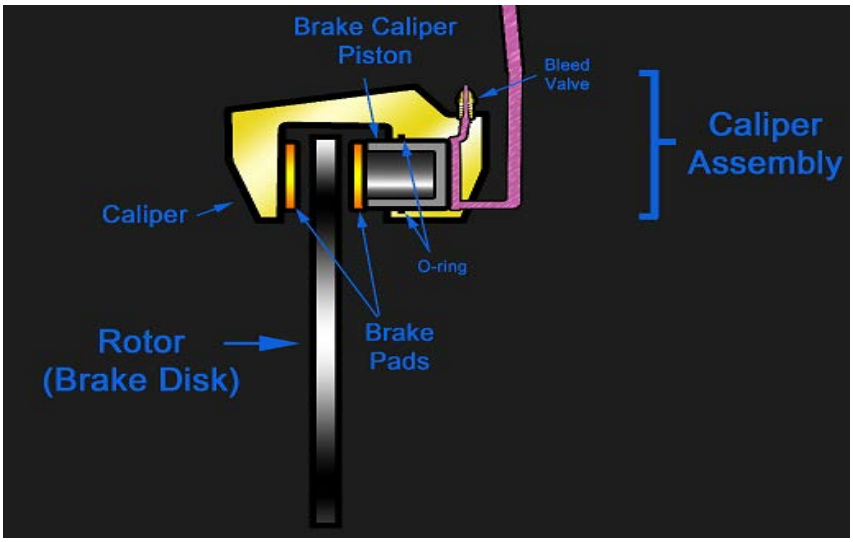


Figure 3.3.6 – Brake Caliper assembly (Hydraulic Brake Diagram)

The brake calipers from the previous year's car were not functioned properly due to the caliper pistons becoming jammed. New brake calipers were purchased from Wilwood Engineering to replace the malfunctioning brake calipers. These brake calipers are designed for go karts, but due to the light weight of the vehicle design, they were a perfect addition to the braking system. This caliper features a self retracting and adjusting piston system which enables the piston to retract as the brake line pressure is reduced. It also includes deep cup stainless steel pistons for reduced heat transfer. The caliper also comes with high performance and high friction brake pads. Figure 3.2.3.2 shows the caliper that is installed in the solar car.

Figure 3.3.7 – Wilwood Brake Caliper

3.3.4 Rotor

The rotor serves two purposes, the first of which is actually stopping the vehicle. As the brake calipers clamp onto the brake rotor, a frictional force is generated on the rotor in the direction opposite of the vehicles motion. This frictional force is what enables the car to stop or slow down. Another purpose of the rotor is to dissipate heat which is created as a result of friction. As friction is applied to the rotor, the kinetic energy of the moving rotor is converted to thermal energy. To help keep the rotor cool, rotors have cooling vanes machined in them to suck in cool air as it rotates. Wilwood did not have any brake rotors that could fit the spindle we salvaged, so a new rotor was designed. Figure 3.2.4. Figure 3.3.8 shows the new caliper mounted on the spindle.

Figure 3.3.8 – Brake Rotor Mounted on Spindle

The new rotor was fabricated using mild steel, which was recommended to by Wilwood Engineering. It has been drilled in order to help the rotor dissipate heat that is generated when friction is applied to it from the caliper assembly. In order to ensure the new design can withstand the pressure and heat from the caliper, they were analyzed using simulation tools on SolidWorks. Figure 3.3.9 shows a static analysis of the brake rotor as it acts from the pressure force from the caliper

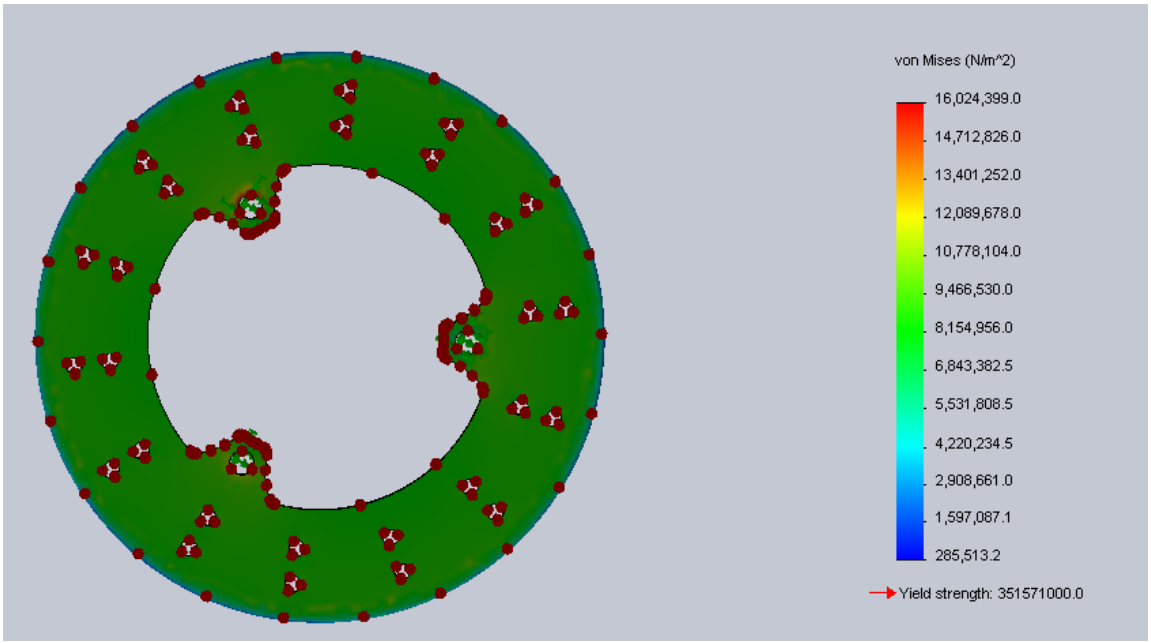


Figure 3.3.9 – Von Mises Stress Distribution through Brake Rotor

The pressure enacted on each face of the rotor was calculated to be 8.78e+6 Pa, as exerted by the master cylinder. It is clear from this analysis that the rotor design can withstand the clamping force of the brake caliper. When analyzing how the rotor reacts to a heat load a thermal analysis was also performed. Figure 3.3.10 shows the broke rotor can dissipate heat using natural convection of the air.

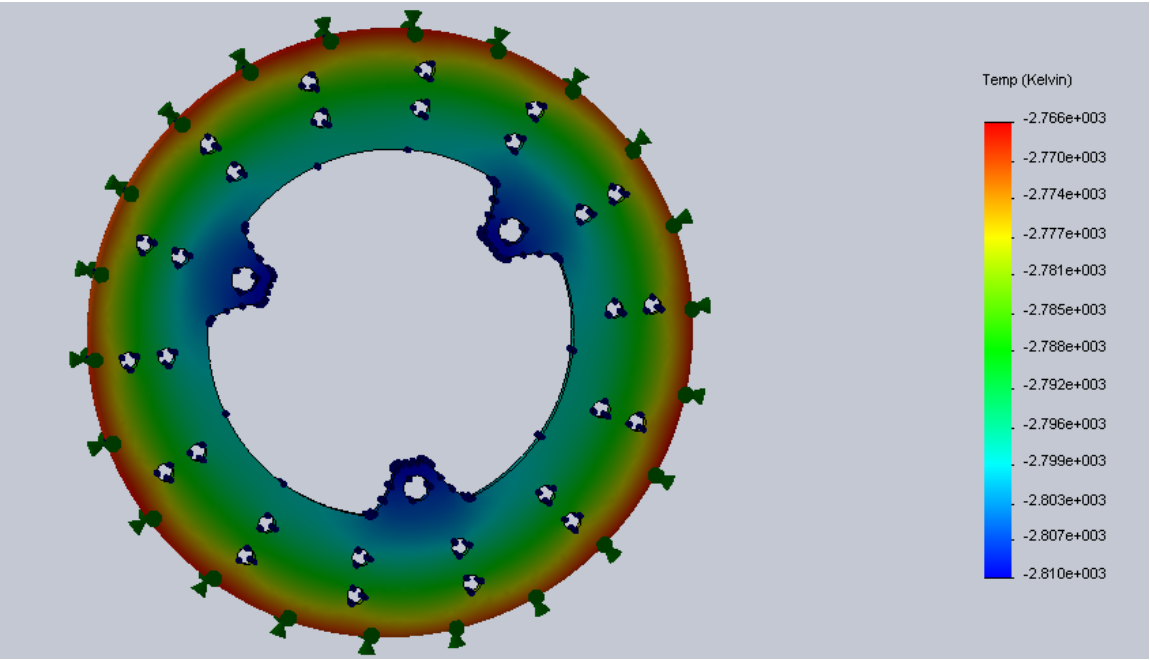


Figure 3.3.10 – Thermal Analysis of Brake Rotor

The brake rotor is to be cooled by natural convection as air flows through the rotor. Holes were drilled into the rotor in order to help induce the cooling of the rotor.

3.3.5 Brake System Selection

When selecting the braking system to use, a decision was made between the disc and drum braking system. Looking at the drum braking system, although it is a cheap system it can be complex and difficult to fix. The internal components of the drum brake can become inefficient when the brakes are applied repeatedly over a period of time. The drum brakes do not dissipate heat as efficiently as disc brakes do, so the efficiency of the drum brakes decrease drastically when heated. The disc brakes have the brake rotor exposed to open air so he can be dissipated efficiently without compromising the efficiency of the braking system. Overall a drum brake is cheaper than the disc braking system, however last year's solar car has various components which can be salvaged to reduce the cost of the system. Taking these considerations into account a decision matrix was constructed to aid in the decision making process. Shown in Table 3.3.1 is the decision matrix of the braking system.

Table 3.3.1 – Brake System Decision Matrix

Brake System Decision Matrix						
	Cost	**Efficiency**	**Durability**	**Complexity**	**Manufacturability**	**Total**
Disc brakes	5	4	4	4	2	19
Drum Brakes	2	2	3	2	3	12

From the decision matrix it was an obvious choice to go with the disc brakes over the drum brakes. When selecting the actual disc brake system to use, the required braking force for each tire is to be calculated. This can be done by using *Equation 3.3.1.*

$$F_{friction} = F_{clamp} * \mu_{bp} \qquad \textit{Equation 3.3.1}$$

Where F_friction is the frictional force the rotor applies to oppose motion, Fclamp is the force applied by the caliper clamp onto the rotor; the equation for F clamp is shown in *Equation3.3.2*, where μ_{bp} is the coefficient of friction between the rotor and the brake pad.

$$F_{clamp} = F_{cal} * 2 \qquad \textit{Equation 3.3.2}$$

3.4 Suspension

The suspension in the car will help maximize the friction between the tires and the road surface, and provide steering stability with good handling to ensure the comfort of the driver. Main components of a suspension, Figure 3.4.1, include spring, damper, control arms, and upright. Most suspension designs use a passive spring to absorb impact and a damper to control spring motion. A study found that humans perceive a ride to be comfortable when the bouncing frequency is 1 to 1.5 Hz; after 2Hz, most people feel the ride to be tough. Therefore, the ride quality is controlled by the selection of appropriate springs and dampers (Wan, 2000).

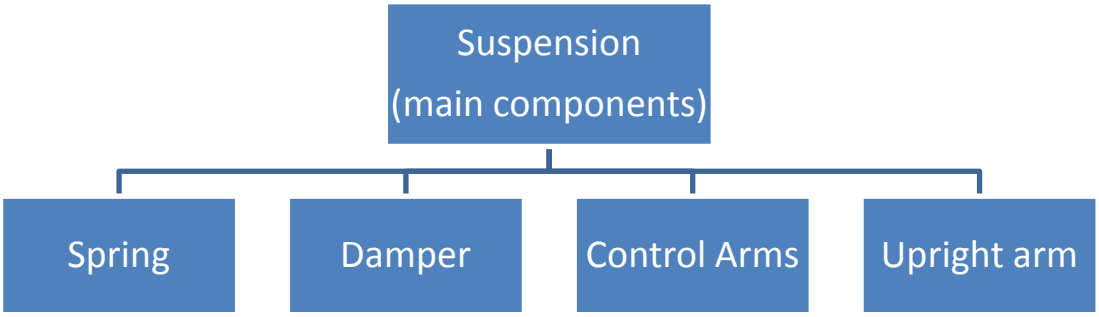


Figure 3.4.1 – Suspension Main Components

A car's suspension can be non-independent or independent. In a non-independent suspension, a rigid axle fixed is between the left and right wheels, and the body is suspended by leaf springs or coil springs on the axle. Consequently, the wheels are not independent and when one wheel rides on a hump, the shock is transferred to the other wheel. In contrast, in an independent suspension, the wheels' suspension systems are independent of each other (Shiota, 2010). This will provide the rider with a more comfortable ride isolating the vehicle by its points of contact from the road and eliminating the disadvantages of the beam axle. Some of these disadvantages include loss of friction by the wheels, small maximum spring deflection, no control of the steering system, and over-steer. Due to the advantages of an independent suspension system, the solar car will feature an independent suspension system for each of the three wheels.

Figure 3.4.2 compares an independent and non-independent suspension design. It shows a solid rear axle held by leaf springs for the non-independent suspension, and a spring and damper combination for the independent suspension design.

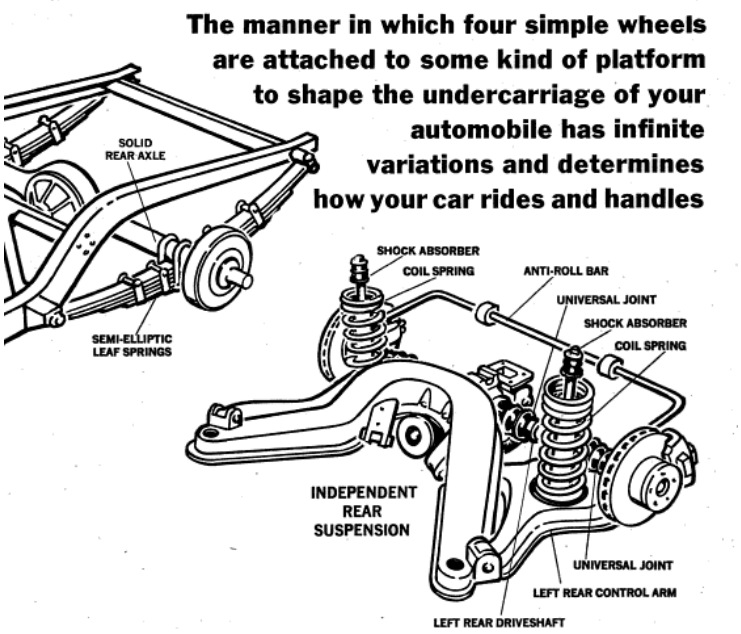


Figure 3.4.2 – Non-independent Suspension (Temple, 1969)

Important parameters to take in consideration in the suspension design include: spring rate, damping, travel, roll center height, and body dimension constraints. The spring rate or spring coefficient, k, is a ratio measuring how resistant a spring is to being compressed or expanded during the spring's deflection with units of lbf/in. or N/mm. Damping controls the movement of the car; un-damped cars oscillate, whereas a damped car settles back to the equilibrium state in a minimal time. A car's travel must be established to set the spring's displacement, *x*, and prevent the car from bottoming. Hooke's Law, *Equation 3.4.1*, can be used to calculate the force exerted by the springs.

$$F = -k * x \qquad \textit{Equation 3.4.1}$$

The roll center height is important to body roll and stiffness distribution for both front and rear of the car. Lastly, after analyzing the final design of the bottom shell of the car's body, points on the body and ribs will be chosen to connect the control arms of the suspension.

3.4.1 Front Suspension

The front suspension is linked to the steering system, thus some of the design parameters are constrained by the steering design. Two suspension designs, the MacPherson strut and double wishbone suspension systems, were analyzed and compared to choose the best fit for the front suspension. The MacPherson strut, as shown in Figure 3.4.3, is a simple system comprised of a strut-type spring and shock absorber combo pivoting on a ball joint on the single, lower arm.

Figure 3.4.3 – MacPherson Strut (Longhurst, 2010)

The telescopic shock absorber also serves as a link to control the position of the wheel as well as the load bearing member, thus replacing the upper control arm making it compact. However, this design does not offer very good handling as body roll and wheel's movement lead to variation in camber (degree to which the wheel tilts in and out), shown in Figure 3.4.4, usually ending with positive camber.

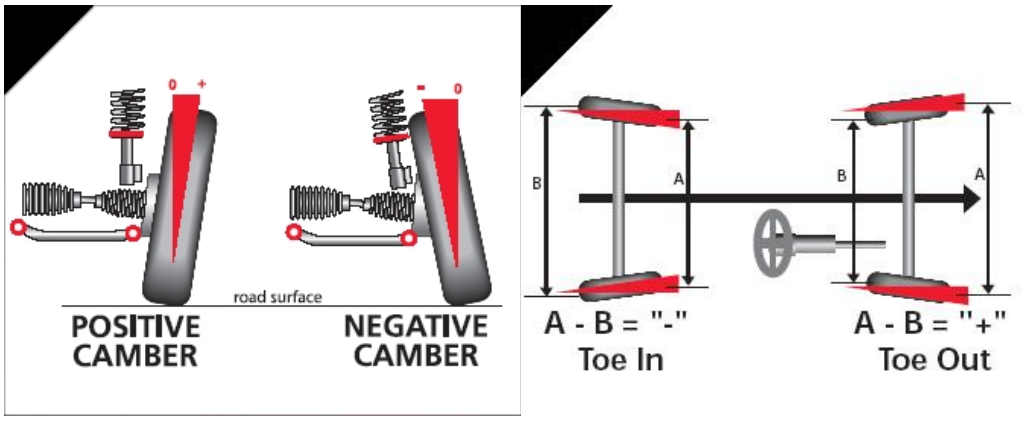


Figure 3.4.4 – Camber Angle and Toe Angle (Barrys Tyre & Exhaust Centre, 2010)

Consequently, the control arm will experience expansion rather than the ideal state of compression. This gives engineers less freedom to adjust the camber angle and roll center. Its high overall height requires a higher hood line, which is not desirable in the design of the solar car body as it will increase drag and decrease its streamline body design.

A double wishbone suspension design, shown in Figure 3.4.5, is regarded by many designers as the most ideal suspension. It includes two (2) links forming a wishbone shape where one end is fixed to the frame of the car and the other end to the lower and upper ball joints supporting the upright arm that holds the wheel. A coil spring and damper combination is fitted between the two wishbones. Its parallelogram design allows the wheels to travel vertically up and down and a slight side-to-side motion know as scrub. There are two other wheel movements relative to the body produced by this suspension: toe angle (Figure 3.4.4) or steer angle (difference in the distance between the front of the tires and the back of the tires), and camber angle or lean angle. This results in a complex system, but it provides engineers the freedom to adjust the kinematics minimizing roll or sway resulting in a more consistent steering feel.

Moreover, this design always maintains the wheel perpendicular to the road surface, irrespective of the wheel's movement ensuring good handling.

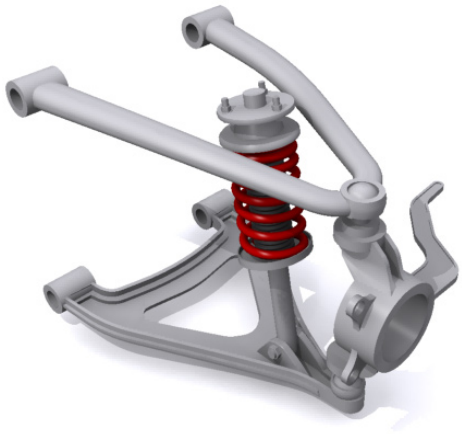


Figure 3.4.5 – Double Wishbone Suspension (Longhurst, 2010)

Table 3.4.1 shows a comparison table between the two (2) suspension designs.

Table 3.4.1 – MacPherson Strut vs. Double Wishbone Suspension

MacPherson Strut		Double Wishbone	
Advantages	**Disadvantages**	**Advantages**	**Disadvantages**
Compact	Average handling	Ideal camber control	Complex
Cheap	High overall height	Good handling	Space engaging
Simple	Camber angle change	Easily tuned kinematics	Costly
	Expensive replacement	Optimized lightweight parts	

After comparing the two (2) suspension designs, a double wishbone design was chosen as the best fit to the front suspension of the solar car. The double wishbone design gives the freedom to adjust camber and toe angles, as well as scrub radius, and allows a vertical wheel movement perfect for the constrained airfoil shaped wheel enclosure.

The control and upright arms were manufactured in the college's machine shop out of aluminum allowing for optimized lightweight parts, another advantage in achieving a light weight car. The designed control arms have a clearance of twelve (12) inches between each point of contact with the car's frame rib, as shown in Figure 3.4.6.

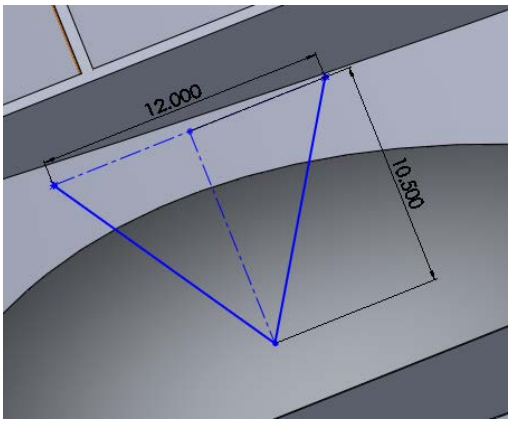


Figure 3.4.6 – Front Suspension Linear Sketch in Inches

However, the perpendicular length from the midpoint between both points of contacts to the ball joint linking the control arm to the upright, shown by the dashed line in Figure 3.4.6 – Front Suspension Linear Sketch in Inches, differs for both lower and upper control by 0.5 inches. The upper control arms have a length of ten (10) inches whereas the lower control arms have a length of 10.5 inches. The short/long control arm design was created to keep the position of the contact patch of the wheel in a straight vertical line under bump and rebound conditions. The lower and upper control arms design is shown in Figure 3.4.7.

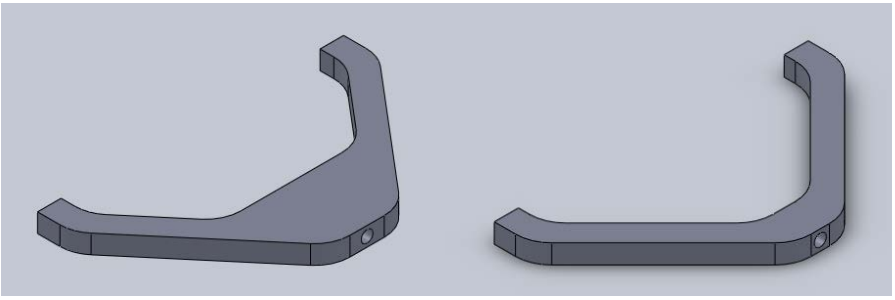


Figure 3.4.7 – Lower and Upper Control Arms

The control arms will be connected to the ribs and upright using heim joints (Figure 3.4.8). A heim joint is an extremely rigid mechanical articulating joint containing a ball swivel with an opening, through which a bolt may pass, pressed into a circular casing with a threaded shaft attached. The threaded portion may be either male or female.

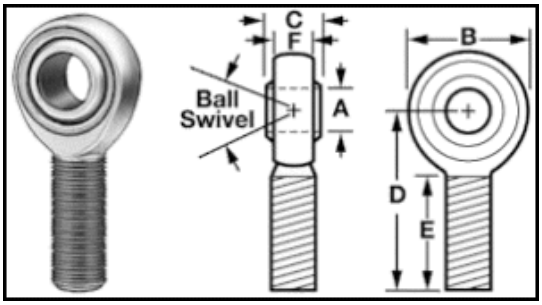


Figure 3.4.8 – Heim Joint

Figure 3.4.9 shows the assembled lower and upper control arms with the heim joint attached at each end.

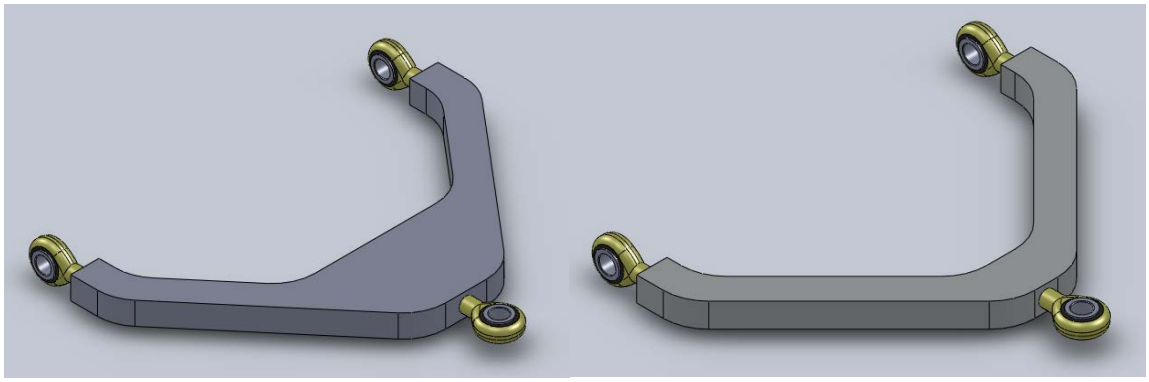


Figure 3.4.9 – Lower and Upper Control Arms with Heim Joints

There are many different types of spring/damper systems for a vehicle's suspension such as leaf springs, air bag suspension, gas shocks, and coil over spring/damper. The coil over spring and damper system is the best fit for the solar car suspension design. It is easy to integrate to the double wishbone suspension, durable, and easy to adjust and maintain. Assuming a total car weight of 600 lbs, a desired displacement of two (2) inches, using Hooke's Law (Equation 3.5.1), and moment and force static analysis, the spring force and shock location were calculated. The calculated spring needed for each left and right shock was 462 lbs. The 2005 FOX Racing Shox Vanilla R (Figure 3.4.10) was chosen as the shock for the front suspension due to its superior quality and the capacity to hold a 500 lb spring. Springs are not sold for exact measurements, thus a 500 lb spring was chosen for safety purposes for each left and right front shock. Moreover, this model has compression and rebound adjustments, as well as spring preload.

Figure 3.4.10 – Fox Racing Shox Vanilla R 05

Figure 3.4.11 shows a sketch of the front suspension depicting the designed and calculated hard point locations (control arms and shock locations).

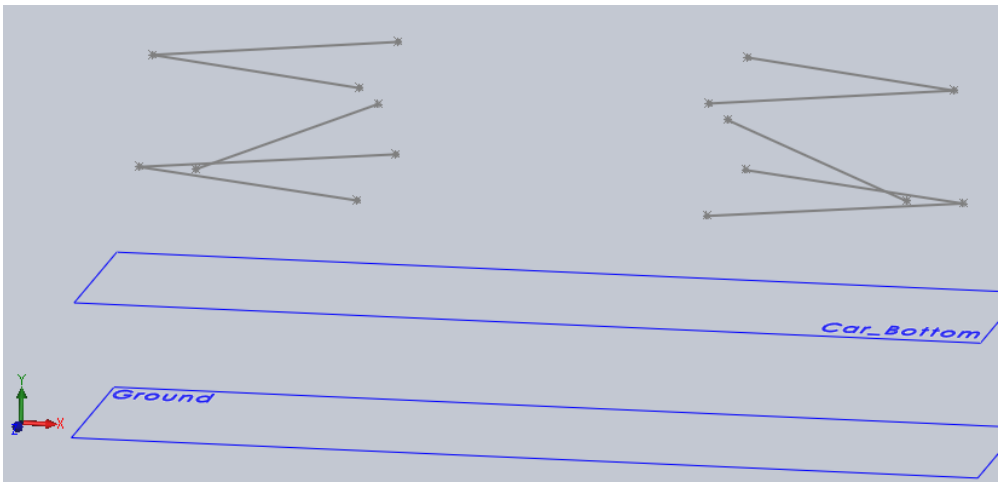


Figure 3.4.11 – Front Suspension Sketch

An upright (Figure 3.4.12), also known as kingpin, connects the single outer ends of the control arms to the wheel of the car serving as the main pivot in the steering mechanism of the vehicle. In the case of our design, the upright is bolted to the spindle and caliper mounts, and holds the tie rods on the stem. The upright was manufactured in the machine shop out of aluminum.

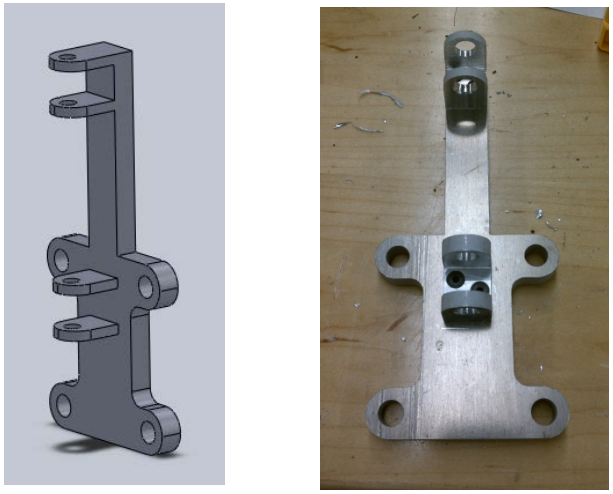


Figure 3.4.12 – Upright

Figure 3.4.13 shows the assembled lower and upper control arms with the upright and spindle without the shock.

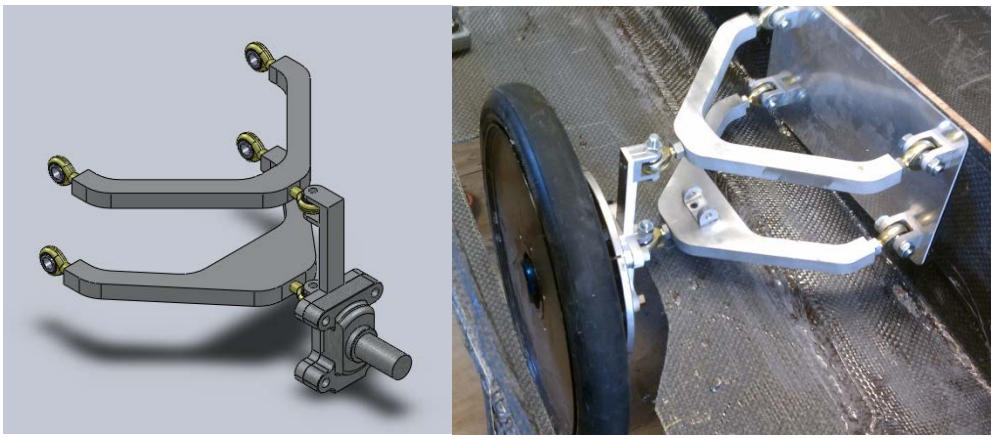


Figure 3.4.13 – Assembled Front Suspension with Spindle

As seen on the above right figure, the heim joints on the control arm are linked to the car using aluminum brackets bolted to an aluminum plate on the rib wall of the body. The heim joints are secured to the brackets using bolts and locknuts to prevent loosening under torque and vibrations. Moreover, to increase rigidity of the wall and distribute the compression force of the springs, aluminum square tubing was placed in the center between the two rib walls where the front suspension is located. However, two more bars of square tubing were placed in between the walls of the left and right suspension as mounts for the rack and pinion, and pedal cluster, but also keep the walls rigid and absorb the springs' compression. These square tubings, marked by the oval in Figure 3.4.14, were welded at each end to a small aluminum plate then bolted to the plate and wall of the suspension.

Figure 3.4.14 – Compression Support Square Tubing

Figure 3.4.15 shows the completed front suspension.

Figure 3.4.15 – Front Suspension

Simulations in MSC ADAMS/Car, such as parallel wheel travel, were performed to observe the behavior and characteristics of the designed suspension. The results to this simulation are in the front suspension test section.

3.4.2 Rear Suspension

The rear suspension system supports the single rear wheel as well as the motor connected to it. Similar to last year's Phase I of the solar car project, a single trailing arm suspension design was used as it proved to be the best fit for the application as well as perform the desired movement and operation. However, calculations and analysis on this design were done using the constraints of the new body design, resulting in modifications to the design.

A trailing arm, or swing arm, suspension (Figure 3.4.16), is similar to that of a motorcycle. It has an arm joined at the front to the chassis that allows the rear to swing up and down, a suited motion for the single rear wheel. This prevents side-to-side scrubbing allowing only vertical motion, thus no change in the camber angle.

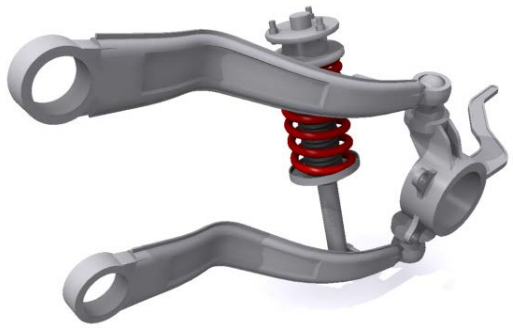


Figure 3.4.16 – Trailing Arm Suspension (Longhurst, 2010)

Last year's trailing arm design had the swing arm holding the wheel on one side as shown in Figure 3.4.17. This created a torque on the wheel making it bend and not be perpendicular to the road surface. To prevent torque and moment from developing, the control arm will hold the wheel and motor on both sides.

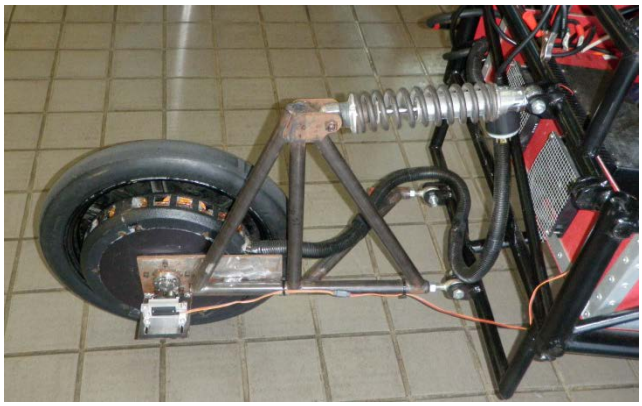


Figure 3.4.17 – 2009-2010 Solar Car Trailing Arm Suspension

The new control arm was designed using square aluminum tubing to make it light weight and increase rigidity by preventing the weight of the motor from causing it to bend. The control arm consists of two extensions of tubing welded to each other by two transverse and one diagonal sections of tubing. Then, two vertical mounts on each extension were welded to hold the motor and wheel in place.

Similar to the front suspension, a coil over spring and damper system was chosen. The Koni 8212-1408 damper and coil from last year's Phase I solar car was selected for this application (Figure 3.4.18). It has an aluminum body and a twin tube hydraulic construction with adjustable rebounding and compression damping with valvings rated for 350-650 lbs/in springs, and a coil with a rating of 600 lb/in. The shock is connected to the lower arm on the transverse closest to the wheel and to the back wall of body on the other end (depicted by black ovals in Figure 3.4.19).

Figure 3.4.18 – Koni Damper and Coil

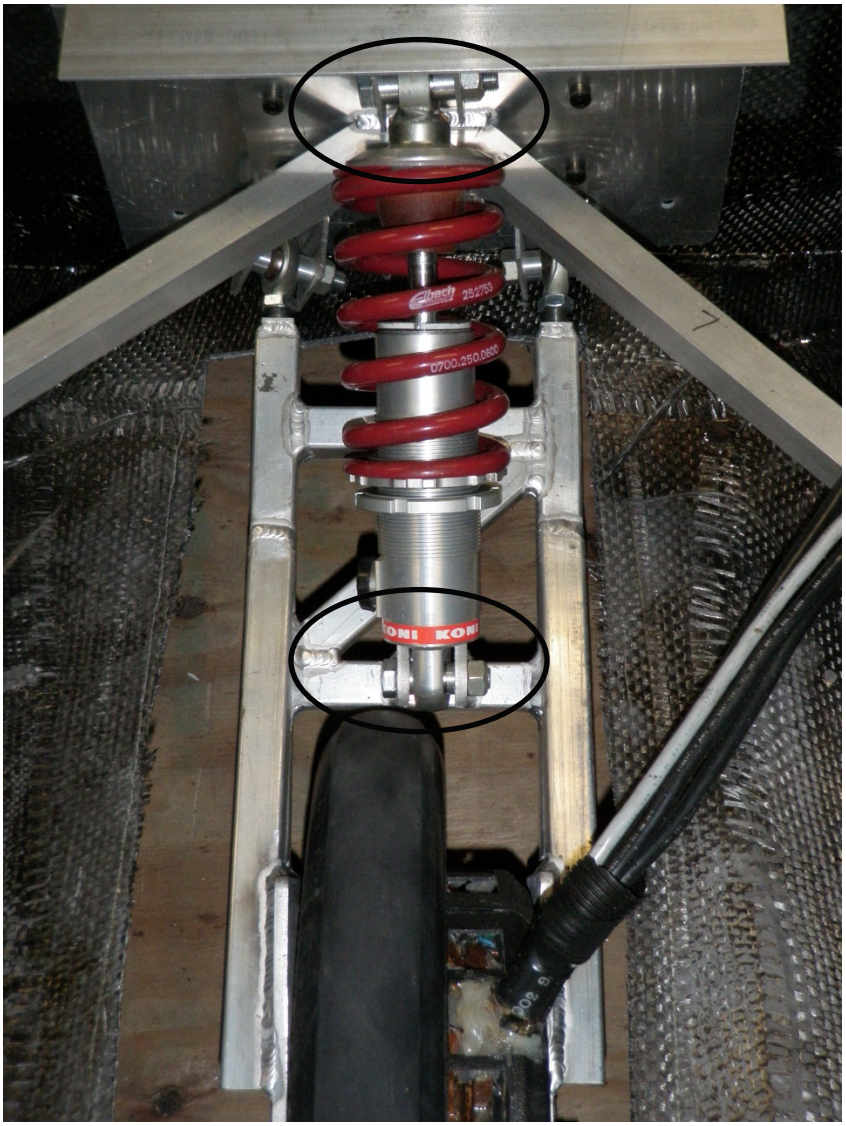


Figure 3.4.19 – Rear Shock Joints

The heim joints are attached at each end of the control arm linking it to the car using aluminum brackets bolted to an aluminum plate on the back wall of the body. These two heim joints are secured to the brackets using bolts and locknuts. Since a great amount of forced is absorbed by the shock and then applied to the wall, two square tubing members were welded to the aluminum plate at the center and bolted in a diagonal to the side rib walls on both front and back of the wall supporting the rear suspension. Also, an L-shaped aluminum bar was placed in between the spring's wall bracket and aluminum plate to absorb the spring's compressive force. The rear suspension is shown in Figure 3.4.20.

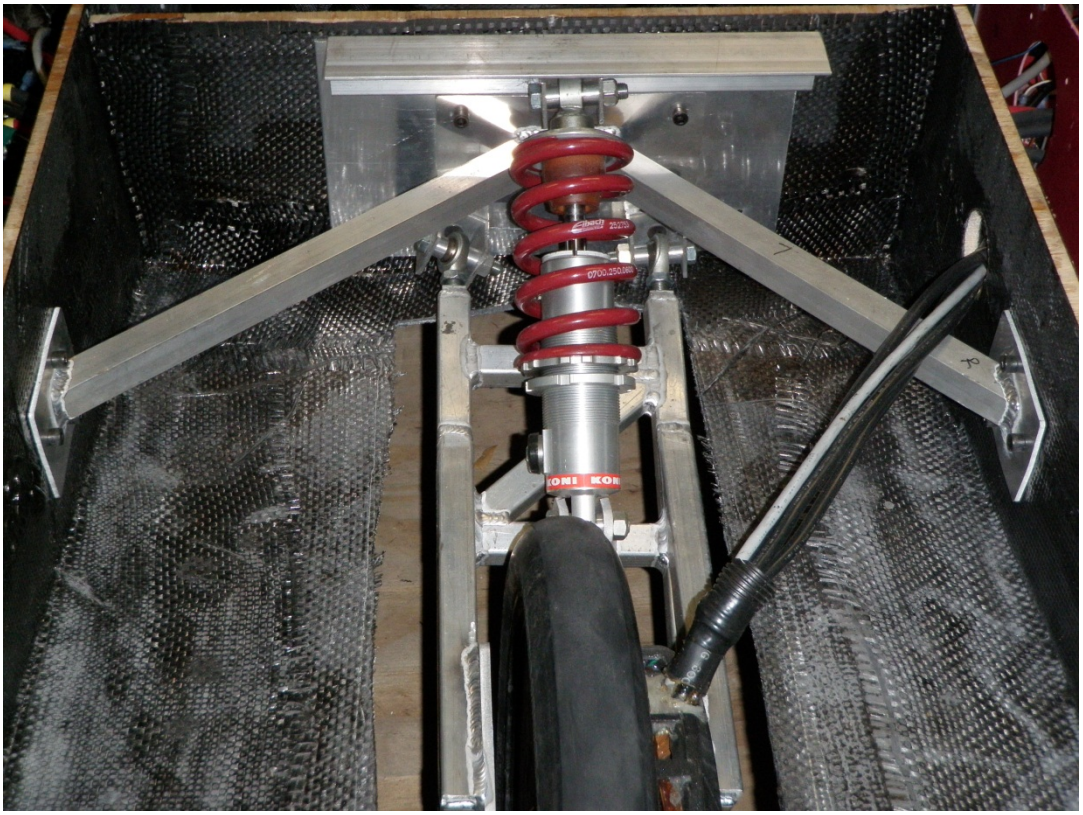


Figure 3.4.20 – Rear Suspension

A minor risk associated with the design of the suspension was the budget. The 2011 FOX Racing Shox Vanilla R cost $210, thus for two shocks, the total was $420, decreasing our available budget. However, after further research, the 2005 models were found at a much lower cost of $58 each.

3.5 Power Generation

The power generation system will be composed of solar array system, regenerative braking system, and a maximum peak power tracker (MPPT). The solar array system will channel energy from solar radiation into electrical energy. This energy will either propel the vehicle, or charge the vehicle's battery system. The MPPT will optimize performance of solar array system to provide maximum amperage to either charge the batteries, or propel the vehicle. The regenerative braking system will charge the battery system through the motor controller, when asserted by the driver. The regenerative braking system and mechanical frictional braking system will provide total braking output. Figure 3.5.1 below displays the overview of the power generation system.

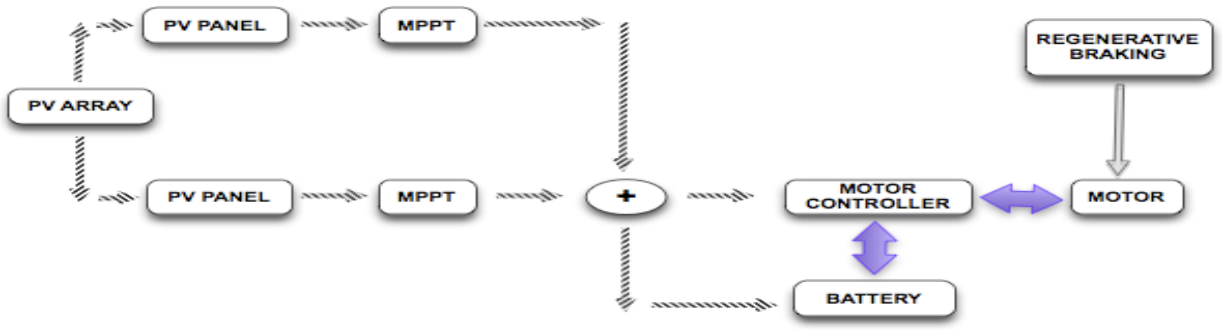


Figure 3.5.1 – Overview of Power Generation System

3.5.1 Solar array system

The solar array system is an important component in the solar car. It is responsible for conversion of electromagnetic radiation energy of the sun into electrical energy. It is an array of solar cells configured to provide an output power suitable to propel the vehicle, or charge the battery.

The solar array is designed to input solar radiation energy and output electrical power. The fundamental unit of this system is a solar cell. A solar array is parallel and/or series configuration of solar cells. **Error! Reference source not found.** below displays the functional block diagram of the solar array system.

Figure 3.5.2 – Top level diagram of the solar array system

The only design decision that truly needs to be made in reference to the solar arrays is the type of cells to use. There are many commercially available cells available, but obviously only one will be utilized for purposes of the project. Currently on the market are amorphous single junction, amorphous multi-junction, monocrystalline, and polycrystalline.

The first two amorphous cells are also referred to as thin film cells but are drastically different in quality. Generally single junction thin film cells use primarily Silicon as a base component and will yield efficiencies between 8% and 10%. The single junction cells will create a single layer for solar radiation collection, meaning that a significant portion of the solar radiation will pass through. To correct this problem a multi-junction cell can be used. These cells will typically use a material such as Gallium to create multiple layers for collection. Since multi-junction cells have been drastically improving over the years, the range of efficiencies can be as low as 16% and in excess of 40% (Renewable Energy Access, 2006).

The other solar cells are crystalline cells. As can be divined from the name, these cells are produced in crystalline structures and are most comparable to pieces of glass. While the science between how these cells collect energy is similar, the creation of these cells utilizes a different process, resulting in the different product. Contrary to the way that the thin film cells operate however, the monocrystalline cells are more efficient than their polycrystalline counterparts. Monocrystalline cells are a single crystal wafer which ideally will be perpendicular to solar radiance and therefore collect a peak amount of photons. Polycrystalline are almost an unrefined version of monocrystalline in the fact that it is not cut to a single perpendicular layer, reducing the uniformity of photon collection. This reduction in uniformity results in overall lower efficiencies. When the first crystalline solar cells were created their efficiencies were less than 1%, but in today's market average capacity is between 14% and 20%.

In order to determine which solar cells should be used for the solar car application, a comparison needed to be made between the crystalline and amorphous. The Gallium cells were immediately removed from the decision process due to the cost being outlandish compared to the overall budget of the project. The next step was to find available solar cells for the purchasing process. For the amorphous silicon cells PowerFilm PT15-300 was chosen and a SunPower SPR-320 was used to compare crystalline panels.

Table 3.5.1 – Quick comparison between PowerFilm and SunPower Cells

Panel	PowerFilm	SunPower
Efficiency	9.07 % (from testing)	19.6 % (from datasheet)
Total Weight $6m^2$	17 lbs	~150 lbs
Temperature Coefficiency of P_{mp}	-0.2 %/°C	-0.38 %/°C

Average sunlight radiation in Florida is around 800 W/m^2 and will be used for further calculations

$$Total\ power\ in\ 6m^2 = 6m^2 \times 800\ {}^{W}/_{m^2} = 4800W$$

$$PowerFilm = 4800W \times 9.07\% = 435W$$

$$SunPower = 4800W \times 19.6\% = 940.8W$$

These values are calculated at 25 °C. As can be seen from the table above, cells lose efficiency as they heat up. According to solar-facts.com most solar cells while actively collecting radiation operate at around 50 °C.

$$PowerFilm = 25℃ \times 0.2\% = 5\% \rightarrow 435W - 435W \times 5\% = 413.25W$$

$$SunPower = 25℃ \times 0.38\% = 9.5\% \rightarrow 940.8W - 940.8 \times 9.5\% = 851.42W$$

In order to estimate the lifetime of the stored battery power the Tesla Roadster will be used as a sample. The roadster probably has a much higher efficiency in its design compared to that of our solar car, but has a higher air drag, powerful cooling system for the batteries, and other power uses such as radio and A/C which will bring closer balance between the two vehicles. Figure 3.5.3 shows the power usage based on the roadster's velocity. At 50 mph there is about 200 Wh/mi used to drive the 2723 lb roadster. The estimated weight of our car is 600 lbs without solar panels and contains 3840 Wh of stored energy.

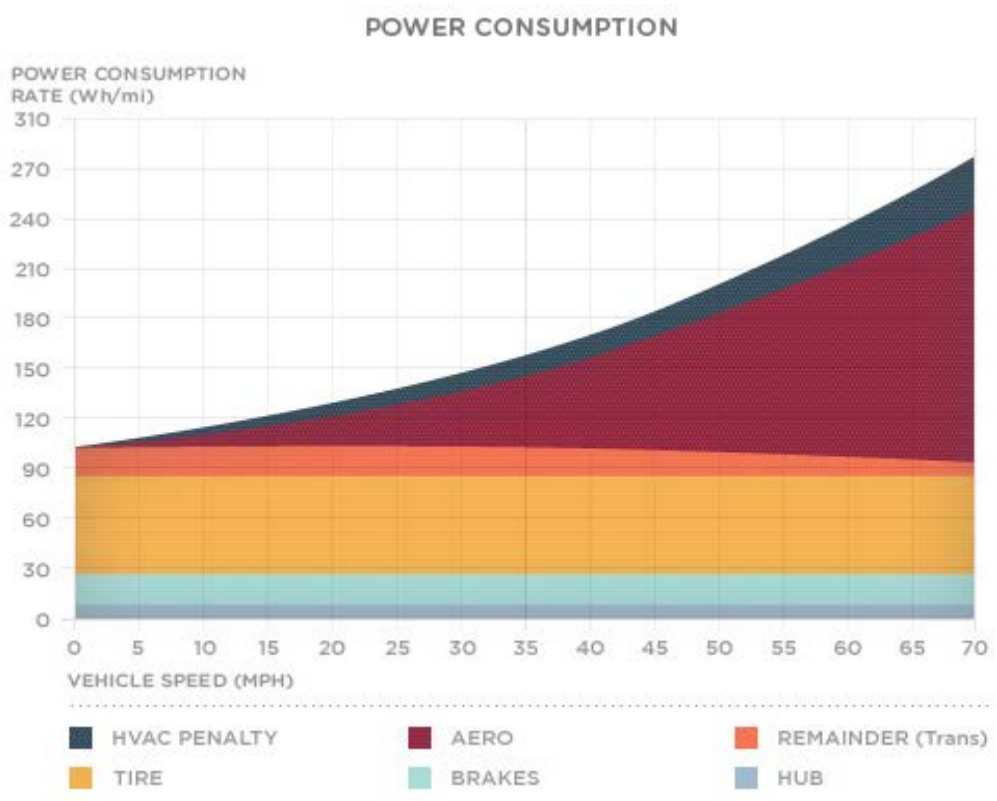


Figure 3.5.3 – Power consumption of Tesla Roadster (Tesla Motors)

$$PowerFilm\ power\ use = \left(\frac{617\ lbs}{2723\ lbs}\right) \times 200\,{}^{Wh}/_{mi} = 45.31\,{}^{Wh}/_{mi}$$

$$PowerFilm\ time\ without\ solar = \frac{3840\ Wh}{45.31\,{}^{Wh}/_{mi}} \div 50\frac{mi}{h} = 1.695\ hours$$

$$PowerFilm\ power\ use = \frac{3840\ Wh}{1.695h} = 2265W$$

$$SunPower\ power\ use = \left(\frac{750\ lbs}{2723\ lbs}\right) \times 200\,{}^{Wh}/_{mi} = 55.09\,{}^{Wh}/_{mi}$$

$$SunPower\ time\ without\ solar = \frac{3840\ Wh}{55.09\,{}^{Wh}/_{mi}} \div 50\frac{mi}{h} = 1.394\ hours$$

$$Sunpower\ power\ use = \frac{3840\ Wh}{1.394h} = 2755W$$

These are the power uses of the motor at 50 mph based on the weight of the car. The rest of the power used by the system is neglected because it is significantly smaller when compared to the motor power use. Now to determine the time the car will last when utilizing the solar power.

$$PowerFilm\ power\ use = 2265W - 413.25W = 1851.75W$$

$$PowerFilm\ time = \frac{3840Wh}{1851.75W} = 2.07\ hours$$

$$SunPower\ power\ use = 2755W - 851.42W = 1903.58W$$

$$SunPower\ time = \frac{3840Wh}{1903.58W} = 2.02\ hours$$

The PowerFilm solar panels will allow the car to last longer at 50 mph if the assumption can be made that power use is directly proportional to weight of the vehicle. If a similar calculation was performed at 30 mph where power use for the roadster is 145 Wh/mi, the PowerFilm will last 6.71 hours and the SunPower will last 11.08 hours. There is a huge discrepancy in how long the batteries will last with a variation is speed.

While the calculations above may not be very accurate in terms of the system's overall power use it does highlight the possible difference between the two solar panels. IESES has granted the solar car team with an additional $4000, which will be spent on solar cells and bubble for the driver. A choice needs to be made about which solar cell will need to be purchased.

The reasons to choose PowerFilm:

- Lightweight will help improve efficiency at higher speed
- Flexible and can be mounted flat on the body to keep air drag lower that mounting with crystalline
- 10 panels have already been purchased and preliminary designs for a charging device (MPPT) are utilizing these cell parameters
- Small to no lead time, these cells could be ordered and shipped within a week, unlike the SunPower cells which may not be available for a month

The reasons to choose SunPower:

- Higher efficiency overall which will allow the car to run for many hours at lower speeds
- Reduced charge time when car is parked outside and not being used

The team as a unit decided to utilize the amorphous silicon solar cells primarily for the flexibility. There would be a great deal of assumed risk involved in attempting to utilize the crystalline cells when none of the team members have any experience with these types of cells. The risk of damaging cells during installation, on a project which was already under budget was too big a risk to take.

3.5.2 Maximum Peak Power Tracker

Solar cells have a non-linear I-V relationship; this relationship varies widely with respect to solar irradiance level. This causes fluctuations in output power. The MPPT is basically a DC: DC converter. It has an efficiency of 92-97%. Its main mode of operation is optimization of power output from the solar panels to provide maximum amperage to the system. MPPT's provides protection to the battery and solar array. There is some loss of power due to efficiency of the component (MPPT). MPPT's are known to have efficiencies from 92-97%. Our design team had looked at two MPPTS:

Drivetek AG MPPT-Race V 4.0 from a company in Germany and AERL RACEMAX 600B from Australia. Figure 3.5.4 depicts the block diagram of an MPPT along with a picture of Drivetek AG MPPT; then follows with

Figure.3.5.5 comparing the previously mentioned two different types of MPPT suitable for high voltage solar car application.

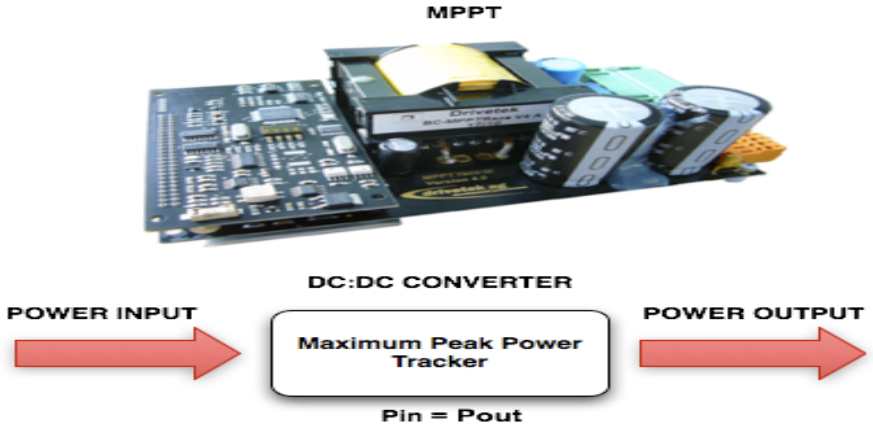


Figure 3.5.4—MPPT Block Diagram

DRIVETEK RACE

Parameter	Unit	Minimum	Typical	Maximum
Input Power Continous	W	5		800
Input Power Peak[3]	W			1250
Input Current	ADC			9
Peak Efficiency[2]	%		99	
Input Voltage Range	VDC	36		144
Output voltage Range[4]	VDC	40		200
Output Shutdown Voltage[7]	VDC			236
Input to Output Voltage Ratio[6]	-	1.05		4

AERL 600 B

Parameter	Max
Maximum ambient air temperature	50°C
PV panel short circuit current - constant	6A
PV panel short circuit current – transient	8A

Parameter	Min	Max
Solar panel peak power	0W	600W
PV panel open circuit voltage	40V	135V
Efficiency @ 6A, 100Vmp, & 25Camb	98.00%	-
Battery Voltage (Selectable)	72, 96, 120, 144, 168V	

Figure.3.5.5 – Comparison of two different MPPT

Figure.3.5.5 above shows Drivetek RACE V 4.0 MPPT provides a wider range of power handling capability, greater input current, and wider input/output voltage range than AERL 600B MPPT. AERL 600B MPPT is also limited to a battery selectable voltage level of 72, 96,120,144,168V. It should be noted that AERL 600B has a lower cost than Drivetek RACE V 4.0. Commercial market does offer basic charge controllers and PWM charge controllers (Pulse Width Modulated). However, they are not able to track maximum power point of solar panels, or offer protection to the battery and solar array system. They are only able to charge the batteries until they are “full”, then the charge controller disconnects the battery from the solar array. On the other hand, MPPT operates the PV array at a voltage which can deliver maximum output power at the prevailing solar irradiance.

The MPPT is a component that connects the solar panels with the battery system of the solar car. Figure 3.5.6 below depicts a component diagram of this system. It is followed by Figure 3.5.7 which displays a top level shunt type interconnection schematic for the charge controller and the MPPT.

Figure 3.5.6 – Component diagram of MPPT with solar panel and battery system

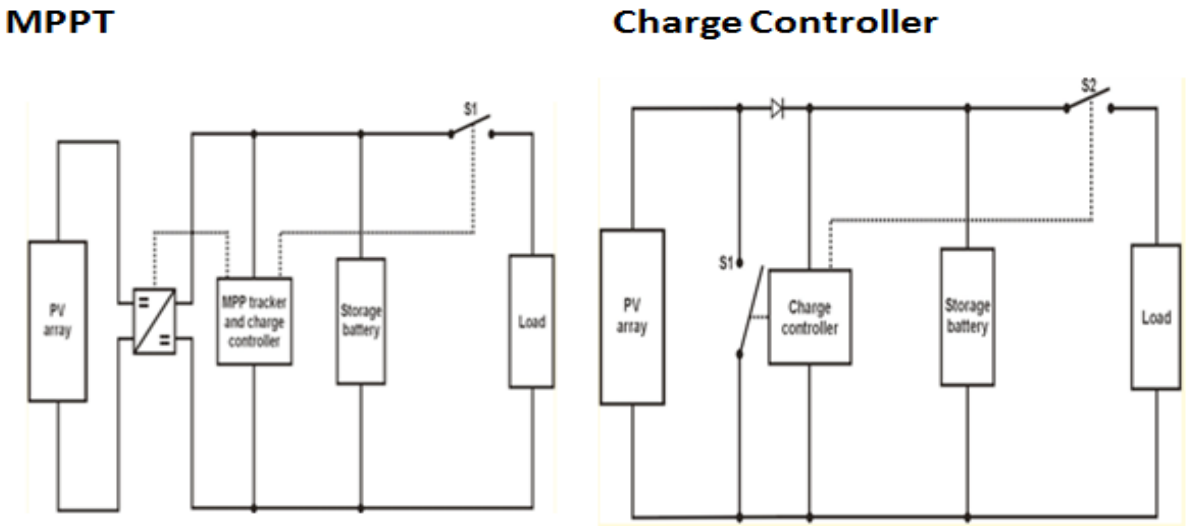


Figure 3.5.7 – MPPT and charge controller top level component integration

Due to budgetary constraints, either the DRIVETEK MPPT or the AERL 600B MPPT could not be purchased in phase 2. As such, design and simulation phase of an MPPT suitable for the solar car was initiated by the team.

3.5.2.1 Design and Simulation of MPPT

Figure 3.5.8 below displays a basic schematic of a boost convertor. It is followed by equations to solve for the parameters of the passive elements used in the convertor and the duty cycle for the pulse width modulation of the transistor. Table 3.5.2 below the equation contains values of the parameters used for the modeling of the boost converter given input solar panel voltage is 80 V and the desired output voltage for battery charging is 126 V.

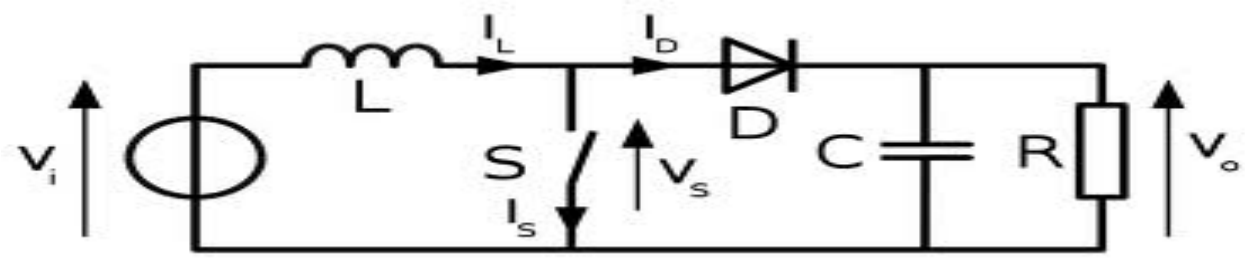


Figure 3.5.8 – Basic Boost Convertor Circuit

Equations to find parameters for the boost convertor:

- Vout = Vin / (1 – D)
- L = (Vout – Vin) * (Vin / Vout) * (1/ (Frequency * ΔI))
- C > = (Vout * D) / (Frequency * ΔVout * R)
- At 100V, the motor was drawing 5 A in steady state drive, so we modeled the resistor as R = 20Ω

Table 3.5.2 – Circuit Parameters for Boost Convertor

Parameters	Boost Converter
Solar Array Voltage (V_PV)	80 V
Desired Output Voltage (V_Link)	126 V
Duty Cycle (%)	36.5 %
Frequency (kHz)	40 kHz
Inductor (mH)	7.3 mH
Capacitor (uF)	221 uF
ΔV (V), ΔI (A)	ΔV = 0.5 V, ΔI = 0.1 A
IGBT w/ antiparallel Diode	Bidirectional, 2 Quadrant SPST switch

3.5.2.1.1 Case I: DC/DC Boost Converter

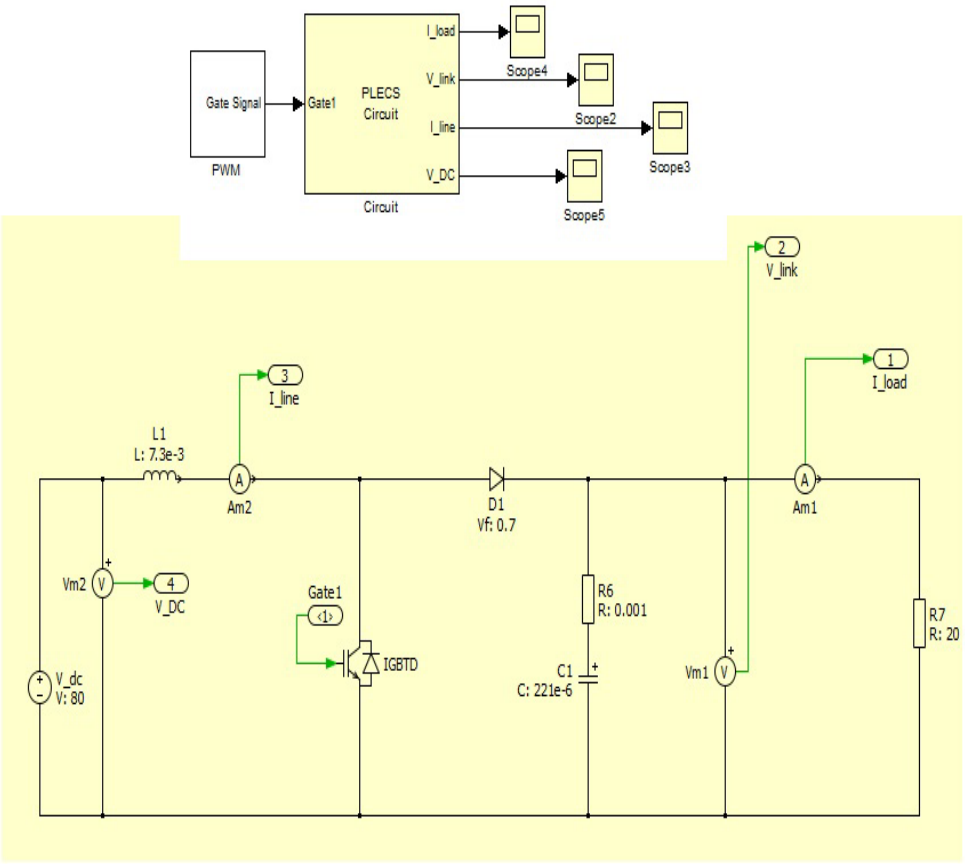


Figure 3.5.9 – DC/DC Boost Converter

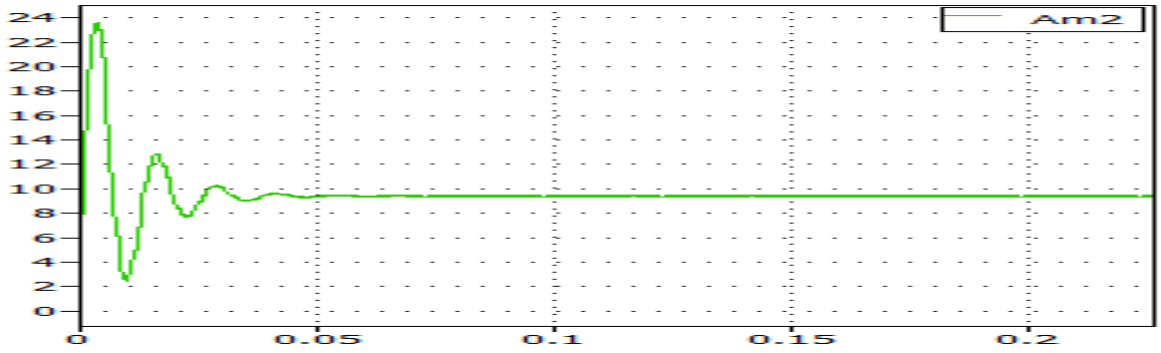


Figure 3.5.10— I_Line from Figure 3.5.9 (A) vs. Time

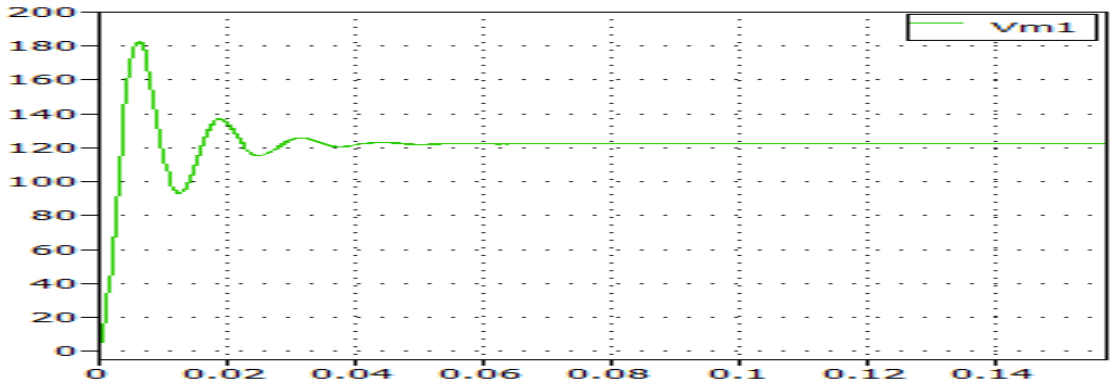


Figure 3.5.11 – V_Link from Figure 3.5.9 (V) vs. Time

Figure 3.5.12 – I_Load from Figure 3.5.9 (A) vs. Time

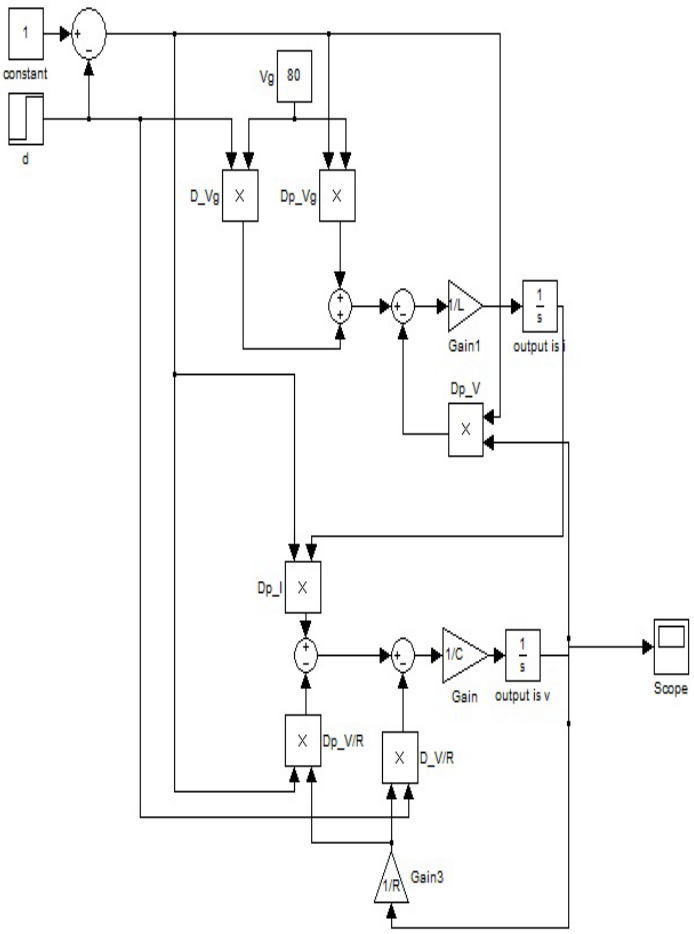


Figure 3.5.13 – Average Model of a Boost Converter

In Case I, we designed and simulated a DC/DC Boost converter. The source is a DC voltage source. This case was basically a test to confirm if the passive component element parameters and the duty cycle ratio selected yielded the desired output. The switching frequency model is followed by an average circuit model. The average circuit model was achieved from the help of Jesse Leonard from CAPS. In an average circuit model, the ripples present during switching are not present because the model outputs an average value.

3.5.2.1.2 Case II: Solar Array Model

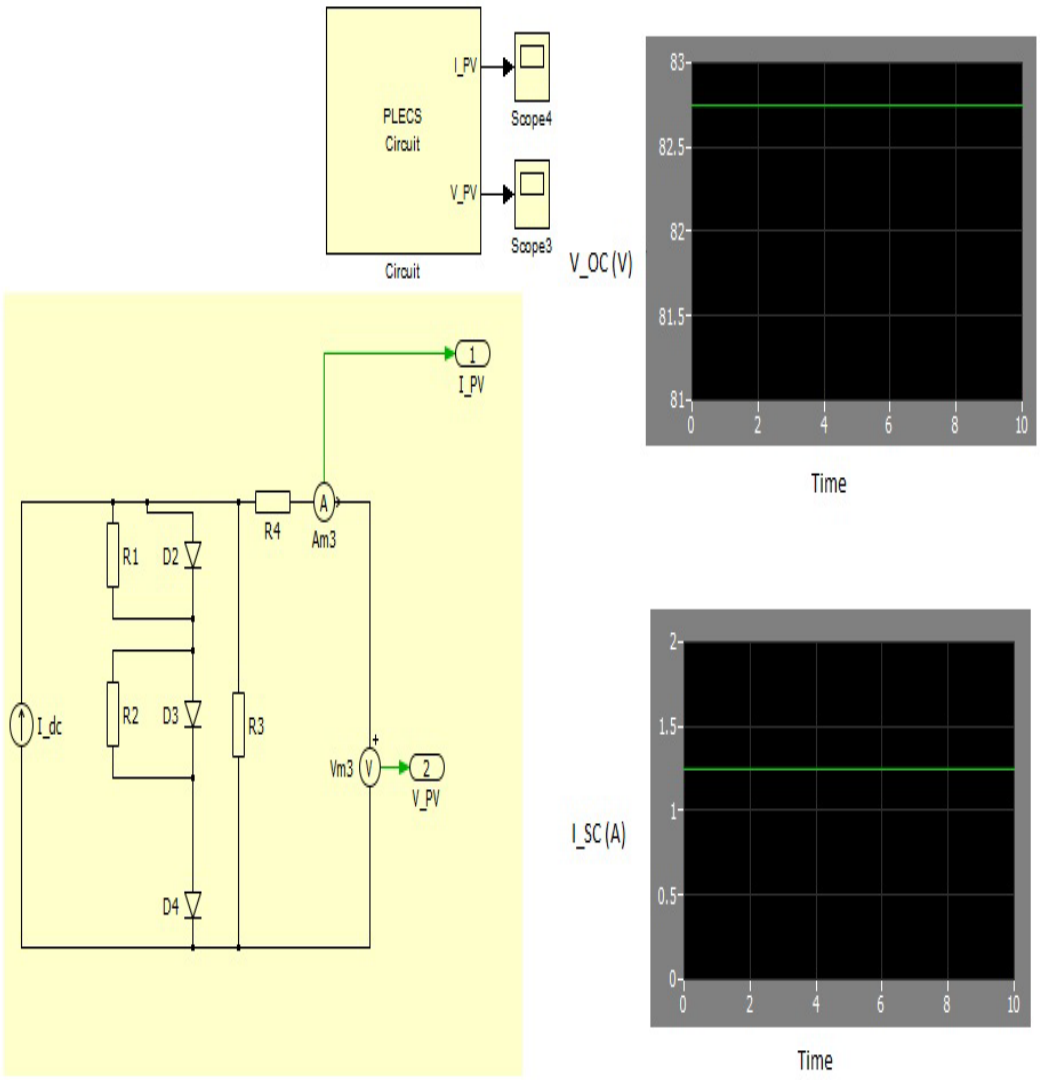


Figure 3.5.14 – Solar Array Model Simulation

Case two represents design and simulation of a solar cell model. It was attained with the help of Dr Saritha and Dionne from CAPS. The forward diode voltage is based on a ratio of the present diode forward bias voltage and 120 V, the ratio of this is multiplied by a factor of desired panel output voltage. The resistances need to be scaled up and down by some factor (trial and error, initiate with 10). The simulation result in Figure 3.5.14 displays the model gives us an output voltage of approximately 80 V, and a short circuit current of 1.25 A; this model fits suitably to the solar cell parameters of the solar cell that the team currently has ordered. This model will be used as an input source to the boost convertor to the boost convertor.

3.5.2.1.3 Case III: Solar Array Model – Boost Converter – NO Load

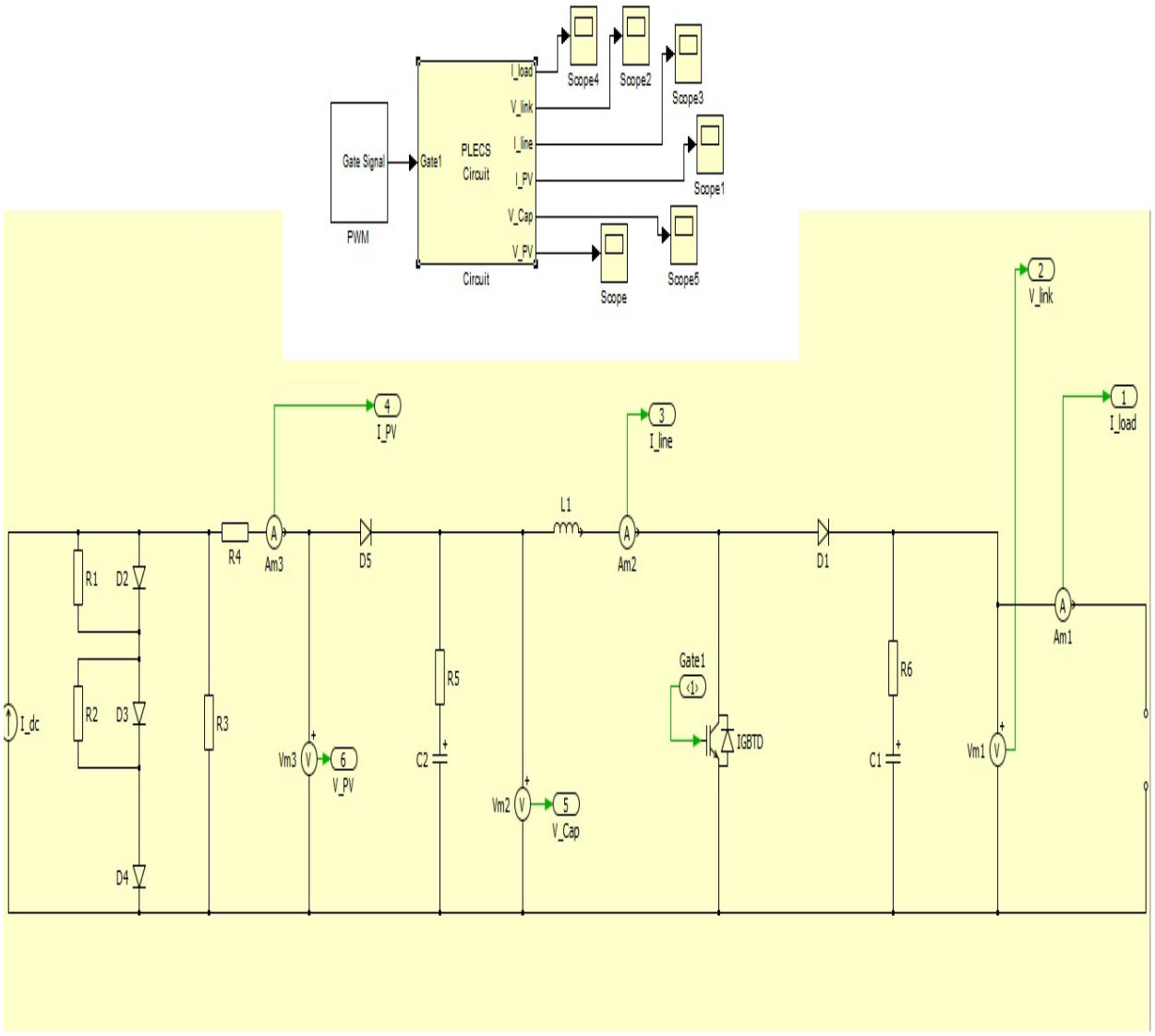


Figure 3.5.15 – Solar Array Boost Converter with no load

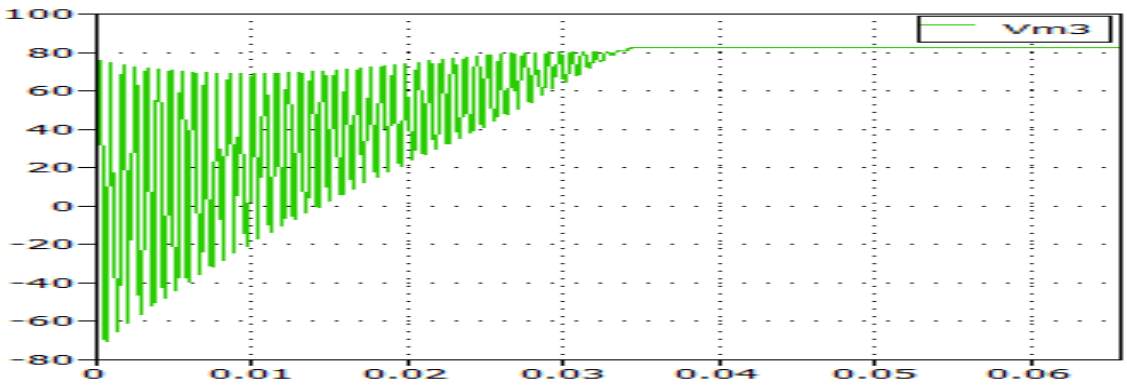


Figure 3.5.16 – V_PV from Figure 3.5.15 (V) vs. Time

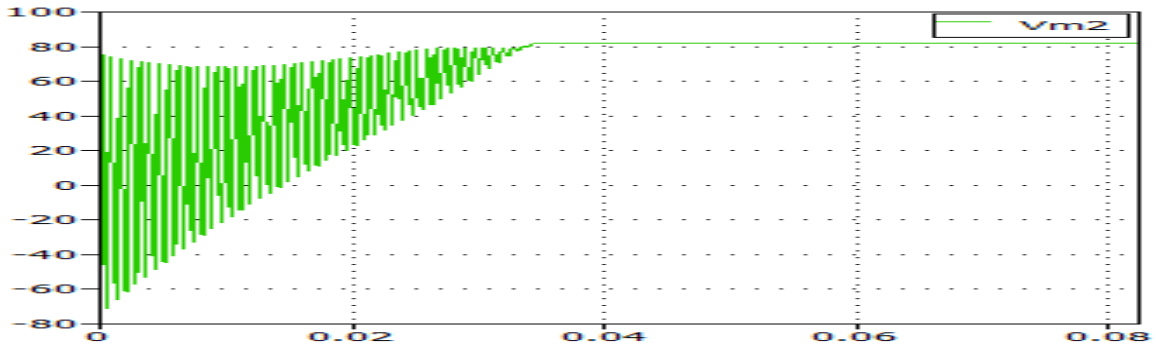


Figure 3.5.17 – V_Cap from Figure 3.5.15 (V) vs. Time

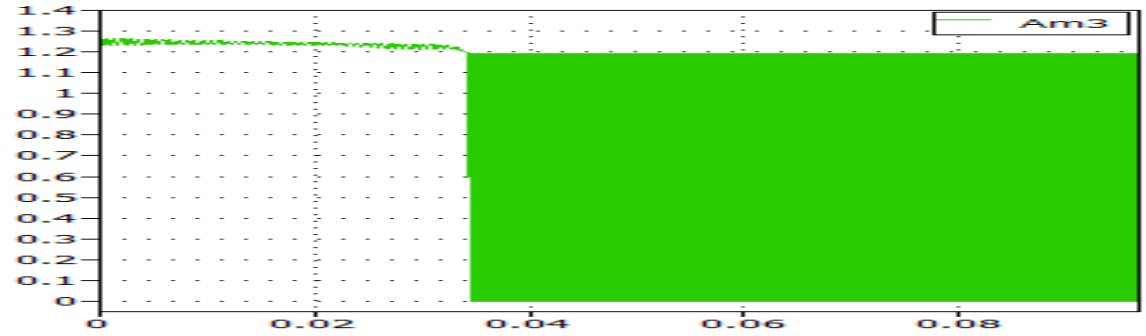


Figure 3.5.18 – I_PV from Figure 3.5.15 (A) vs. Time

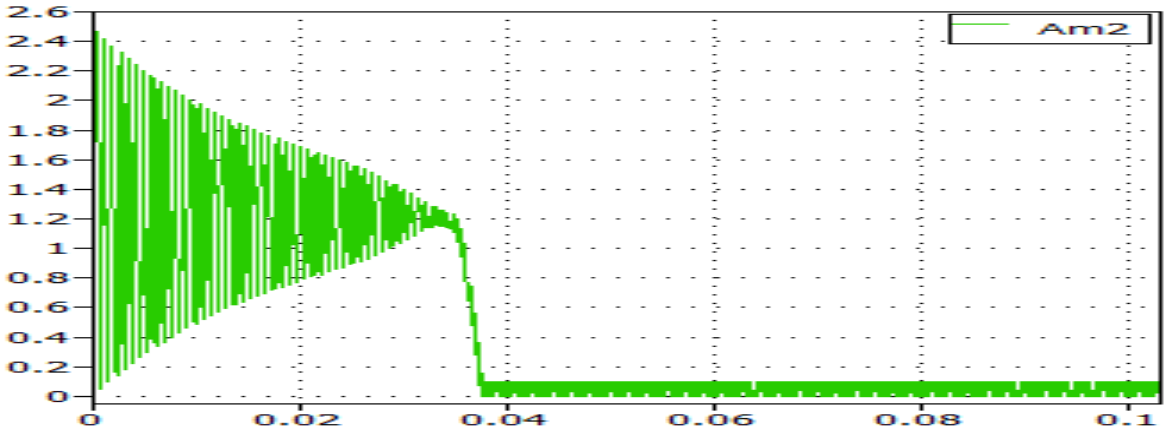


Figure 3.5.19 – I_Line from Figure 3.5.15 (A) vs. Time

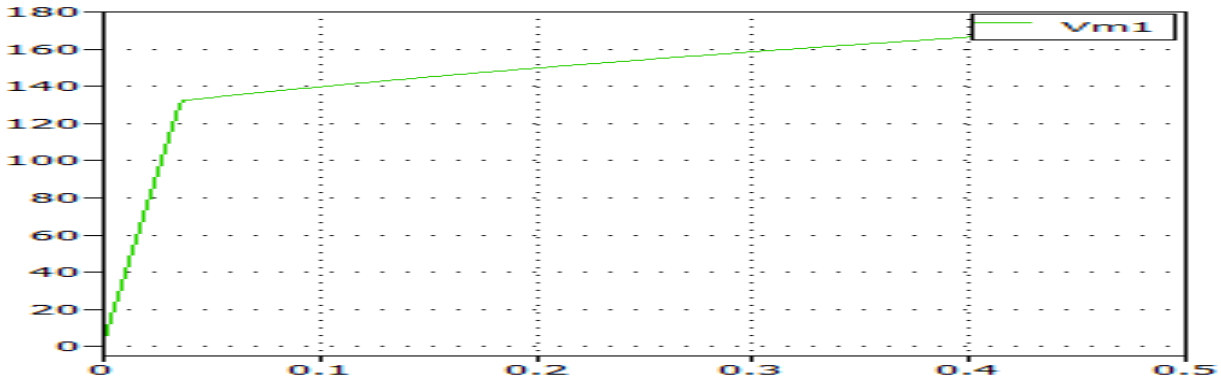


Figure 3.5.20 – V_Link from Figure 3.5.15 (V) vs. Time

The solar array and the boost convertor is simulated under no load condition. We note that DC link voltage increases gradually with time. The figures above display simulation results for other parameters. This result shows that we do not want the convertor connected to the solar array under no load condition.

3.5.2.1.4 Case IV: Solar Array Model – Boost Converter – Battery Load (modeled as Voltage Source)

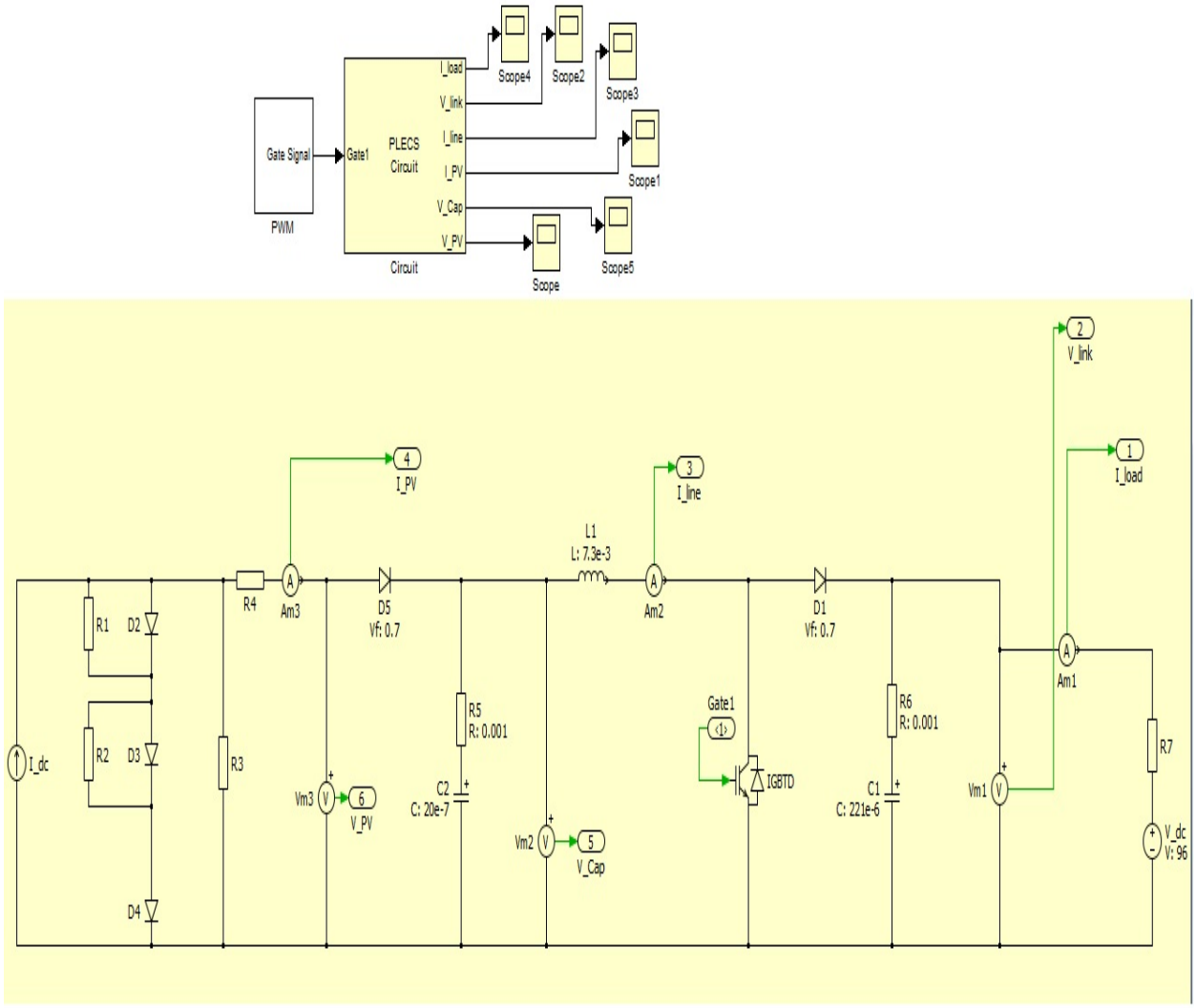


Figure 3.5.21 – Solar Array Boost Converter with a battery load (96 V)

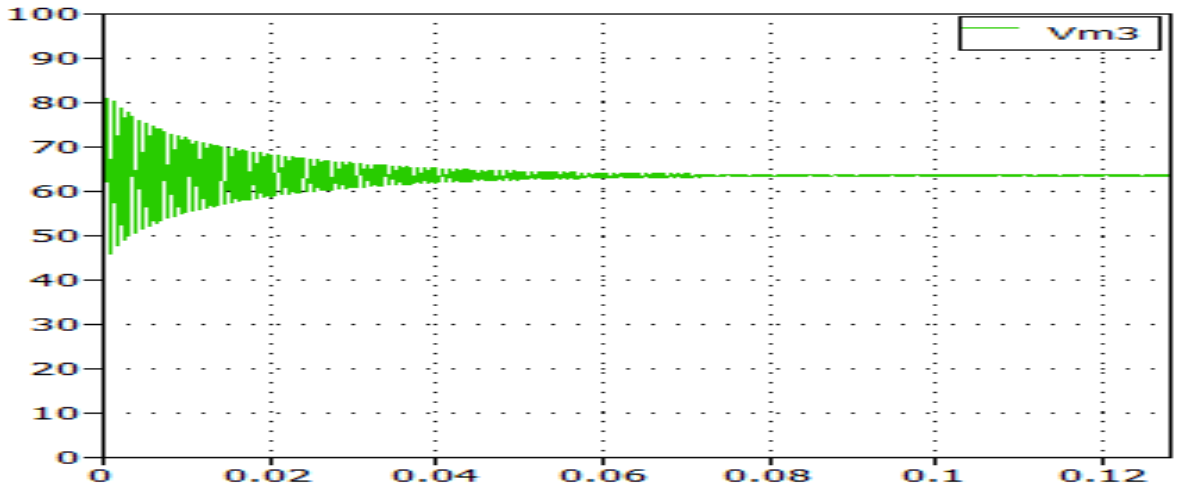


Figure 3.5.22 – V_PV from Figure 3.5.21 (V) vs. Time

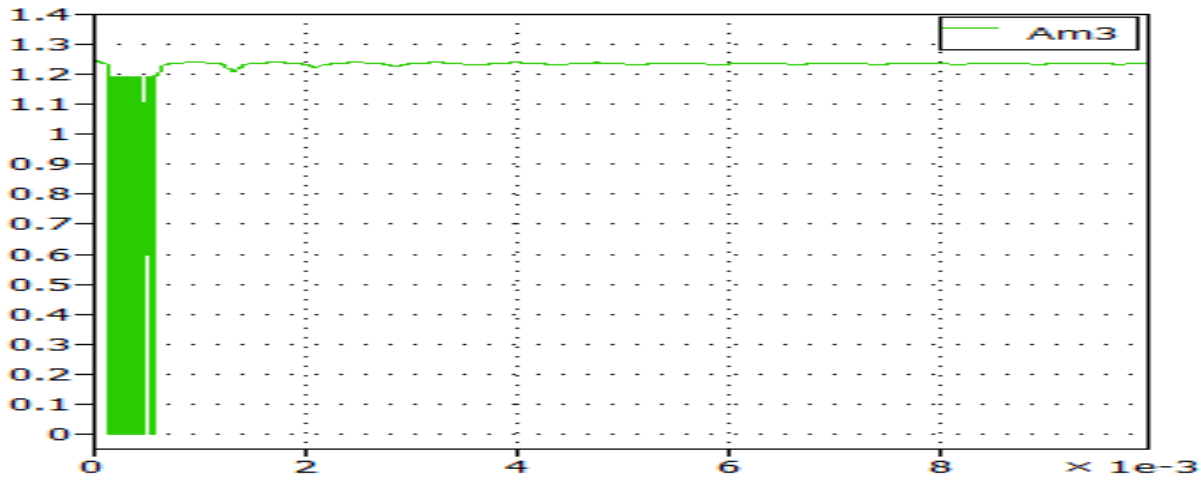


Figure 3.5.23 – I_PV from Figure 3.5.21 (A) vs. Time

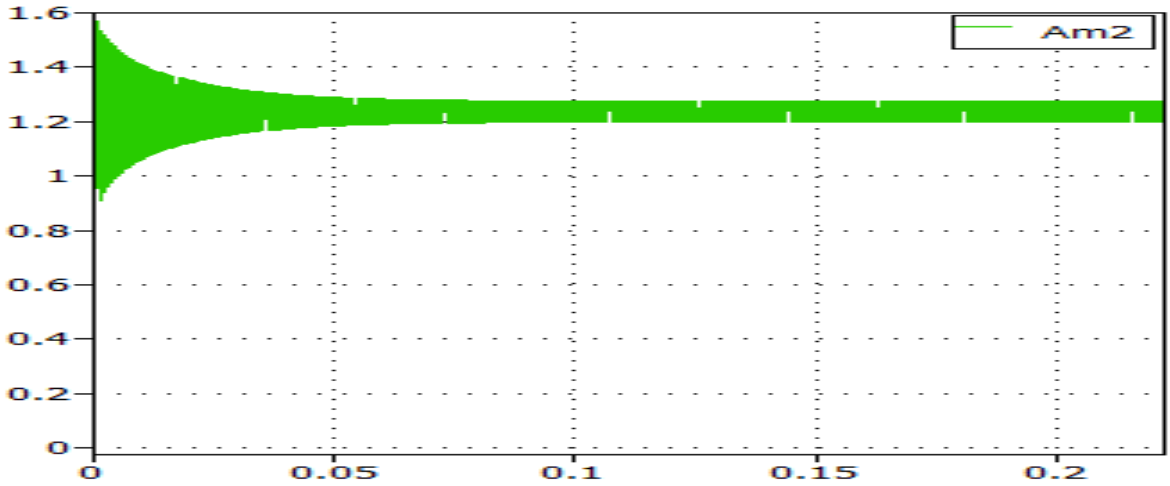


Figure 3.5.24 – I_Line from Figure 3.5.21 (A) vs. Time

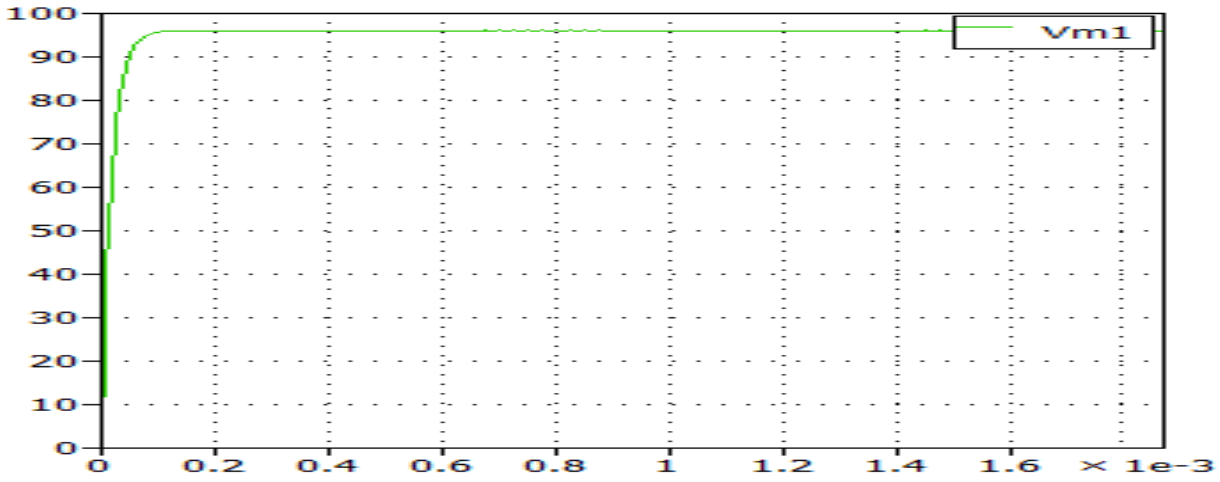


Figure 3.5.25 – V_Link from Figure 3.5.21 (V) vs. Time

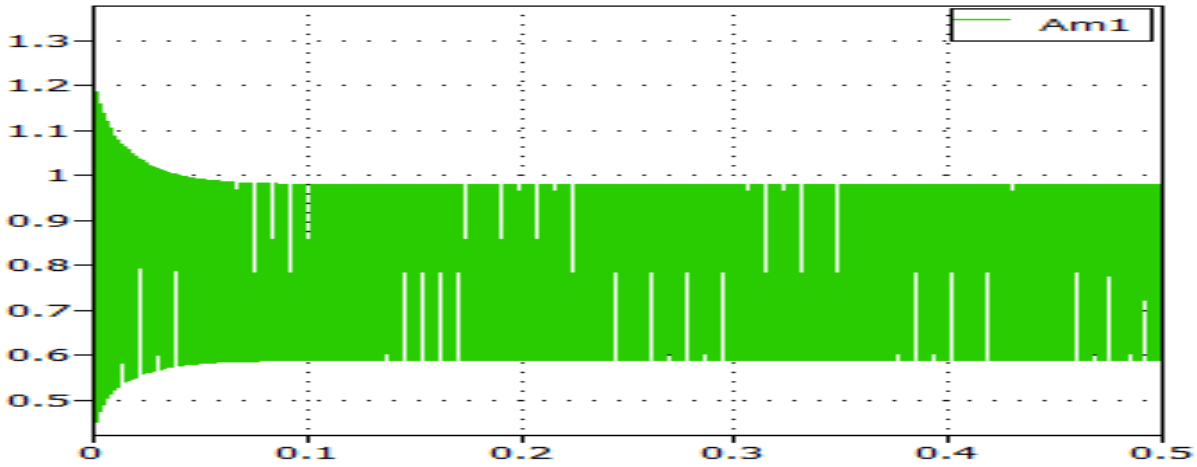


Figure 3.5.26 – I_Load from Figure 3.5.21 (A) vs. Time

In this case, we used the load as a 96 V battery source with some resistance. 96 V is chosen because it is the nominal voltage of the battery bank. The value of the resistance of the battery is chosen from a matlab/ simulink model for lithium polymer battery. From the figures above, we note that the array voltage went down, yet positive current is being supplied to the battery load.

3.5.2.1.5 Case V: Matlab Model of Lithium Ion Battery to find Parameters for Sheperds Equation

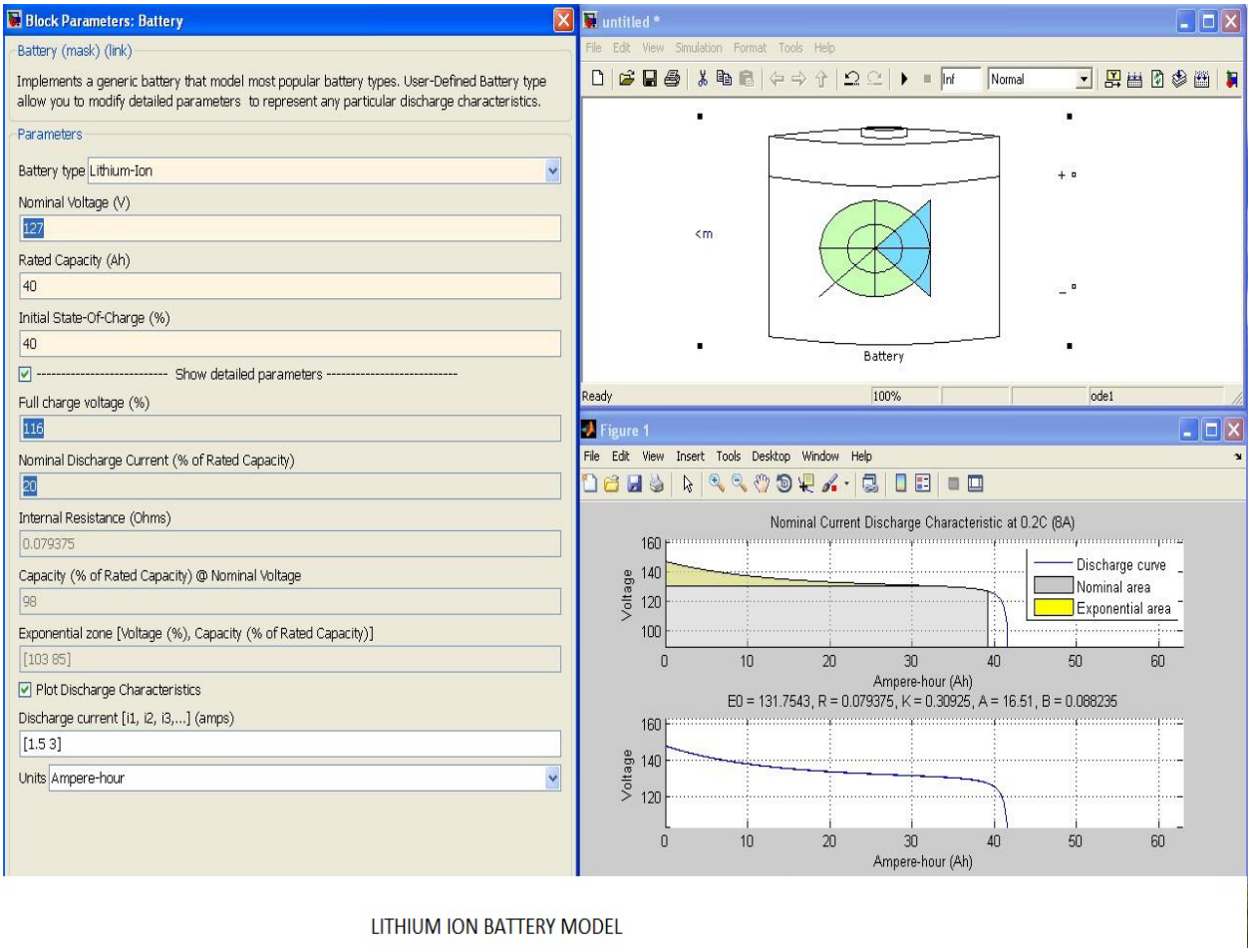


Figure 3.5.27 – Matlab Lithium Ion Battery Model

The battery model in matlab/simulink for lithium ion battery was used and its result is displayed in Figure 3.5.27. Upon entering the manufacturers parameters from the battery datasheet, values of E0, R, K, A, and B is given as an output upon simulation. R represents the Resistance of the battery. The nominal voltage was selected as 126 V. The other parameters are needed as input for shepherd's equation. These values are entered in a battery model parameter discussed in the next case.

3.5.2.1.6 Case VI: Solar Array Model – Boost Converter – Battery Model (Sheperd's equation)

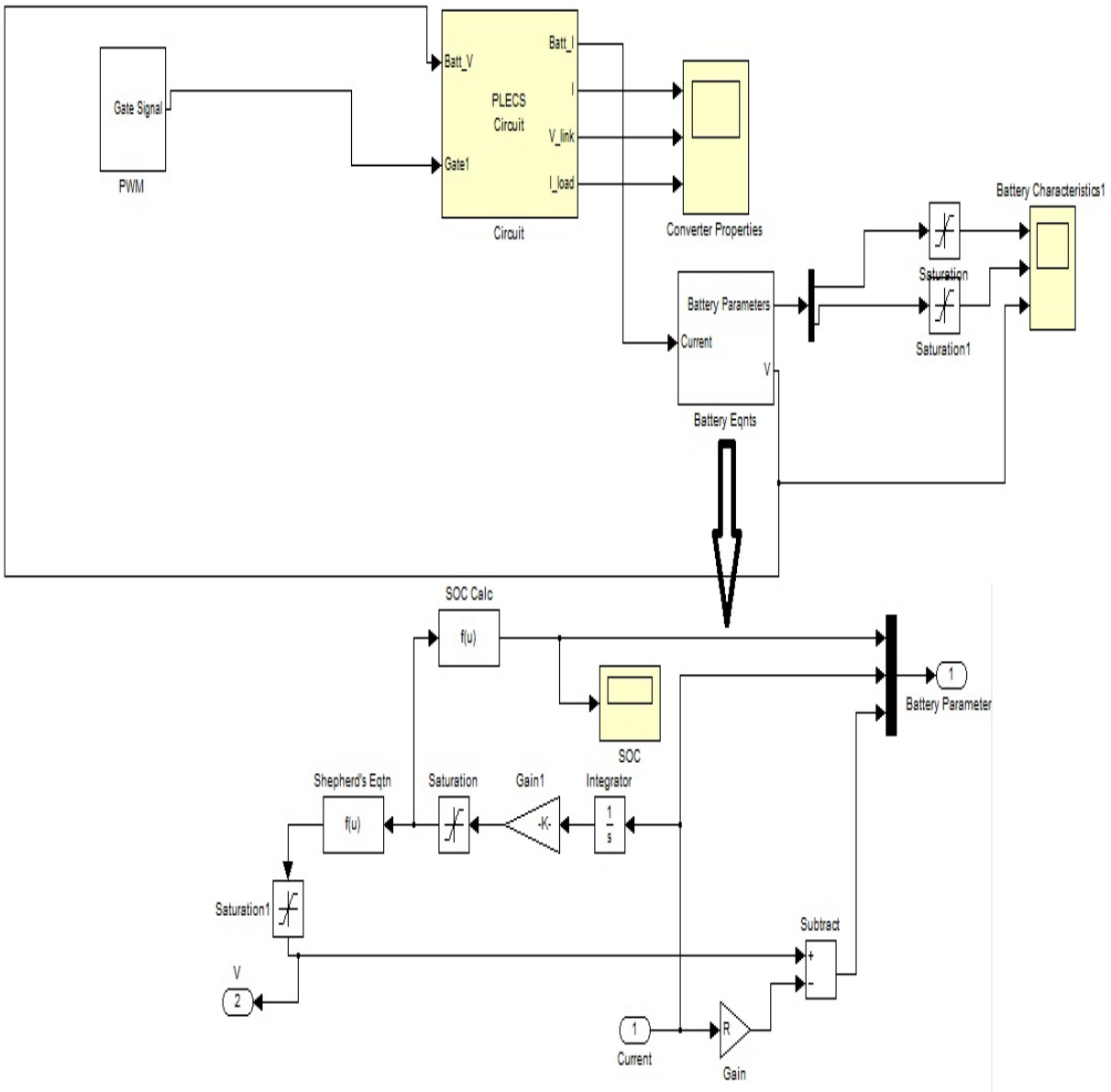


Figure 3.5.28 – Block diagram of Solar Array Boost Converter with battery model (Sheperd's equation)

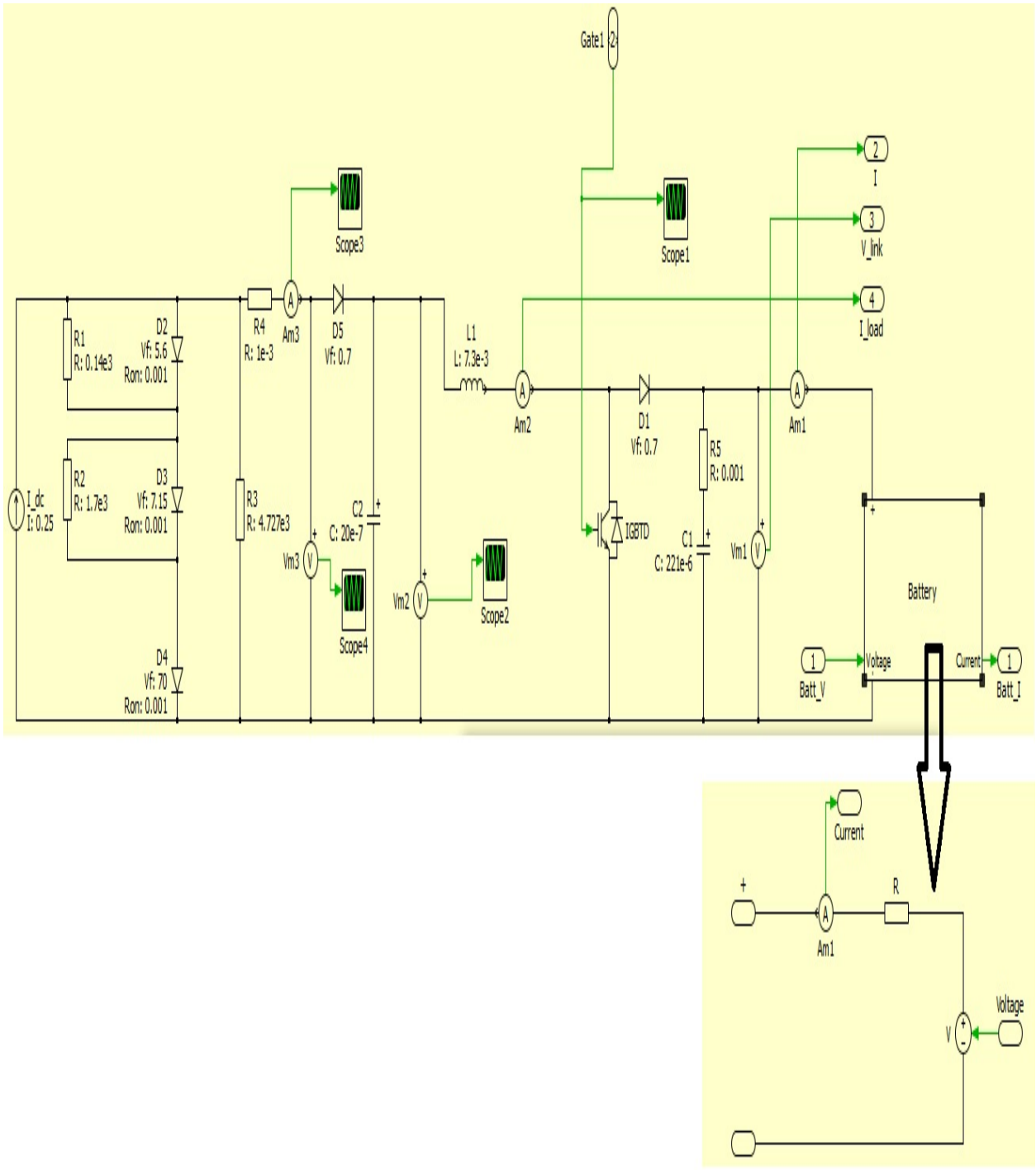


Figure 3.5.29 – PLECS schematic for the Solar Array Boost Converter with battery model (Sheperd's equation)

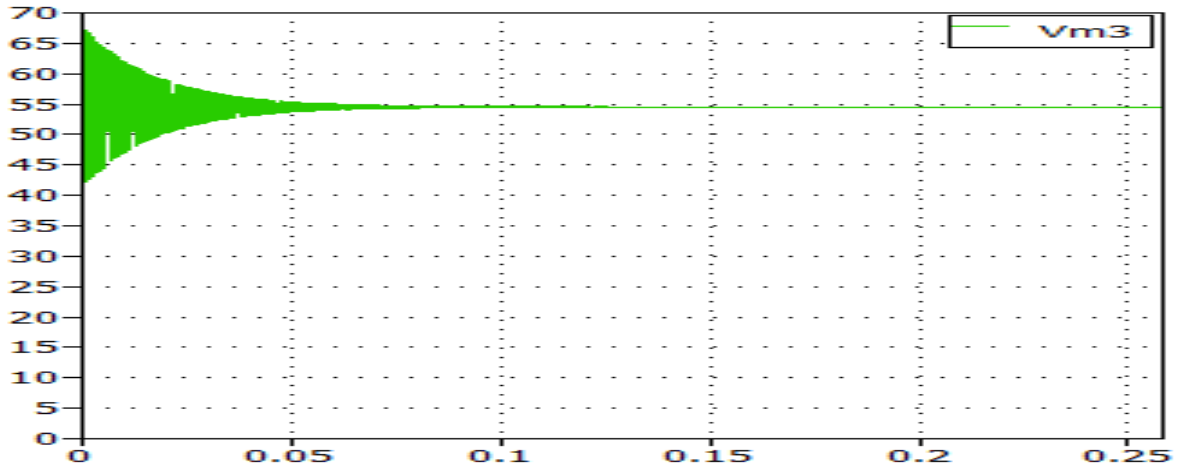


Figure 3.5.30 – V_PV from Figure 3.5.29 (V) vs. Time

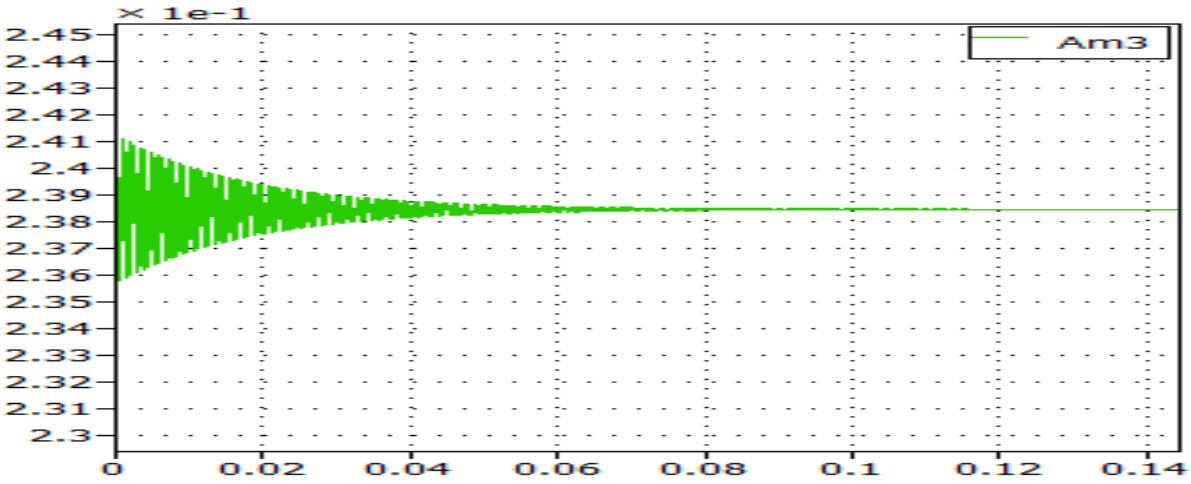


Figure 3.5.31 – I_PV from Figure 3.5.29 (A) vs. Time

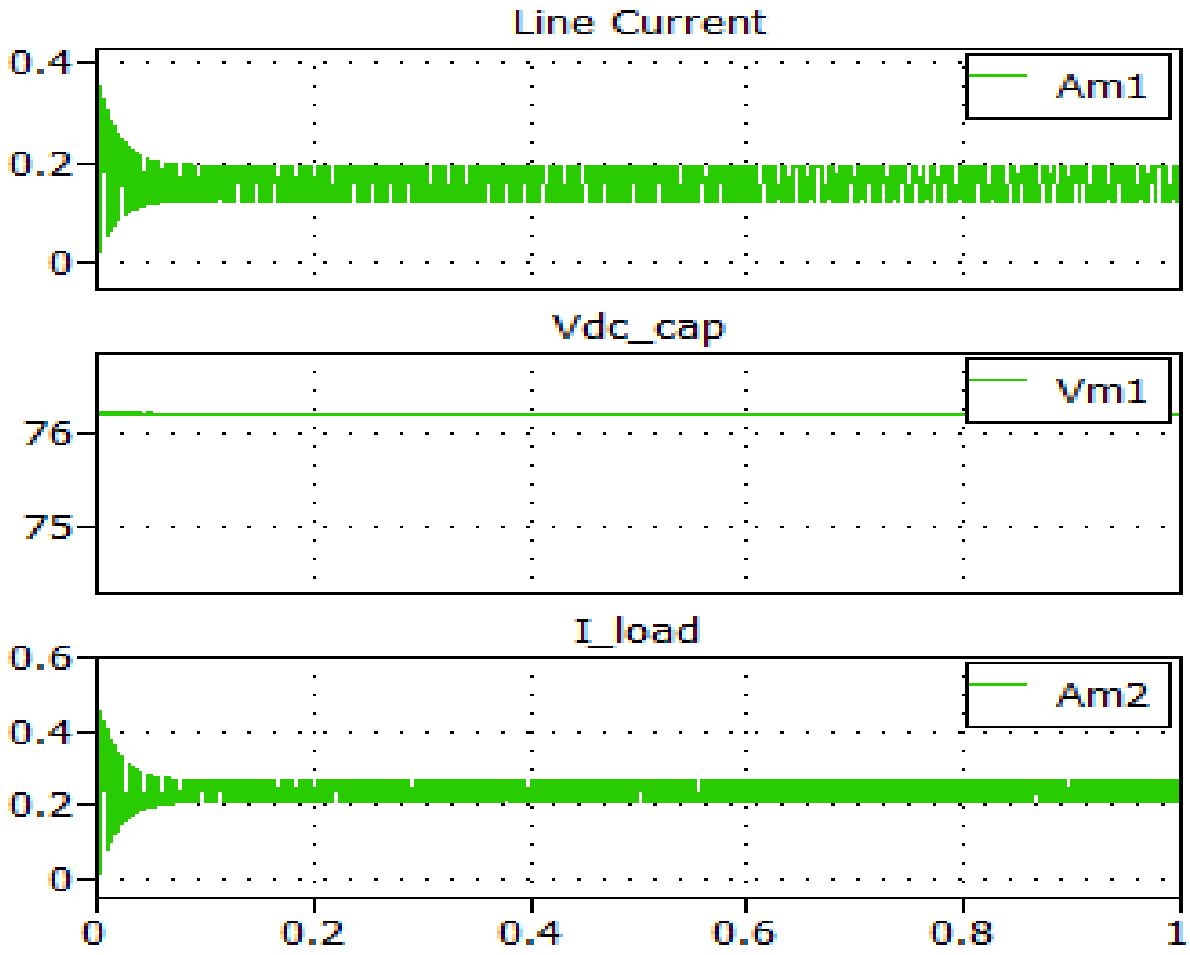


Figure 3.5.32 – Converter properties (load current, dc link voltage, and inductor current)

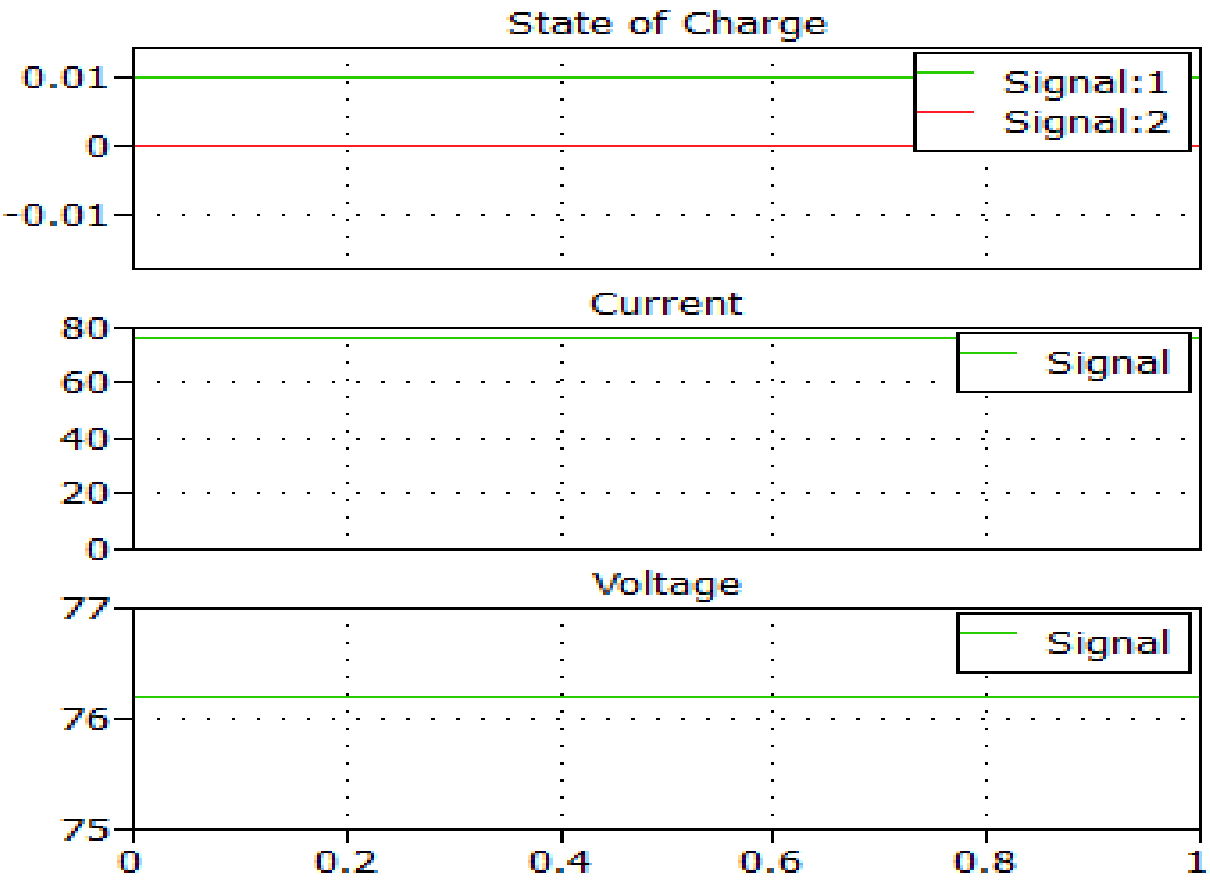


Figure 3.5.33 – State of charge of the battery during simulation

This is the most complex case the team has investigated. It involves solar array model, boost convertor, and a battery model forwarded to the team by Brian Hacker from CAPS. The battery model inputs the parameters of the lithium polymer battery model simulation retrieved from case v. It can be noted from the simulation in the figures above that the solar array is operating a lower voltage and the array current is very low. The DC link voltage is below the nominal voltage set for the battery. This is not good, yet the load current and inductor current are both positive. The state of charge of the battery displays the battery voltage to settle around 76 volts. The DC Link voltage and battery voltage appear to be equal. This model needs to be consulted with Brian Hacker.

3.5.2.2 Incremental Conductance (IncCond) Algorithm

The IncCond Algorithm is a commonly used algorithm in present MPPT's. It performs precise control under rapidly changing atmospheric conditions. The system is capable of tracking maximum power point of the solar panel accurately and rapidly without steady state oscillations. The dynamic performance of this system is satisfactory. Use of the algorithm also results in the elimination of proportional-integral control loop. The incremental conductance algorithm and the duty cycle to the converter are to be controlled by a microprocessor. There are other types of algorithm in use; Perturb and Observe is also a common algorithm. The two algorithms are compared in Table 3.5.3 below.

Table 3.5.3 – Comparison of P&O algorithm vs. IncCond algorithm

Perturb & Observe	Incremental Conductance (IncCond)

Effective extraction of maximum power	Effective extraction of maximum power
Ease of hardware implementation	Ease of hardware implementation
Less sensor requirements	Less sensor requirements
Low Cost	Low Cost
Oscillates around MPP	Does not oscillate around MPP
Considerable Power Lost	Efficient
Slow response in dynamic environment	Fast Response in dynamic environment

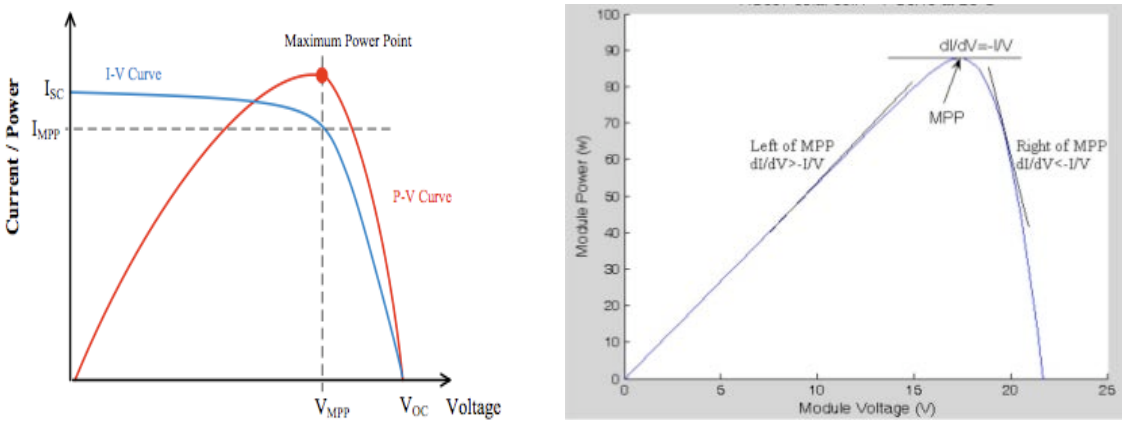


Figure 3.5.34 – Maximum Power Point of PV Panel

Figure 3.5.34 above displays the current, voltage, and power characteristic of a solar panel. The idea of the incremental conductance algorithm is based on the figure that displays the Power –Voltage curve. According to the algorithm and the mathematical analysis associated with it, there are three regions to note in the power curve.

- dP/dV = 0 , or, ΔG = G Region of Maximum Power Point (MPP)
- dP/dV < 0 , or, ΔG < G Region in the Right of MPP
- dP/dV > 0, or, ΔG > G Region in the Left of MPP
- here, G = I / V = Current / Voltage = Conductance

With respect to the above given regions or conditions in the power curve of PV panels, the IncCond algorithm, when applied via a microcontroller, will sense instantaneous current and voltage with the help of sensors. These data will be processed by the microcontroller and the incremental change in conductance with respect to instantaneous conductance will also be computed. Depending upon the region the panel is operating, the microcontroller will then, according to the algorithm, increase or decrease the duty cycle to operate at the maximum power point. Figure 3.5.36 below displays a flow chart of the algorithm followed by a matlab code of the algorithm.

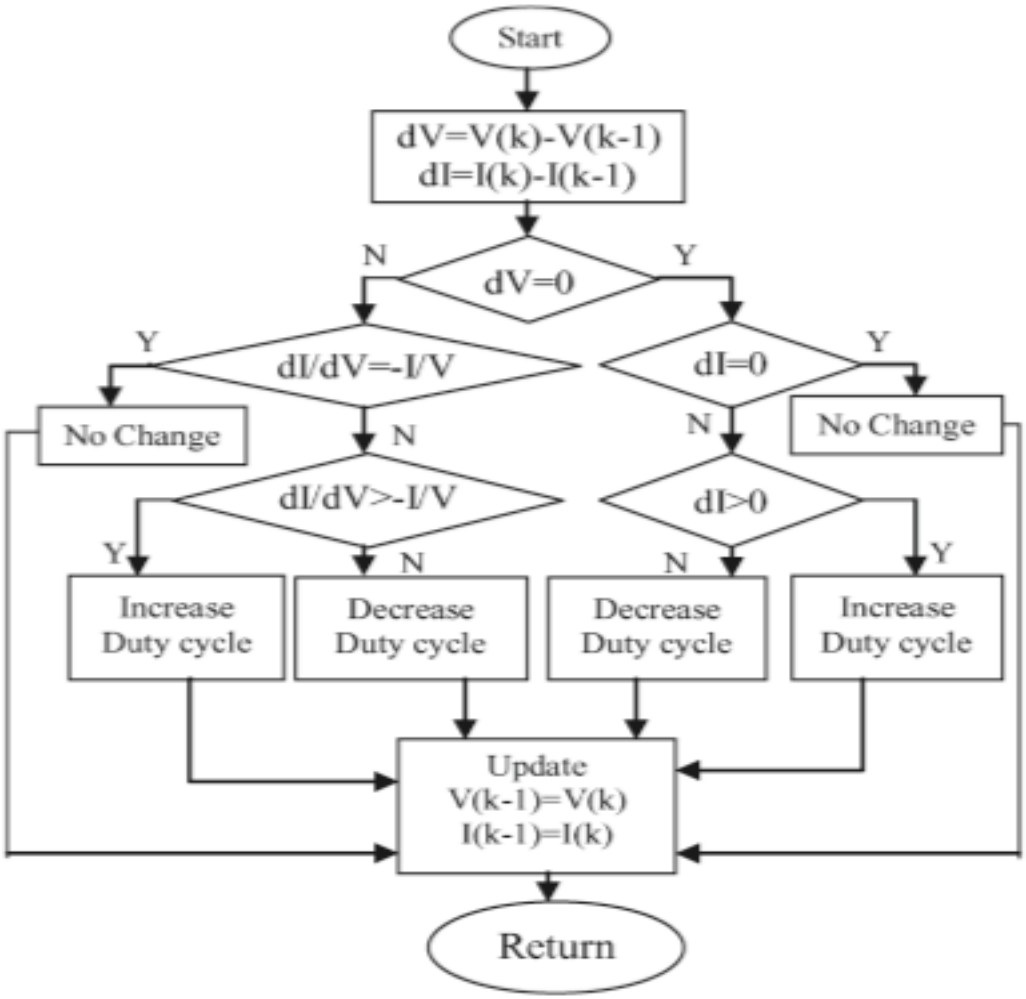


Figure 3.5.35 – Flowchart of incremental conductance algorithm

```
function y = MPPT (u, i, u0, i0, D)
m = 0;
du = u-u0; di= i-i0;
if du == 0
        if di == 0, m = D;
        else
            if di>0, m = D-0.01;
            else
                m = D + 0.01;
            end
        end
        elseif di/du == -(i/u)
        else
            if di/du > -(i/u), m=D-0.01;
            else
                m=D+0.01;
            end
        end
         y = m ;
end
```

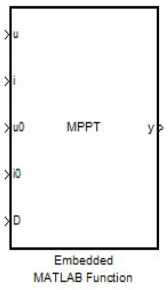


Figure 3.5.36 – Flow Chart and Matlab code for Incremental Conductance Algorithm

3.5.2.3 Control Proposal for MPPT

The team is currently researching into two different control proposal for the MPPT for phase three. One of the control principles the team had looked at is to measure the output voltage and current of the solar array. The current and voltage are measured using sensors in a sampling circuit. These data are sent to the microcontroller via analog-to-digital convertor. The microcontroller looks for the voltage (Vm) of the PV array at the maximum power point using the IncCond algorithm programmed in the microcontroller. The difference between the voltage at the maximum power point and the current panel voltage is sent to the PI controller. The output of the PI controller is the varying duty cycle D via the limiter. The output signal of the limiter is sent to the CPLD circuit which will generate a 40 kHz PWM pulse of various different duty cycles. The microcontroller controls the duty cycle. The PWM pulse is signaled to the IGBT via a driving circuit. Figure 3.5.37 below shows a control proposal for the MPPT. It is followed by Figure 3.5.38, a matlab/plecs simulation circuit of the current control scheme the team is seeking.

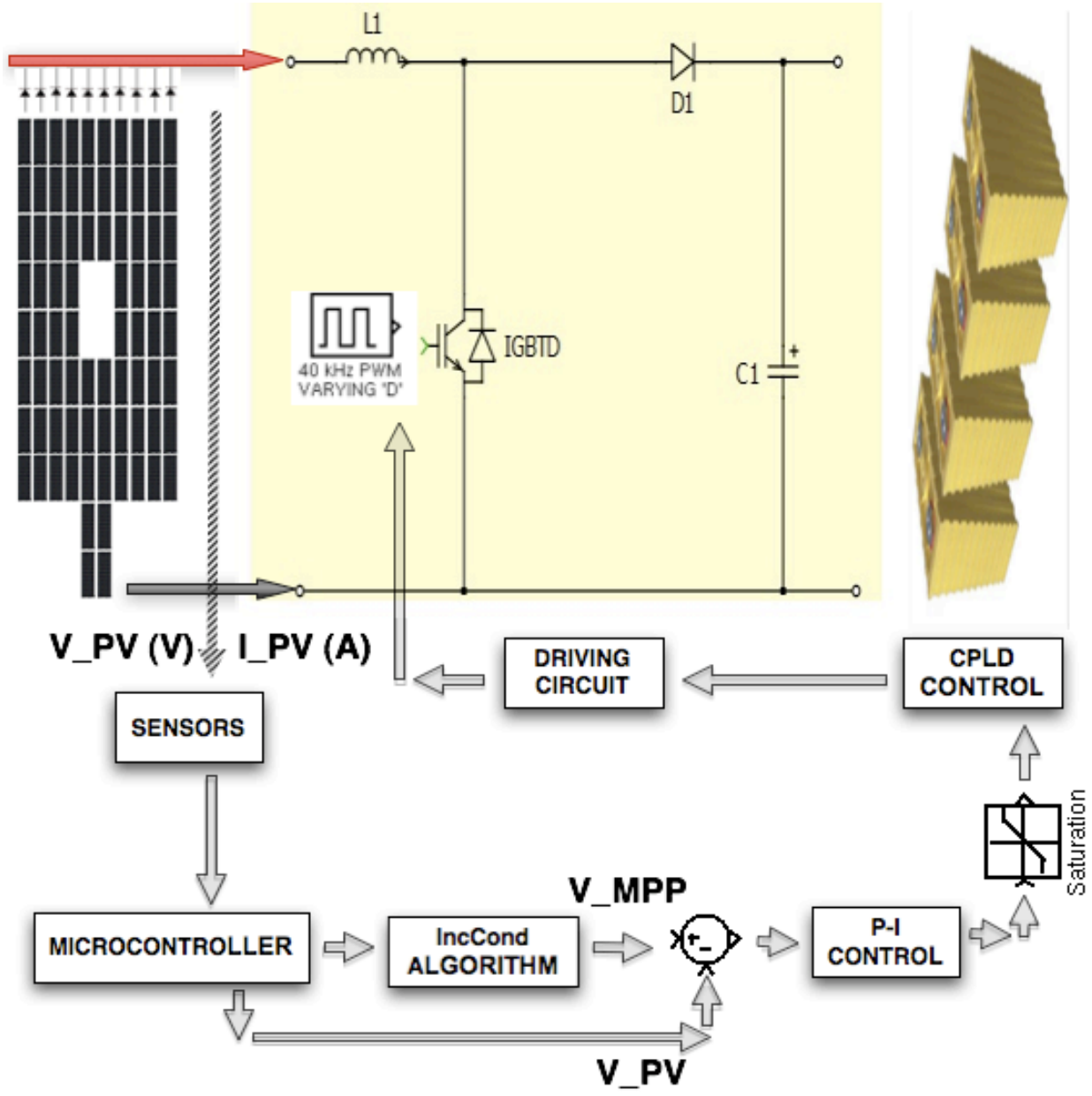


Figure 3.5.37 – Initial Control Proposal

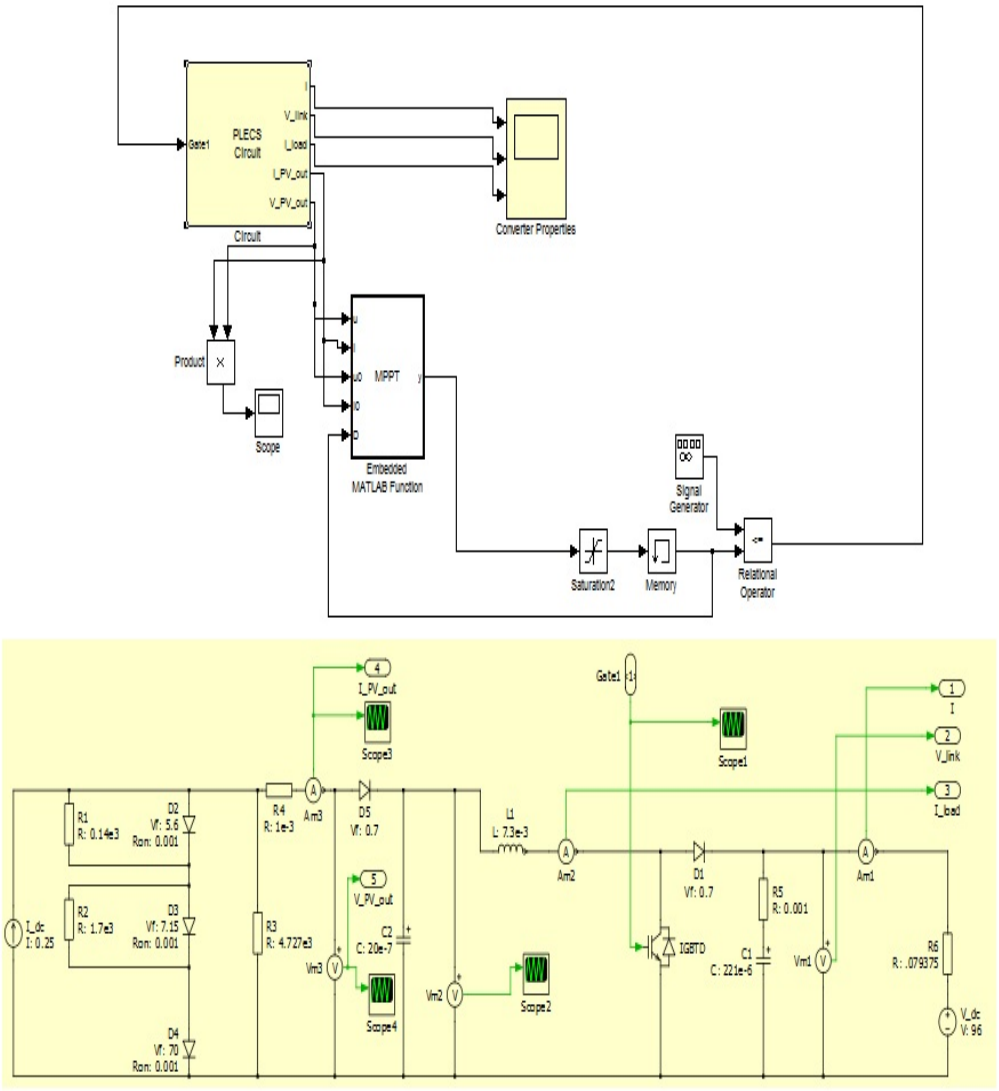


Figure 3.5.38 – MPPT Control Simulation Schematic Proposal

The load for the converter is a voltage source (96V, nominal car battery voltage) along with its internal resistance. The MPPT Matlab code based on incremental conductance algorithm is implemented. The output of the MPPT (embedded Matlab function) is saturated, stored in memory, compared to a sawtooth waveform of amplitude 0.375 (duty cycle), and the resultant pulse width modulated signal is sent to the IGBT of the boost converter. Initial run on simulation displayed the following error: “Unable to locate 'mexopts.bat', and therefore cannot determine which compiler to use for simulation builds. Use 'mex -setup' to select a supported compiler.” The error is being investigated by the team, and a conclusion will be presented in the final design presentation. We give sincere thanks to Jesse Leonard, Dr Saritha, and Brian Hacker for providing us the model and guiding us in the design and simulation of a MPPT connected to a solar array with a lithium battery as a load.

3.5.3 Regenerative Braking

The regenerative braking system will convert kinetic energy of motion into electrical energy. This electrical energy is stored as charge in the battery bank. The regenerative braking system, upon asserted, will change the polarity of the motor; as such, the motor essentially behaves like a generator. Regenerative braking along with mechanical friction will provide total braking output.

From the previous phase a potentiometer was utilized in order to send the information signal to the motor controller, enabling the regenerative braking. The potentiometer was mounted next to the driver and through the use of a handle the driver could activate the potentiometer to the desired level. This could place the driver at a mild risk because one must remove a hand from the steering wheel and other system controls. Integrating the regenerative braking and mechanical braking was a goal for this phase of the project.

There were two methods discussed as possible solutions. First, when the mechanical brake was first engaged a signal could be passed to the microcontroller and in turn utilizing one of the I/O pins on the development board send the signal to the motor controller. This electrical solution would require the use of a switch on the brake pedal along with programming the microcontroller to pass this signal. The other option was to keep the potentiometer and mechanically fabricate the pedal to implement both the mechanical and regenerative braking simultaneously. Upon discussing the two options it seemed apparent that utilizing the existing potentiometer would be just as simple as attempting to attach a switch to the pedal. For the sake of simplicity and to reduce the amount of wires in the car, the braking was implemented by mechanically integrating the two braking systems.

3.6 Control Systems

The primary task of the control system is to provide a means to the operator to control the car and make available to the operator the current status of the car's components. The driver must have full control of all the systems of the vehicle during operation, and must also be provided with telemetry information and the status of system components. Figure 3.6.1 shows the dashboard from the previous year's design.

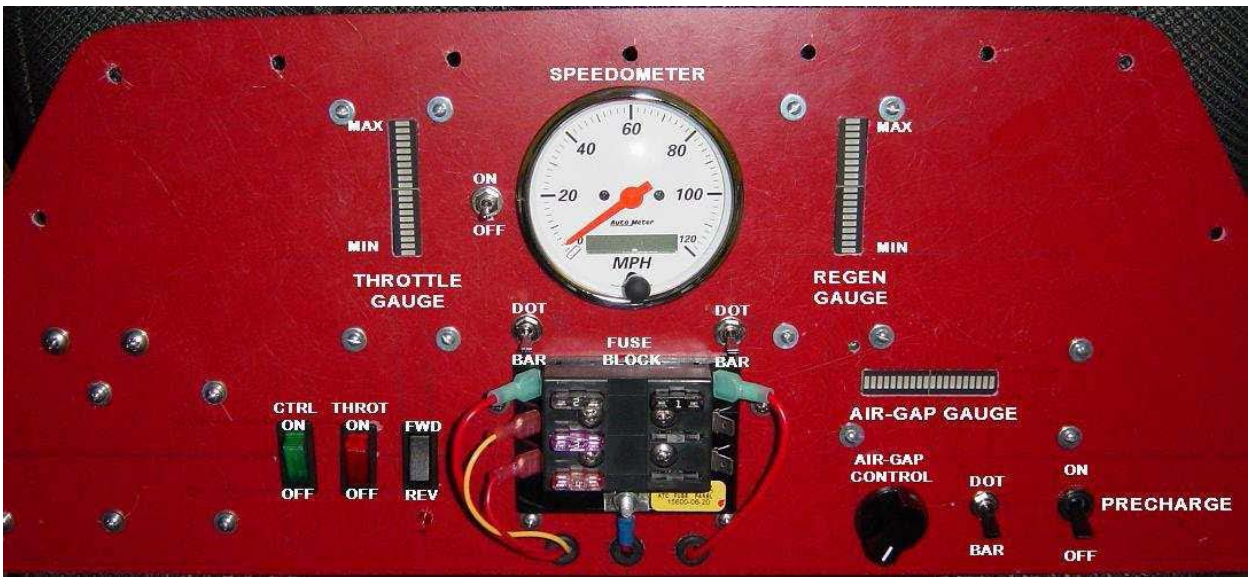
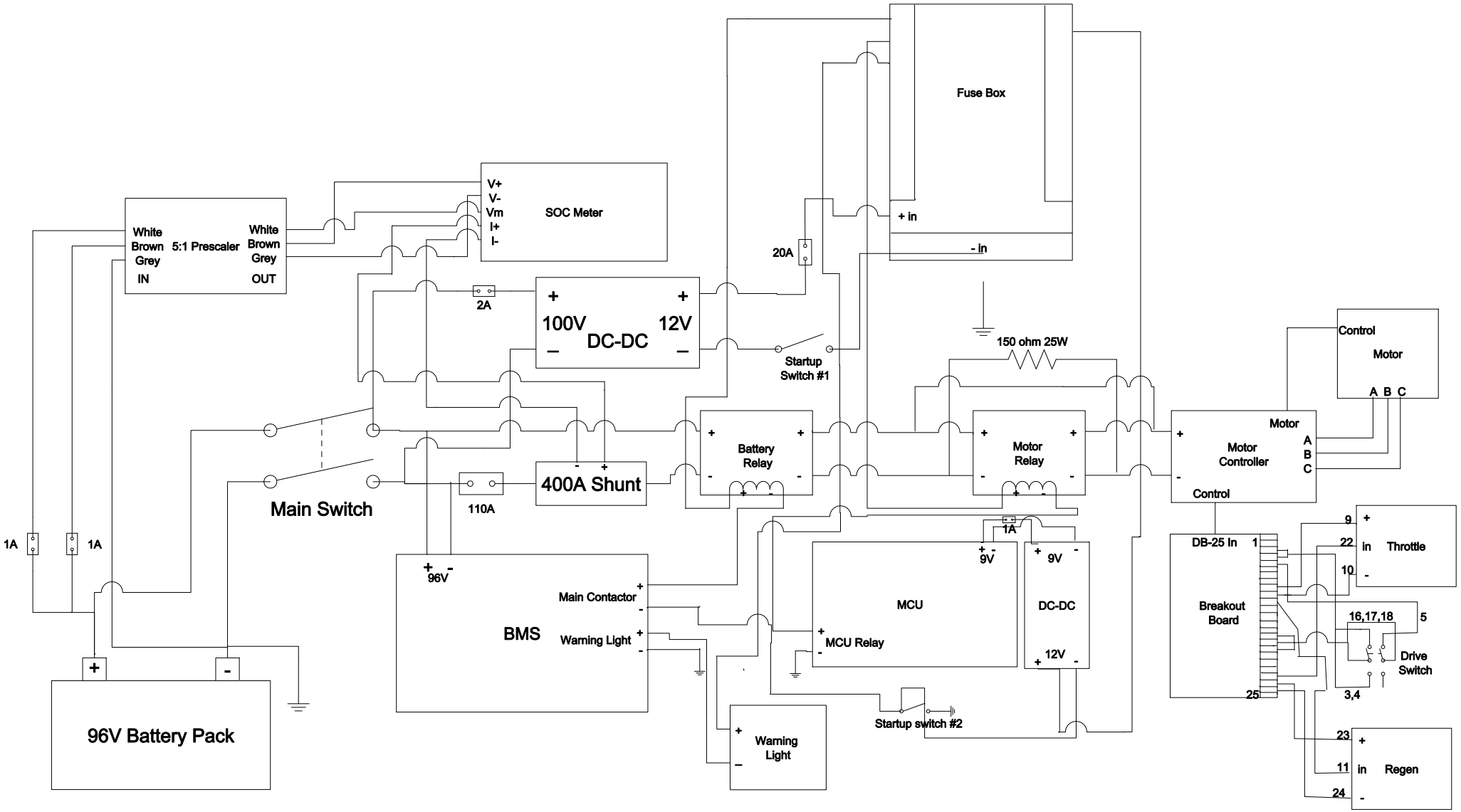


Figure 3.6.1 – Phase I Dashboard


Fuse Box
+ in
- in
20A
V+
V-
Vm
I+
I-
SOC Meter
White
Brown
Grey
IN
5:1 Prescaler
White
Brown
Grey
OUT
2A
100V
DC-DC
12V
Startup
Switch #1
150 ohm 25W
Control
Motor
A B C
Motor
Motor
Controller
A
B
C
Control
Battery
Relay
Motor
Relay
400A Shunt
Main Switch
110A
1A
1A
1A
9V
9V
DB-25 In
1
Breakout
Board
25
9
22
10
in
Throttle
16,17,18
5
Drive
Switch
3,4
96V
Main Contactor
BMS
Warning Light
MCU
MCU Relay
DC-DC
12V
Startup switch #2
Warning
Light
23
11
24
in
Regen
96V Battery Pack

3.6.1 Master Control Unit

The master control unit (MCU) will function as the interface between the driver, the motor controller, and the battery management system (BMS). The MCU needs to be able to communicate serially with the motor controller. The microcontroller chosen is the Wytec Dragon12 Plus-USB development board shown in Figure 3.6.2. The Dragon12 Plus uses a Freescale HCS12 16-bit Microprocessor which is designed for use in automotive applications. This board was chosen because it was the only board that could be found that contains all of the I/O components need for the process.

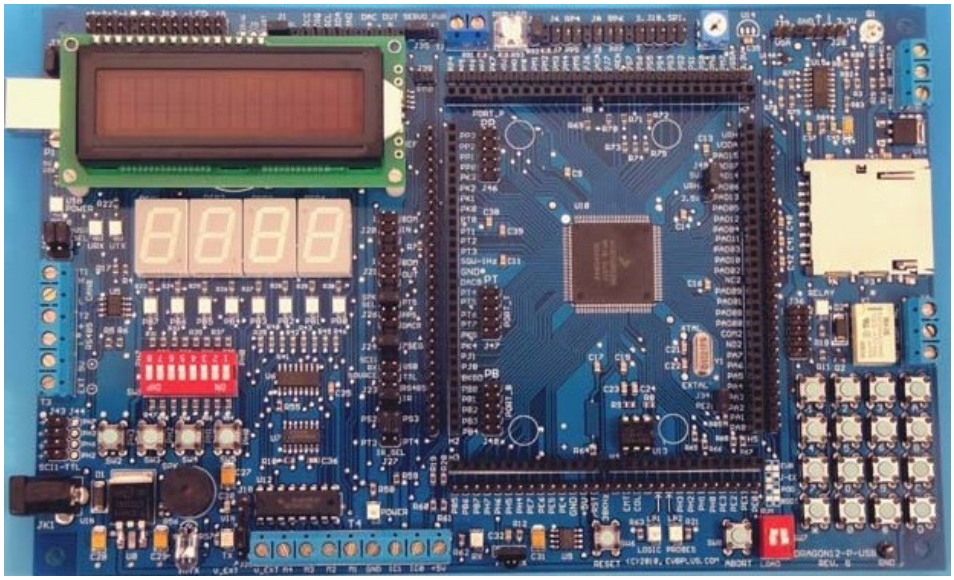


Figure 3.6.2 – Wytec Dragon12 Plus-USB

The code for the microcontroller was written using the Freescale CodeWarrior IDE software. CodeWarrior is offered for free by Freescale for use in development of applications using Freescale's products. Using an IDE will increase productivity and provide simulation, debugging, and programming capabilities in order to decrease development time.

The board needs to be powered by a 9V source. In phase I of the project only two voltage sources were available, 100V and 12V sources. A DC-DC converter was then sought out in an attempt to change one of the aforementioned sources to a suitable source for the microcontroller. Due to availability a DC-DC converter was purchased that would convert the 12V to 9V usable by the microcontroller.

3.6.2 Motor Controller

The motor controller is used to power the motor. It is controlled by a microcontroller that can be accessed through a serial interface. The motor controller has two control modes, discrete control mode and serial control mode. The car is currently configured to operate in discrete mode. The main advantage of discrete mode over serial mode is that is it easier to implement. However, in discrete mode, there is no access to the internal functions of the motor controller's microprocessor, which contains very useful diagnostic, and status data and motor control mode settings. Figure 3.6.3 shows an example of a discrete control configuration for a motor controller.

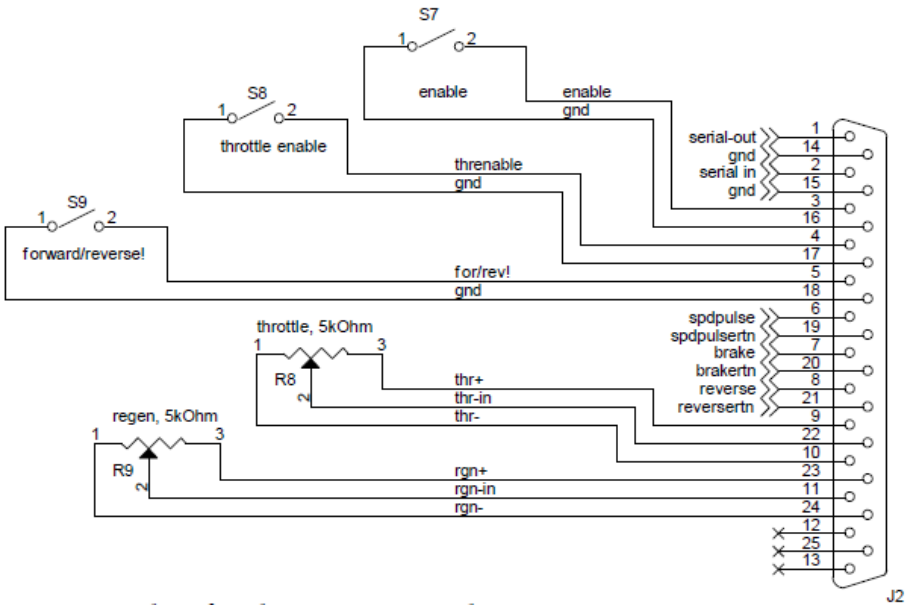


Figure 3.6.3 – Motor controller discrete control configuration

The motor controller has been reconfigured for a combined discrete and serial mode operation. The MCU has been connected to the motor controller through the serial interface. The throttle potentiometer will remain connected directly to the throttle input to the motor controller, but the microcontroller will read the data from the motor controller and use it to make adjustments to the air gap of the motor based on the data collected from the motor controller. All of the diagnostic data produced by the motor controller has been collected by the MCU and processed to be available to the driver. The forward/reverse switch and the throttle enable switch has been connected the MCU which will send the command to the motor controller serially.

3.6.3 Dashboard

The dashboard will act as the medium between the driver and the car's operation. The dashboard implemented by the previous phase, as seen in Figure 3.6.1, was overly complicated and it was decided that simplicity would be more ideal for this car. For example, the previous dashboard had several LED lights that would inform the driver when the throttle or regenerative braking are being engaged. While this information is useful for testing purposes, it seems overly redundant to inform the driver of such information, since the driver will most assuredly know when the throttle is engaged by the motion of the vehicle.

The previous phase has eight different switches on their dashboard, so this should drastically reduce the complexity for the driver. First it was discussed which of the switches could be removed utilizing circuit logic or the microcontroller. For example one of the switches was to implement the pre charge circuit for the motor controller. The solution to this problem was to initiate the pre charge as soon as power became available in the system, by flipping the main contractor switch. The microcontroller was then programmed with a 10 second delay, which would prevent the driver from using the throttle or regenerative braking until the pre designated period of time. Previously the driver could flip the switch and immediately engage the throttle, which in turn could damage the motor controller. By implementing the microcontroller to perform this function a measure of safety was also added to the components. Similarly as was discussed before the LED lights would be removed from the dashboard, freeing another three switches which were implemented to control these LEDs. Finally the throttle and control enable switches were integrated in parallel since the driver will never implement on without the other.

The dashboard for this phase of the project will consist of three switches and the state of charge display, as can be seen in Figure 3.7.5. One two position switch will be used to power the fuse box. The fuse box provides the power for all devices operating at 9V including the BMS and relays. The second two position switch, as mentioned above, is used to power the microcontroller, which in turn will implement the pre charge circuit for the motor controller. This means with the flip of that switch the motor will be accepting all commands from the discrete signals of the potentiometers. Finally a three position rocker switch will be used to designate motor direction. When the center position the car will be in neutral, essentially the motor will not be engaged, and the other two positions will designate forward and reverse.

This simplified version of the dashboard will allow the driver a greater focus on the driving itself as a pose to finding the correct switch during operation. The more intuitive design will also allow a wider range of driver to comfortably take the wheel and operate the vehicle without extensive amounts of training.

3.7 Management Systems

The management system will consist of the batteries, the battery management system, cell modules, circuit protection elements (i.e. relays and fuses), and state of charge devices. The motor and motor controller are also an integral portion of this system because the motor controller is a management device. The first phase of the project was able to get bring the motor to a fully functioning state and because there are no foreseeable modifications necessary will not be discussed in this portion of the paper. The batteries for the system have already been purchased in the previous phases and will again be utilized during this phase of the project. Due to the importance of the batteries in the final product the rest of this system will be designed to meet the needs of these batteries. Figure 3.7.1 and Figure 3.7.2 show a large majority of the electrical components and control devices for the car.

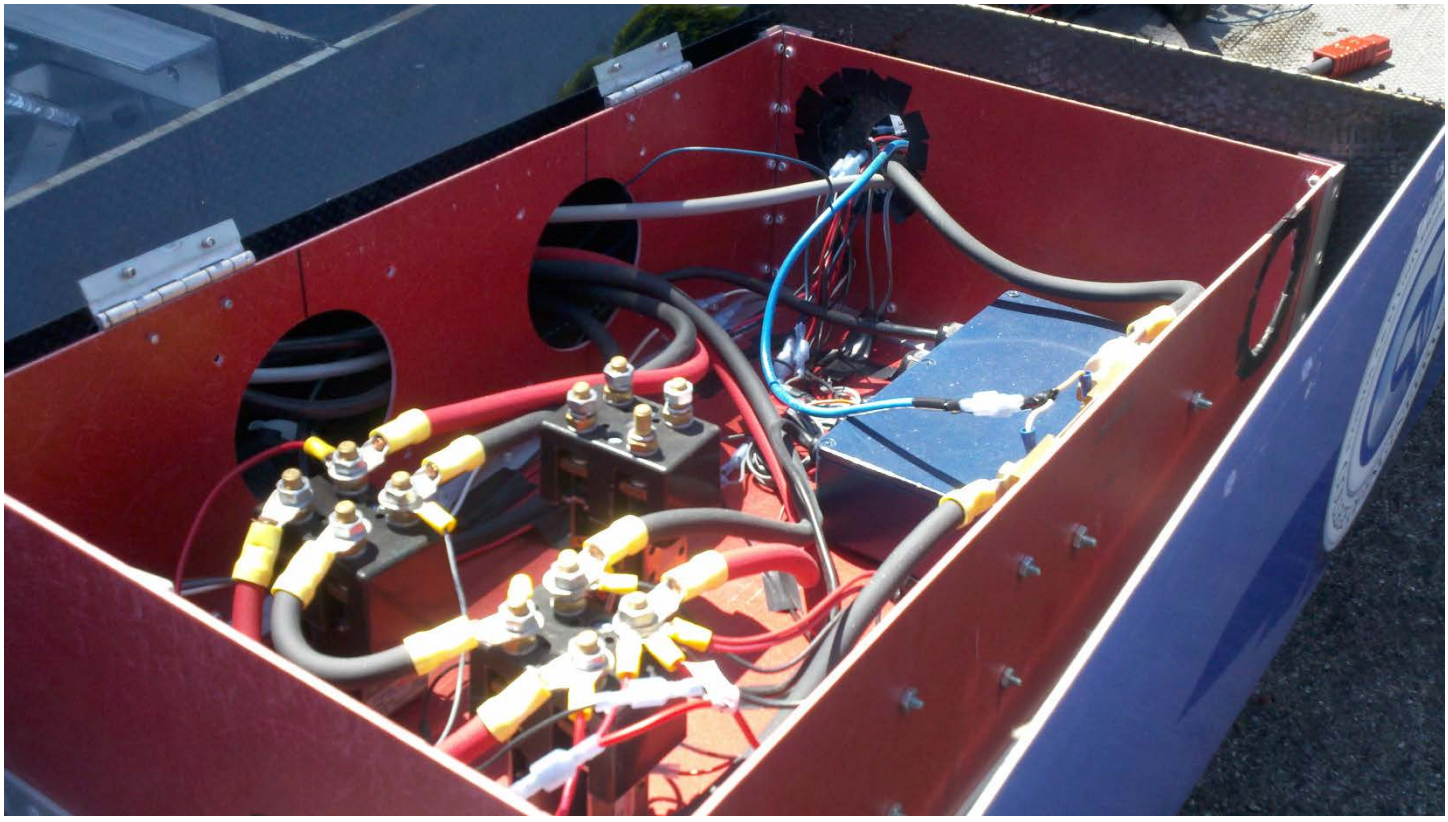


Figure 3.7.1 – The high power components are protected by a fiberglass box

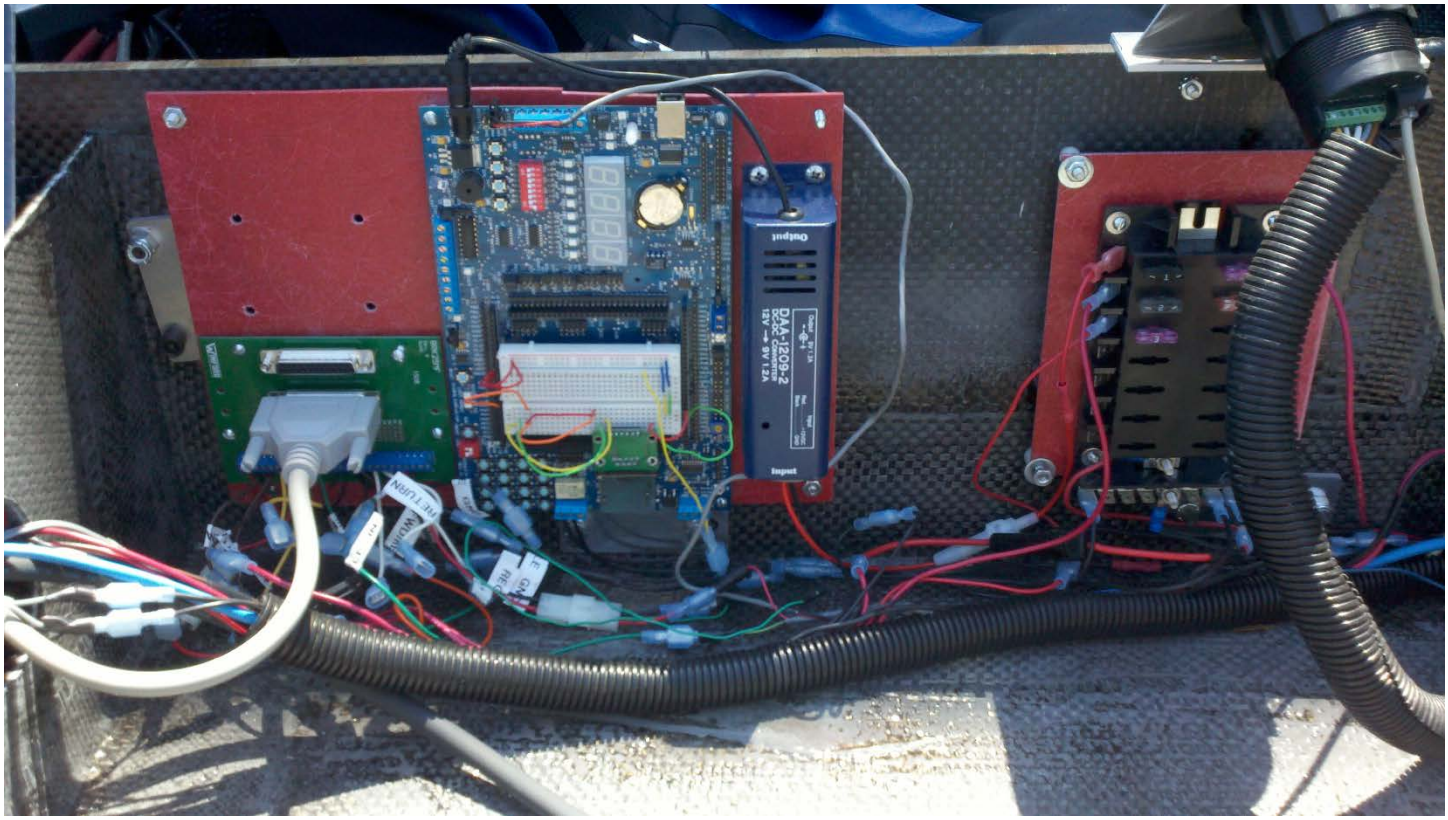


Figure 3.7.2 – Fuse box, DC/DC 12 V to 9V, MCU, and breakout board

3.7.1 Batteries

Currently thirty Thundersky batteries have been implemented into the system. Each cell has an ideal operating voltage of 3.2 V and therefore the system as a whole will operate at 96 V. The only time that the batteries are outside of this range should be at a point of complete charge or at complete discharge. Figure 3.7.3 shows the discharge cycle of the Thundersky batteries.

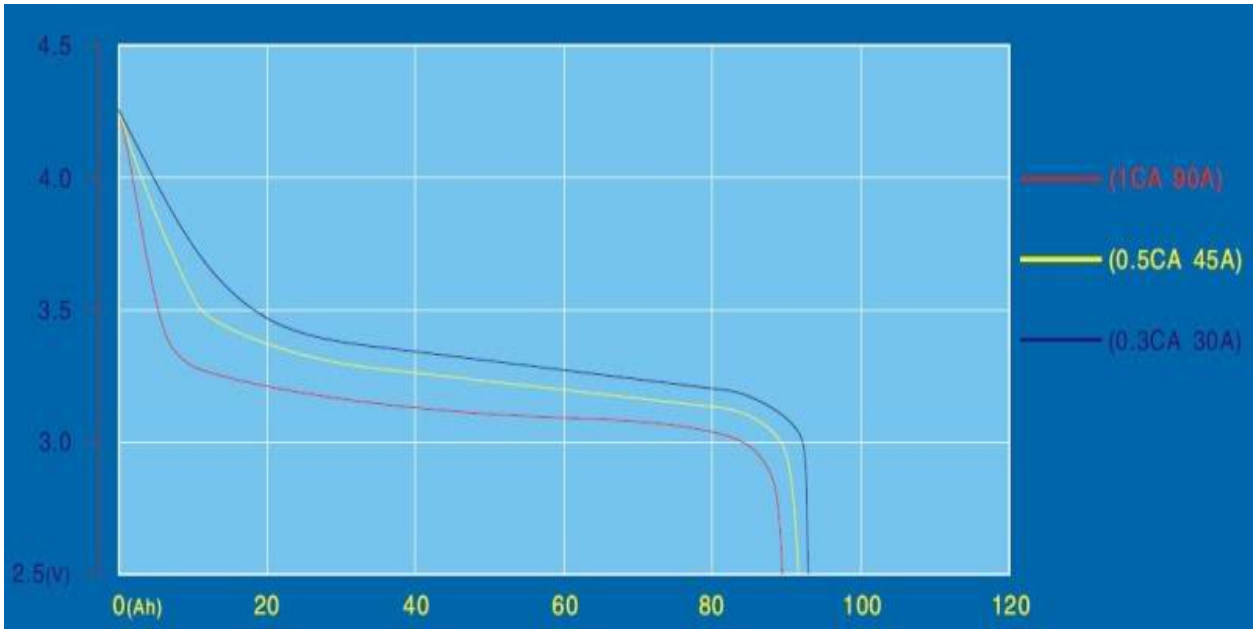


Figure 3.7.3 – Graph of Thundersky battery discharge cycle (Endless-Sphere)

The Thundersky batteries like all batteries have ideal operating parameters and must be kept in this range or risk causing damage to the batteries. A protection circuit will have to be implemented to keep the batteries in the safe range. The protection circuit will be designed to the ideal operating ranges displayed in Table 3.7.1.

Table 3.7.1 – Safe battery operating parameters

Protection Type	Restraining Value
Over Voltage	4.25 V
Under Voltage	2.5 V
Over Current	120 A
Over Temperature	75 °C

Several devices have already been purchased in conjunction with the batteries to simplify this state of charge monitoring. A battery management system (BMS) designed specifically for these batteries will be utilized as a means to isolate the batteries from the rest of the electrical system. The battery management system contains four signal wires through which passes a small current. As long as the signal circuits are closed then the BMS will allow operation of the batteries. As soon as one of the signals is broken then the BMS will slowly power down the batteries and finally separate them entirely from the rest of the system utilizing one of the normally open relays as seen in Figure 3.7.4. This will be the controlling device to prevent batteries from out of bounds conditions.

Figure 3.7.4 – High power electrical relay used as a safety device (Tecknowledgey, 2002)

The voltage protection for the batteries will utilize a cell module device attached to each of the batteries. The signal wire from the BMS will be run through each of the cell modules. During the operation of the vehicle the cell module will monitor the voltage potential of the individual battery that it is attached too. When the battery is in a safe operating range the cell module will be a closed circuit and when outside of this range the cell module will be an open circuit. If even one of the cell modules is an open circuit then the BMS will begin the shut down phase.

Current protection for the batteries will be through the use of fuses. Throughout all of the electrical circuitry many fuses will be placed as a means to protect all equipment from shorts. The main fuse between the battery and the rest of the systems, where the largest current will be flowing, a 110 amp fuse has been installed. The thundersky batteries are capable of discharging at 3C or three times capacity, meaning a maximum of 120 A (40Ah * 3). However the motor is only rated for up to 9 kW, which at 100V is about 90 A. In practice most electricians will utilize a fuse rated for the maximum current draw plus twenty to twenty-five percent, to account for various fluctuations, which would mean for this system a fuse rated around 110 A would be desirable. The fuse was designed based solely on the motor draw since the entirety of the other electrical components used only .3 A to operate during testing. If a fuse does break during operation the driver will have to rely on the mechanical systems, such as the steering and braking and remaining momentum of the car to pull off to the side of the road.

Finally the temperature protection for the batteries will come directly from the state of charge device. A ring terminal is connected directly to one of the contacts on the battery itself and therefore should accurately measure the internal temperature in the battery. This ring terminal will communicate the information to the state of charge device and allow the driver to view the core battery temperature at all times. It will be entirely the responsibility of the driver to observe the temperature of the batteries and stop the car when temperatures exceed the 75°C threshold. During vehicle operation, when outside temperatures were 76°F, the core temperature of the batteries only rose to 90°F, well short of the 167°F over temperature mark.

3.7.2 State of Charge

The state of charge being used is a multi-functional measuring and display device. In Figure 3.7.5, one can see the display which will be mounted into the car. The arrows allow the driver to cycle through the various display information

which will include: a battery fuel gauge, voltage levels, current currently being drawn, amp-hour use since start up, and temperature.

Figure 3.7.5 -- TBS Electronics E-Xpert Pro (Evolve Electrics: TBS Electronics E-Xpert Pro)

This specific state of charge device is designed to work for potentials up to 35V. For this application the potential across all the batteries is close to 100V at most times and therefore a prescaler was needed to scale the voltage potential from the batteries to an appropriately measurable potential for the state of charge device. The prescalar was hooked between the positive and negative terminals of the batteries and the output give to the state of charge device where the information could be provided to the driver.

To measure the current out of the batteries a shunt line will be used. A shunt line will be a connection across the wires from the batteries and will be capable of measuring the current in those wires. This information will be delivered to the state of charge device in the dashboard so that it can be displayed. The information will also be delivered to the microcontroller, interpreted, and then delivered to the BMS. This way if for some reason the current exceeds the value rated for the batteries (120 A) then the BMS can perform he shut down sequence as before.

The state of charge device will be programmed with the amp-hour capacity of the batteries and in return will calculate the remaining capacity of the batteries by continually measuring the current output. This information is highly desirable for any vehicle; imagine driving to drive a car without knowing how much gas is left in the tank. Utilizing this information will also allow the driver to notice power trending, such as how much power is used during accelerating or at certain speeds. This information can be used by the driver to get the most mileage out of the car, by choosing to drive at speeds where the current flow is at its lowest. It is unclear whether the state of charge system will recognize power being placed back into the batteries through the power generation systems. Ideally it would read a negative current value and make corresponding corrections to the fuel gauge. This information will have to be determined through testing after the part has arrived.

The temperature as explained in the previous section was measured using a ring terminal hooked directly to the contact for a battery. Allowing the driver to access all this information would be one more step towards the over safe operation of the vehicle. It will allow the driver to recognize potential threats to the electrical system hopefully long before the issues become dangerous and allow the appropriate action to take place.

4 Test Plan

A test plan document was created by the members of the previous phase as seen in Figure 3.7.1. In order to keep consistency this phase of the project will also implement the same test plan document. This template displays all the pertinent information about each test, including what is being tested, the goals of the tests, and final results. A well organized system for testing will yield a more successful product in the end.

Scheduled Test Reporting Form – Solar Car Team '09

TEST ITEM (TITLE):

TEST CASE #: (ex: BS-001) TEST DATE/TIME: (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION: TEST TYPE: ☐ TEST ☐ RE-TEST

include Objective and Requirement(s)

EXPECTED RESULTS:

ACTUAL RESULTS:

STATUS: ☐ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Figure 3.7.1 -- Blank Test Plan Format

4.1 System and Integration Test Plan

4.1.1 Mechanical Part Integration

For streamline integration, it is imperative for part to be assembled and tested before implementation into the vehicle. Testing of all parts will be performed for fit, strength and proper performance. Each test plan will be conducted using proper documentation as per requirements. Some parts may require construction to be tested; notes will be made with an estimated test date.

4.1.2 Electrical Part Integration

The electrical system integration will begin with the testing of the main power system, ensuring that all of the components are thoroughly tested for correct wiring and are receiving power. The System will then be configured to operate the motor in discrete mode to verify the functionality of the motor and motor controller. After the motor has been fully tested, the motor controller will be integrated with the MCU and retested using the serial mode controls.

4.2 Test Plan for Major Components

4.2.1 Body

To ensure strength in the carbon fiber material, a tensile test will be performed on a strip of test material. The test will be conducted using a one inch strip of the 12K carbon fiber material pregnated with the polyester blend of resin to be used on the bottom portion of the body. Again, to ensure proper strength, another tensile test will be done on the 3K carbon fiber material with the same polyester blend resin for the top half of the body.

4.2.2 Steering

For the steering system, the rack and pinion gear was tested. The rack and pinion was transferred from the previous year's car and has been re-implemented into the new car design. Once installed there were various components of the system that was checked.

Figure 4.2.1 – Tie Rod

The location of the rack and pinion will be positioned behind the suspension so the tie rods were checked to ensure they reach the steering arms on the wheel.

The new geometry of the steering system may affect the steering capabilities of the rack and pinion gear. Once the gear has been installed, the vehicle must perform several maneuverability tests in order to compete in the race. The car must be able to make a U-turn in either direction, without backing up such that all wheels remain within a 16 m wide lane.

The rack and pinion gear must be tested to ensure that the gears mesh well with each other. The gear must translate the rotational motion of the steering column into linear motion of the rack. The rack and pinion must also be able to push the tie rods effectively to ensure proper steering.

Figure 4.2.2 – Rack and Pinion Gear

4.2.3 Braking

Wilwood Engineering sponsored the solar car this year and has given the team various braking components to utilize in the cars design, one of these being the master cylinder. Due to the vehicles light weight, go kart master cylinders were used to supply hydraulic fluid to the brake caliper.

Two master cylinders were purchased from Wilwood Engineering. Once the Master Cylinders were obtained they were tested to ensure they provide adequate fluid pressure to each caliper assembly. Figure 4.2.3 shows the master cylinders to be purchased.

Figure 4.2.3 -- Wilwood Master Cylinder

Once installed, the master cylinder was tested to ensure it can apply hydraulic pressure to the brake calipers. To test this, pressure was applied to the brake pedal to force fluid through the brake lines. This also tested the integrity of the brake lines. Any leaks in the lines will be found by applying pressure to the pedal. If there is a leak in the line, the brake line must be adjusted or changed in order to keep hydraulic pressure.

Two brake calipers were purchased from Wilwood Engineering. These calipers are shown in Figure 4.2.4. Once the obtained the calipers were tested to ensure they can effectively push the caliper piston. When hydraulic fluid is pumped

to the caliper, the piston must move towards the brake rotor to stop the vehicle. Once pressure is released from the caliper, the piston must retract smoothly away from the rotor.

Figure 4.2.4 -- Wilwood brake calipers

The brake rotor will be fabricated at the machine shop. This custom rotor was tested to ensure it can effectively be used to stop the car as friction is applied to it. As friction is applied to the brake rotor, heat is generated throughout the component. The rotor was to ensure it can dissipate heat effectively. If heat is not dissipated from the rotor, rotor failure is imminent. To prevent this, the heat transfer capability of the rotor was tested by using finite element analysis. Figure 4.2.5 shows the brake rotor undergoing finite element analysis.

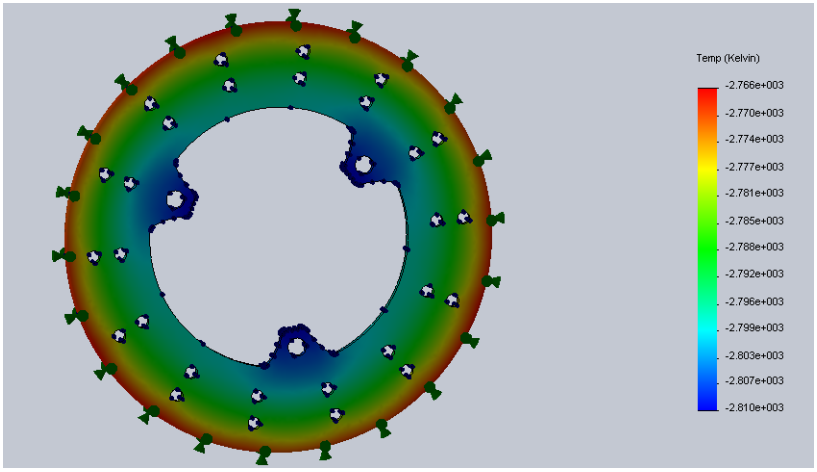


Figure 4.2.5 -- FEA on brake rotor

4.2.4 Suspension

Tests were performed on the suspension's components and as a whole system to ensure desired and efficient performance.

4.2.4.1 Components

Finite element analysis (FEA) was performed on each of the suspension's components to observe the stress points and their deflections under the transferred forces it will experience from the wheel under bump and rebound conditions. This testing checked for part failures by performing analysis at various nodes of the virtual mesh. The components tested were: control arms, and upright arms for both front and rear suspension systems. Refer to Test Plan Appendix section.

4.2.4.1.1 Lower Control Arm

Fixed fixtures were set in each of the holes where the heim joints are threaded. A 500 lbf was applied in the hole containing the heim joint that joins the upright, and a reaction force of 500 lbf was split amongst each hole containing the heim joint that joins the arms to the body. After performing FEA, this resulted in a max von misses stress of 6.1 psi, which is less than the yield strength of 10998.1 psi. Consequently, the lower control arm won't yield as shown in Figure 4.2.6.

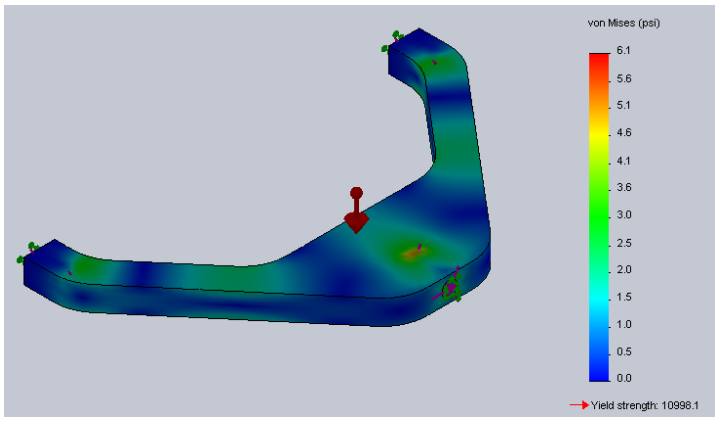


Figure 4.2.6 – Von Misses Stress Analysis on Lower Control Arm

4.2.4.1.2 Upper Control Arm

The same fixture and load characteristics of the lower control arm were applied to the upper control arm. After performing FEA (Figure 4.2.7), the max von Misses was 3.9 psi, which is less than the yield strength of 10998.1 psi. Thus, the upper control arm won't yield.

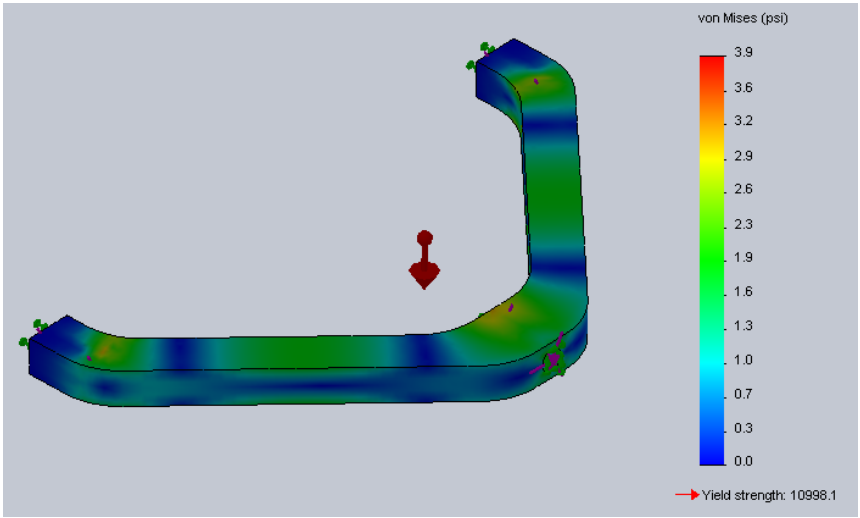


Figure 4.2.7 – Upper Control Arm Von Misses Stress Analysis

4.2.4.1.3 Upright

FEA was performed on the upright with hinge fixtures at the brackets and a fixed fixture on the contact face with the spindle. The forces acted radially in each hole where the bolts go through holding the spindle to the upright. A total

force of 500 lbf was applied over these four points resulting in a max von Misses stress of 422.2 psi. This resulting stress is lower than the yield strength of aluminum 2024 of 10998.1 psi, thus the upright won't yield under this load. This result is shown in Figure 4.2.8.

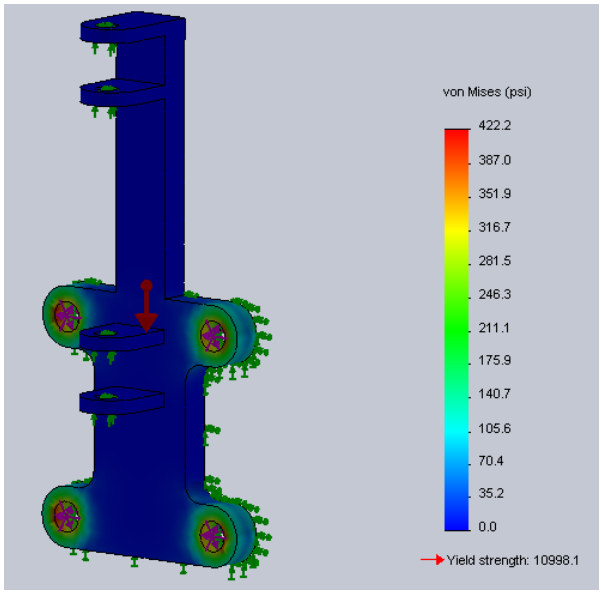


Figure 4.2.8 – Von Misses Stress on Upright

4.2.4.2 Suspension System Virtual Simulation

The assembled front suspension system was analyzed and simulated in MSC ADAMS/Car, as shown in Figure 4.2.9. These simulations provided the data needed to observe the suspension's behavior and adjust the dimensions to achieve the desired results. Parallel wheel travel was performed to observe the behavior of the designed suspension. The results obtained by these simulations include camber and caster angle change, as well as static loads and forces acting on the system.

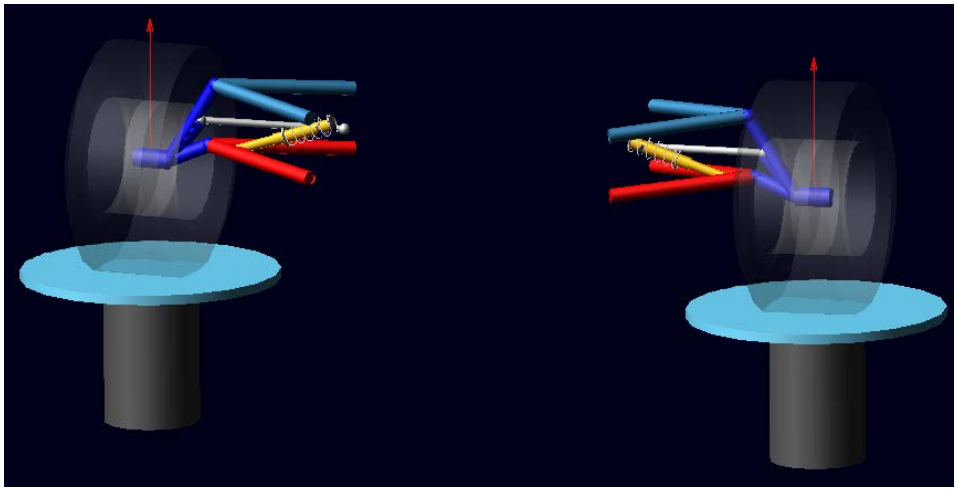


Figure 4.2.9 – MSC ADAMS/Car Parallel Travel

The resulting characteristics of camber and caster angle for the front suspension are very good. Both of these experience a small change as the wheel travels through bound and rebound; a two inch travel (from -1 to 1). Caster angle, shown in Figure 4.2.10, experiences a change of 0.0456° ranging from -0.0606° to -0.0150°.

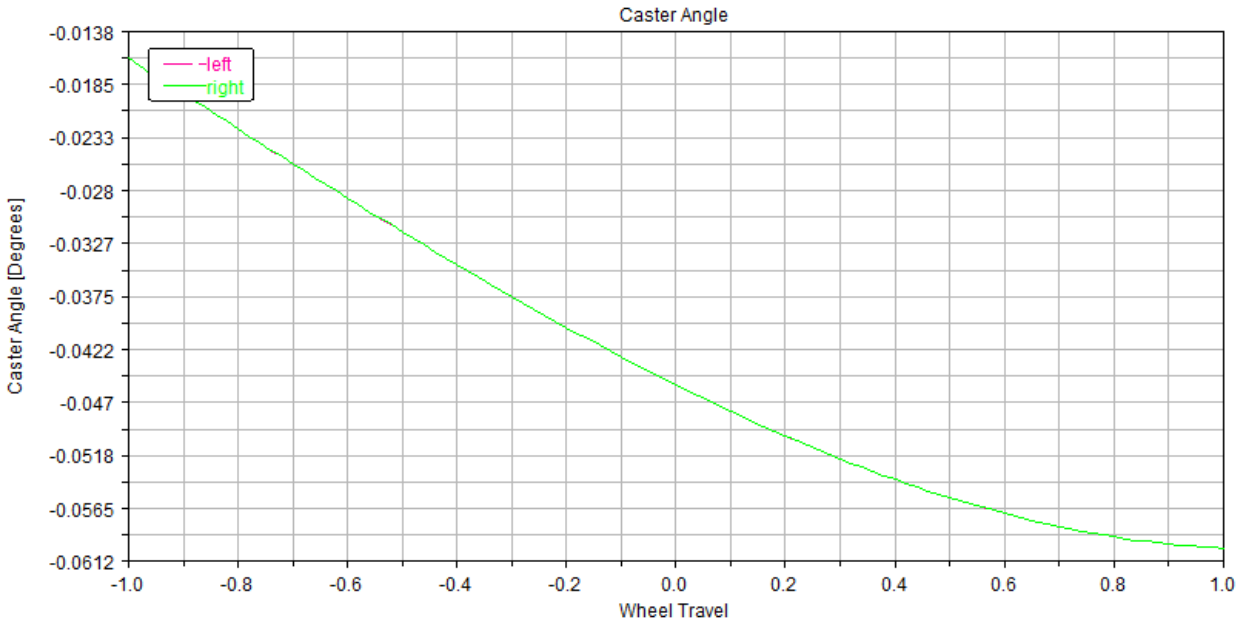


Figure 4.2.10 – Caster Angle vs. Wheel Travel Plot

The camber angle, shown in Figure 4.2.11, also experiences a small change of 0.19° ranging from -1.1° to -0.91°.

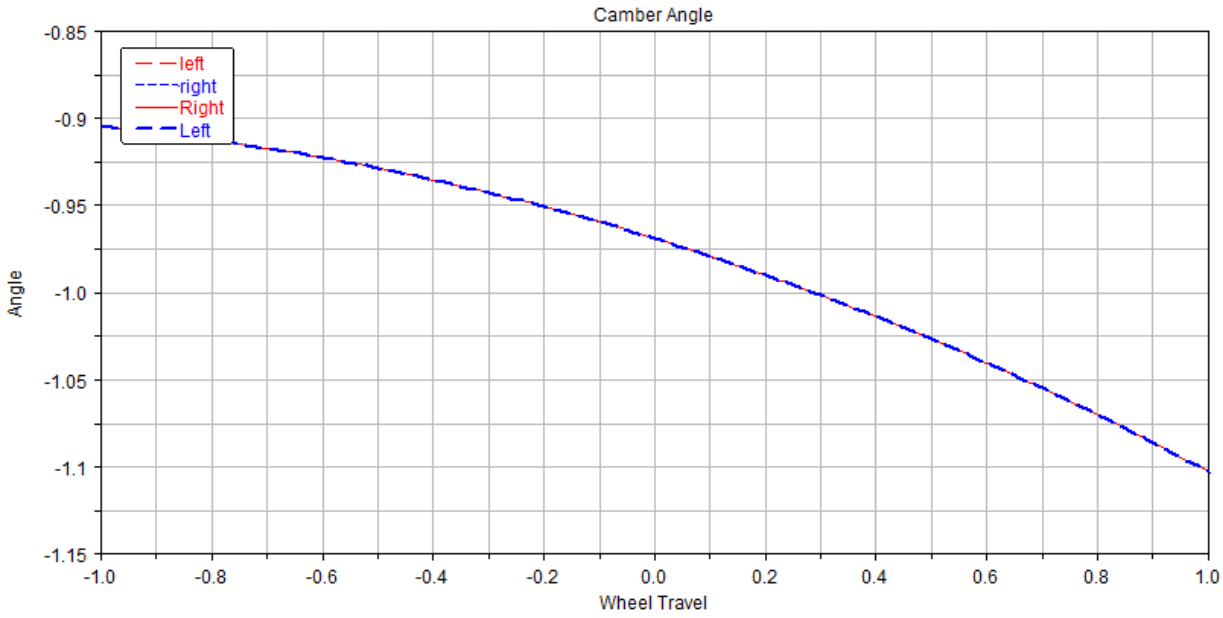


Figure 4.2.11 – Camber Angle vs. Wheel Travel Plot

4.2.5 Power Generation Test Plan

The test plans for power generation system consists of testing the solar array system, MPPT, and the regenerative braking. The solar array system will be tested for proper configuration (series and\or parallel); it will also be tested for manufacture rated open-circuit voltage and short-circuit current. The efficiency of the solar cell and insolation level of Tallahassee will be taken into account when measure these parameters. The MPPT is basically a DC: DC converter; so it will be tested for input and output voltages. The current coming from the MPPT into the battery shall be tested and measured using the state of charge device. The regenerative braking system will be tested to ensure the brake is applied when the system is asserted; it will also be tested to ensure if charge is being supplied to the battery system when the regenerative brake is asserted. Most of the testing will be performed using a digital multi-meter; care should be taken to ensure proper settings in the multi-meter before measurement.

4.2.6 Control Systems

Some of the major components were tested in the previous phase but will need to be retested upon integration into the new system. The MCU will need to be thoroughly tested in order to verify that all of the MCU software is functioning properly. Most of the MCU tests will be performed using simulations.

The Dashboard system components will need to be tested prior to integration. The speedometer and state of charge meter can be tested for functionality, but further tests will need to be performed after electrical system integration. The discrete electrical components will be tested using switches.

The serial functionality of the motor controller and the MCU will be tested using a serial terminal program on a PC before integration. After the MCU and motor controller functionality have been verified, a simple test program will need to be implemented to test the integration functionality.

4.2.7 Management System

The new features of the management system have been tested thoroughly during the integration phase as well as tests performed on the old components from the previous phase. These tests were necessary to verify that no damage occurred to the system during the summer and also to verify the systems were tested correctly in the first place. For this reason testing of the management system began with the batteries and propulsion subsystems.

The batteries when first examined were tested for charge. The anticipated charge for a battery was 3.3 V and each of the batteries was tested using a volt meter. Out of the thirty batteries only one of the batteries was below this value. To correct this problem that battery was hooked up to an iCharger that allows the charge of an individual battery instead of charging all the batteries in series. The battery was again tested after using the iCharger and was verified to be able to hold a charge. The other important subsystem that was tested involving the batteries was the wall charging system. This subsystem was tested as a whole due to the thoroughly with which the first phase of the project tested this subsystem. The batteries were able to be charged successfully using the wall charging unit.

The propulsion system was rather simple to test. After the batteries were tested and reinstalled in the car the propulsion test began. It consisted of getting the car started and hitting the accelerator. This however did encompass a preliminary test of the existing dashboard systems as well. The car did run properly on the battery power once it was turned on responding to all the existing controls, which includes the steering and braking systems. Upon completion of a few preliminary tests it was noted that the motor controller, which should send signals up to 5V, seemed unable to exceed

about 2.4V. Further testing revealed that the source of this error was due to a calibration issue, something that most likely was an issue during the previous phase as well. This problem was corrected and now the full power of the motor was available to the driver when accelerating.

The subsystems that will be added during this phase can be tested more rigorously during the different stages of development. The state of charge system is comprised of several smaller components and each of these components have been tested before integration. This can be done on a small scale and then again on a larger level once integrated into the complete electrical system.

4.3 Summary of Test Plan

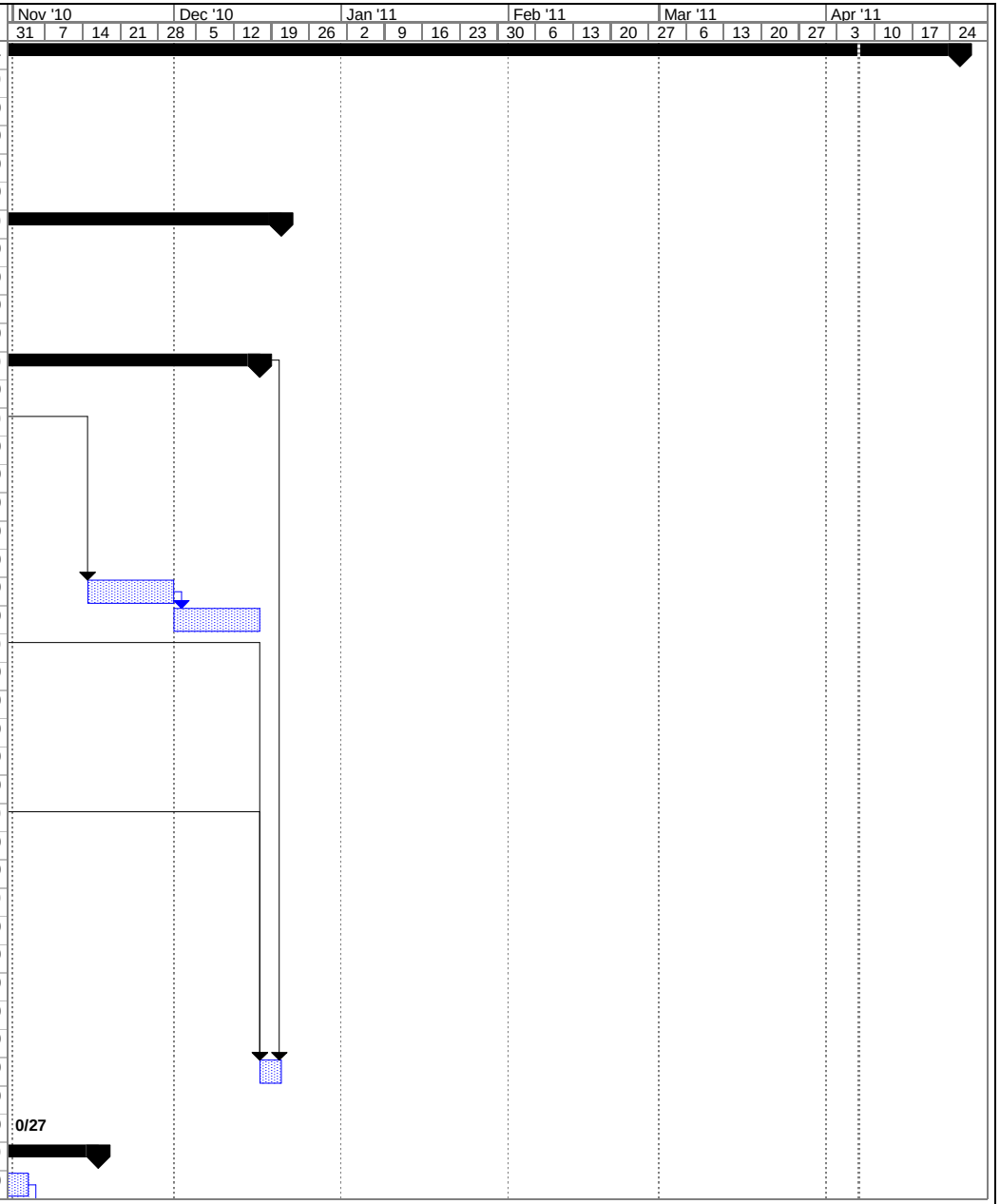
The following table shows a summary of the tests to be performed on the different components and systems of the car.

Test	Test Case #	Result
Carbon fiber tensile test	BD-001	PASS
Rack and pinion steering test	SS-001	PASS
Brake rotor	BS-001	PASS
Master cylinder	BS-002	PASS
Brake line	BS-003	PASS
Brake Caliper	BS-004	PASS
Upper Control Arm Structural Testing	SP-001	PASS
Lower Control Arm Structural Testing	SP-002	PASS
Upright Structural Testing	SP-003	PASS
Front Suspension Simulation	SP-004	PASS
Rear Suspension	SP-005	PASS
Test regenerative braking signal when the regenerative braking handle is asserted	PGS-001	PASS
Test regenerative braking system charges the battery	PGS-002	PASS
MCU power test	CS-001	PASS
SW-192 Relays Test	CS-002	PASS
12V to 9V DC-DC Converter Test	CS-003	PASS
Fuse Box Test	CS-004	PASS
Potentiometers Test	CS-005	PASS
Motor Controller Power Test	CS-006	PASS
BMS Power Test	CS-007	FAIL
MCU/12V to 9V DC-DC Integration Test	CS-008	PASS
MCU/Relay/Fuse Box Integration Test	CS-009	PASS
Breakout Board Test	CS-010	PASS
Speedometer Test	CS-011	PASS
100V to 12V/Fuse Box Integration Test	CS-012	PASS
Full Control System Integration Test	CS-013	PASS
Dashboard Control Integration Test	CS-014	PASS
Battery Charger Integration Test	CS-015	FAIL
State of Charge Meter Power Test	MS-001	PASS
State of Charge Meter Shunt Current Test	MS-002	PASS

5 Schedule

Below the initial schedule for the project can be seen, including the anticipated deadlines for the project. Further below that the actual schedule for the project is included.

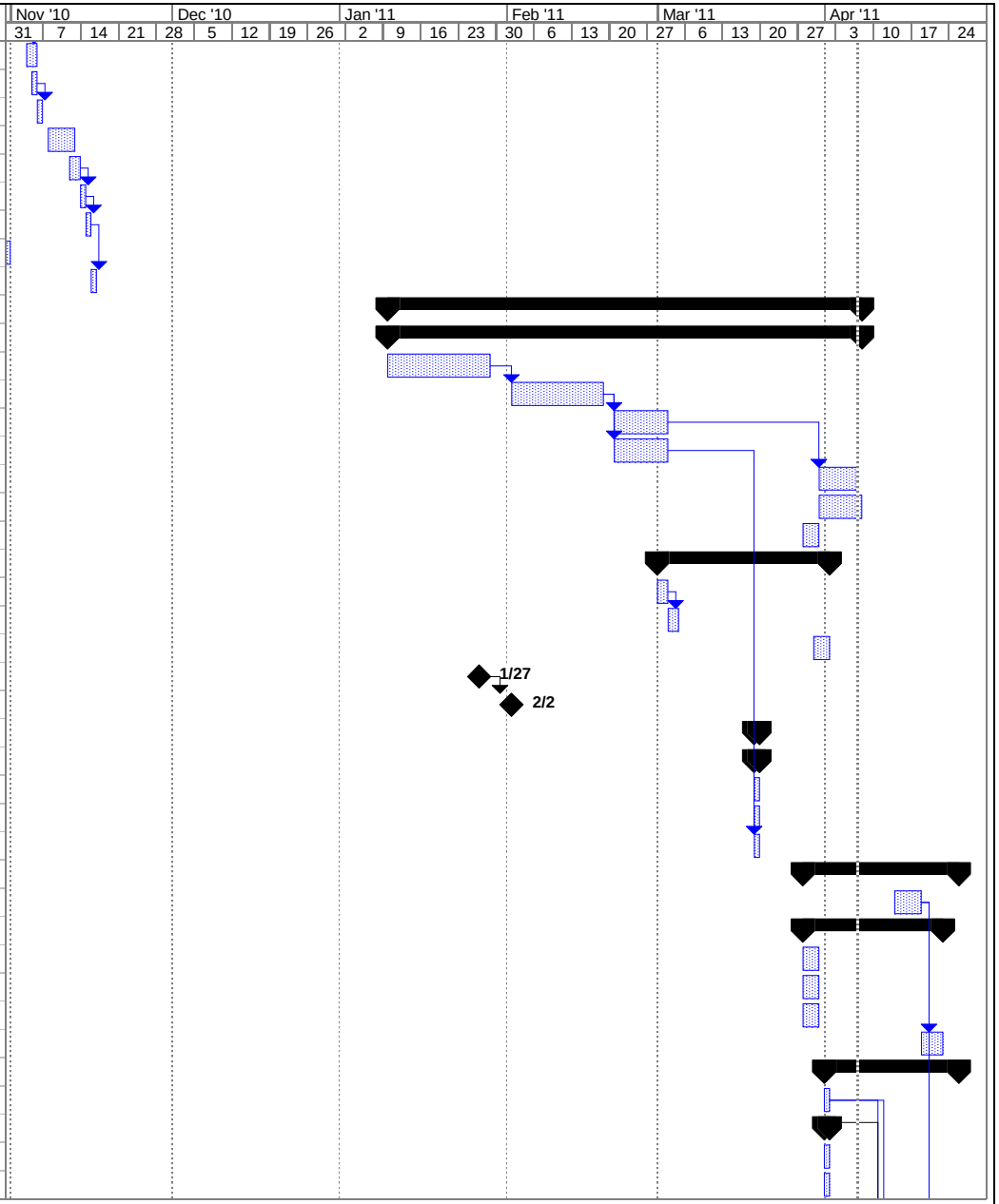
ID	Task Name	Duration	Start	Finish
1	**Solar Car**	**174 days**	**Mon 9/6/10**	**Mon 4/25/11**
2	**Needs Assesment Presentation**	**18 days**	**Mon 9/6/10**	**Wed 9/29/10**
3	Research race constraints	3 days	Mon 9/6/10	Wed 9/8/10
4	Research race requirements	5 days	Wed 9/8/10	Tue 9/14/10
5	Develop system test plans	4 days	Wed 9/15/10	Mon 9/20/10
6	Presentaion	1 day	Wed 9/29/10	Wed 9/29/10
7	**Solar Car Project Proposal**	**62 days**	**Fri 10/1/10**	**Mon 12/20/10**
8	Assigned Project Tasks/ Roles	1 day	Fri 10/1/10	Fri 10/1/10
9	Research Solar Arrays	5 days	Mon 10/4/10	Fri 10/8/10
10	Research Power Trackers	5 days	Mon 10/4/10	Fri 10/8/10
11	Research Battery Protection Systems	5 days	Mon 10/4/10	Fri 10/8/10
12	**Research Suspension**	**58 days**	**Mon 10/4/10**	**Thu 12/16/10**
13	Research Current Technology	5 days	Mon 10/4/10	Fri 10/8/10
14	**Select Components**	**7 days**	**Mon 10/11/10**	**Mon 10/18/10**
15	Hub	6 days	Mon 10/11/10	Sat 10/16/10
16	Ball Joints	6 days	Mon 10/11/10	Sat 10/16/10
17	Loswer wishbone	6 days	Mon 10/11/10	Sat 10/16/10
18	Upper wishbone	7 days	Mon 10/11/10	Mon 10/18/10
19	Shocks/Damper	1 day	Mon 10/11/10	Mon 10/11/10
20	Model Suspension in Solid Works	12 days	Mon 11/15/10	Tue 11/30/10
21	FEM analysis of Component	12 days	Wed 12/1/10	Thu 12/16/10
22	**Research Brake Systems**	**6 days**	**Mon 10/4/10**	**Mon 10/11/10**
23	Research Braking Components	6 days	Mon 10/4/10	Mon 10/11/10
24	Research Composite Materials	5 days	Mon 10/4/10	Fri 10/8/10
25	Research Aerodynamic Design	5 days	Mon 10/4/10	Fri 10/8/10
26	Research Sensors	3 days	Mon 10/11/10	Wed 10/13/10
27	Regesearch Regenitive Brakes	3 days	Mon 10/11/10	Wed 10/13/10
28	**Research Steering System**	**1 day**	**Tue 10/12/10**	**Tue 10/12/10**
29	research rack and Pinion Steering	1 day	Tue 10/12/10	Tue 10/12/10
30	Calculate appropriate Steering Ratio	1 day	Tue 10/12/10	Tue 10/12/10
31	**Select Components**	**6 days**	**Mon 10/4/10**	**Mon 10/11/10**
32	steering wheel	6 days	Mon 10/4/10	Mon 10/11/10
33	steering column	6 days	Mon 10/4/10	Mon 10/11/10
34	steering shaft	6 days	Mon 10/4/10	Mon 10/11/10
35	rack and pinion	6 days	Mon 10/4/10	Mon 10/11/10
36	steering stops	6 days	Mon 10/4/10	Mon 10/11/10
37	Develop Design Proposal	2 days	Fri 12/17/10	Mon 12/20/10
38	Project Proposal Report	0 days	Mon 10/18/10	Mon 10/18/10
39	Presentation	0 days	Wed 10/27/10	Wed 10/27/10
40	**System Level Design Review**	**17 days**	**Wed 10/27/10**	**Tue 11/16/10**
41	Develop Top Level Design	6 days	Wed 10/27/10	Wed 11/3/10



Project: Solar Car Project revised.mpp
Date: Thu 4/7/11

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

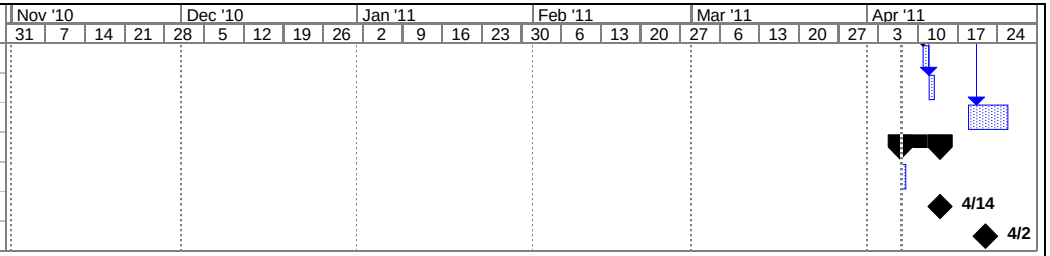
ID	Task Name	Duration	Start	Finish
42	Analyze Top Level Desgin	2 days	Thu 11/4/10	Fri 11/5/10
43	Modify Top Level Design	1 day	Fri 11/5/10	Fri 11/5/10
44	Finalize Top Level Design	1 day	Sat 11/6/10	Sat 11/6/10
45	Develop Modular Level Design	5 days	Mon 11/8/10	Fri 11/12/10
46	Analyze Modular Lever Design	1 day	Fri 11/12/10	Sat 11/13/10
47	Modify Modular Level Design	1 day	Sun 11/14/10	Sun 11/14/10
48	Finalize Modular level Design	1 day	Mon 11/15/10	Mon 11/15/10
49	Research Vendors	3 days	Wed 10/27/10	Sun 10/31/10
50	Purchase parts	1 day	Tue 11/16/10	Tue 11/16/10
51	**Detailed Design Review and Test Plan**	**65 days**	**Mon 1/10/11**	**Thu 4/7/11**
52	**Lower Body Assembly**	**65 days**	**Mon 1/10/11**	**Thu 4/7/11**
53	Construct Lower Shell Mold	15 days	Mon 1/10/11	Fri 1/28/11
54	Construct Lower Body Shell	13 days	Wed 2/2/11	Fri 2/18/11
55	Assemble Front Suspension	8 days	Mon 2/21/11	Wed 3/2/11
56	Assemble Rear Suspension	8 days	Mon 2/21/11	Wed 3/2/11
57	Install Disc Brakes	5 days	Thu 3/31/11	Wed 4/6/11
58	Install Regenerative Brake	6 days	Thu 3/31/11	Thu 4/7/11
59	Install Steering System	3 days	Mon 3/28/11	Wed 3/30/11
60	**Upper Body Assembly**	**25 days**	**Tue 3/1/11**	**Fri 4/1/11**
61	Construct Upper Body Mold	2 days	Tue 3/1/11	Wed 3/2/11
62	Construct Upper Composite Body	2 days	Thu 3/3/11	Fri 3/4/11
63	Assemble Upper Shell to Lower Body	3 days	Wed 3/30/11	Fri 4/1/11
64	Detailed Design Review and Test Plan Report	0 days	Thu 1/27/11	Thu 1/27/11
65	Detailed design Review and Test Plan Presenta	0 days	Wed 2/2/11	Wed 2/2/11
66	**Lower Body Configuration**	**1 day**	**Sat 3/19/11**	**Sat 3/19/11**
67	**Re-install working components of managem**	**1 day**	**Sat 3/19/11**	**Sat 3/19/11**
68	Batteries	1 day	Sat 3/19/11	Sat 3/19/11
69	Charging System	1 day	Sat 3/19/11	Sat 3/19/11
70	Motor	1 day	Sat 3/19/11	Sat 3/19/11
71	**Solar Array Installation**	**22 days**	**Mon 3/28/11**	**Mon 4/25/11**
72	Test Solar Cells	4 days	Thu 4/14/11	Mon 4/18/11
73	**Total Body Configuration**	**21 days**	**Mon 3/28/11**	**Fri 4/22/11**
74	Install Energy Storage System	3 days	Mon 3/28/11	Wed 3/30/11
75	Install Central Interface System	3 days	Mon 3/28/11	Wed 3/30/11
76	Install Propulsion System	3 days	Mon 3/28/11	Wed 3/30/11
77	Install Energy Generation System	4 days	Tue 4/19/11	Fri 4/22/11
78	**Final Testing Phase**	**18 days**	**Fri 4/1/11**	**Mon 4/25/11**
79	Test Steering System	1 day	Fri 4/1/11	Fri 4/1/11
80	**Test Braking System**	**1 day**	**Fri 4/1/11**	**Fri 4/1/11**
81	Regenerative Brake	1 day	Fri 4/1/11	Fri 4/1/11
82	Standard Brakes	1 day	Fri 4/1/11	Fri 4/1/11



Project: Solar Car Project revised.mpp
Date: Thu 4/7/11

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

ID	Task Name	Duration	Start	Finish
83	Test Car Maneuver test	1 day	Mon 4/11/11	Mon 4/11/11
84	Car Stability Test	1 day	Tue 4/12/11	Tue 4/12/11
85	Test Solar Capabilites	5 days	Tue 4/19/11	Mon 4/25/11
86	**Design Fair**	**5 days**	**Thu 4/7/11**	**Thu 4/14/11**
87	Final Report	1 day	Thu 4/7/11	Thu 4/7/11
88	Final Project Oral Presentations	0 days	Thu 4/14/11	Thu 4/14/11
89	Peer Evaluations and Course Review	0 days	Fri 4/22/11	Fri 4/22/11



Project: Solar Car Project revised.mpp
Date: Thu 4/7/11

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

6 Budget Estimate

6.1 Personnel Expenses

Name	Hours	Base Pay	Total
Barge, James	384	$30.00	$11,520.00
Cires, Adrian	384	$30.00	$11,520.00
Dalick, Keith	384	$30.00	$11,520.00
German, Nelson	384	$30.00	$11,520.00
Panther, Emiliano	384	$30.00	$11,520.00
Pradhan, Rajat	384	$30.00	$11,520.00
Prisland, Zachary	384	$30.00	$11,520.00
Rajbhandari, Shishir	384	$30.00	$11,520.00
Roberts, Amanda	384	$30.00	$11,520.00
Subtotal			$103,680.00
Fringe Benefit (29%)			$30,067.20
Total Personnel Cost			**$133,747.20**

6.2 Expenses

ELECTRICAL					
Item	**Quant.**	**Unit Cost**	**Total**	**Reference**	**Reference**
12V:9V DC:DC	1	$40.00	$40.00	website	powerstream
Relay	3	$120.25	$360.75	website	tecknowledgey
Micro controller	1	$188.00	$188.00	website	evb plus
Wires	1 set	$15.00	$15.00	local	home depot
Electric Tape	2	$5.00	$10.00	Local	home Depot
Serial cable	1	$20.00	$20.00	local	radio shack
State Of Charge	1	$424.00	$424.00	website	evolve electrics
Solar Cell	68	$72.50	$4930.00	phone	Flecs Solar
Fuses	1	$17.32	$17.32	local	Four Acres
Breakers	2	$100.00	$100.00	website	mcmaster carr
Switches	1	$42.71	$42.71	local	Four Acres
Connectors	1	$9.99	$9.99	local	Four Acres
110 A Fuse	1	$18.80	$18.80	website	nort. Ariz. W&S
4 Gauge Wire	1 set	$72.50	$72.50	website	nort. Ariz. W&S
Battery Connectors	2	$16.00	$32.00	website	nort. Ariz. W&S
AC Plug	1 set	$130.00	$130.00	website	plumberSurplus
Quick Disconnects	1 set	$86.00	$86.00	local	four acres
Subtotal			$6,497.07		

MECHANICAL					
Item	**Quant.**	**Unit Cost**	**Total**	**Reference**	**Vendor**
Brake Kit	2	$125.00	$250.00	website	wilwood

Wood/Boards	1	$330.00	$330.00	local	home depot
Foam	10	$50.00	$500.00	phone	minco auto
Lumber	1	$15.21	$15.21	local	machine shop
Wax	2	$10.00	$20.00	local	home depot
C-Fiber 12k	197 yds	$20.00	$3,940.00	phone	hexel
C-Fiber 3k	50 yds	$16.80	$842.25	local	machine shop
Resin	20gallons	$25.00	$500.00	local	composites one
Lantor Soric	1	$439.00	$439.00	local	machine shop
Fiberglass	1	$200.00	$200.00	website	mcmaster-carr
Solidworks	9	$150.00	$1,350.00	phone	solidworks
Suspension kit	1	$17.13	$17.13	local	advanced auto
Plexi Glass	1	$54.02	$54.02	local	home depot
70mm Screws	1	$15.55	$15.55	Website	mc master Carr
Brake Line	1	$90.21	$90.21	local	advanced auto
Shocks	2	$69.90	$139.80	Website	icycles
Subtotal			$ 8,703.17		

INDUSTRIAL					
Item	**Quantity**	**Unit Cost**	**Total**	**Reference**	**Vendor**
Driver seat	1	$395.00	$395.00	website	corbeau
Vinyl	1	$250.00	$250.00	signs unlimtd.	local
Filter (top shelf)	1	$17.98	$17.98	local	home depot
Painting Supply	1	$28.19	$28.19	local	lowes
Seat padding	1	$30.00	$30.00	local	Joan fabric
5-Point Seat Belt	1	$75.00	$75.00	Website	Corbeau
Subtotal			$ 796.17		
Total Expenses			**$15,996.41**		

6.3 Overhead

Overhead Cost	
PERSONNEL	$133,747.00
EXPENSES	$15,996.41
DIRECT COST	$149,743.41
Total at 45%	**$67,384.53**

6.4 Total Budget

TOTAL BUDGET	
PERSONNEL	$133,747.20
EXPENSES	$15,996.41
OVERHEAD	$67,384.53
Total Project Cost	**$ 217,128.14**

6.5 Final Balance Sheet

FINAL BALANCE SHEET	
EXPENSES	$15,996.41
IESES	($3,970.00)
FAMU-FSU	($5,000.00)
DONATIONS	($7,586.46)
BALANCE REMAINING	**$ 560.05**

The final budget for our project is displayed above. Donations in the form of Student Licenses were acquired from SolidWorks and MSC ADAMS at the beginning of the semester. This helped us in designing and analyzing the body and suspension system of the solar car. Carbon fiber for the project was donated from Hexel. The machine shop from the College of Engineering also donated carbon fiber and other supplemental materials. The foam was donated from Minco Auto, and the resin for the carbon fiber was donated from Composites One. IESES was our biggest donator; their sponsorship enabled us to purchase enough solar cells to cover the car; special thanks to Melanie Simmons for funding the project. These donations were very helpful in keeping the team on schedule with the design and construction of the project. All these companies and the machine shop are greatly appreciated for their sponsorship. Special thanks and appreciation to for Mr. Jerry Horne from HPMI in leading and guiding us in carbon fiber fabrication of the top and bottom shell of the solar car.

7 Conclusion

When the project began the anticipated goal was a race ready vehicle capable of competing in the American Solar Challenge race in 2012. It quickly became apparent that without generous donations the solar car race would be impossible. With this in mind the group began seeking funds, both fiscal and material, to continue the push towards the lofty goal. The team as a whole was able to secure donations totaling more than $16,500 which greatly impacted the overall success of the project.

Phase I stripped down the 2001 solar car, replaced many of the electrical systems, and attempted to fix suspension issues. During Phase II of the project the team was capable to do a complete overhaul of all systems. As explained above the steal framed body was completely replaced with a carbon fiber body which will be the mounting point for all other components in the system. The total weight of the top shell is 50 lbs and the bottom shell weighed 100 lbs before all the cuts were made for mounting various components. The estimated weight of the fiberglass shell was over 300 lbs and a steel frame was still included! The aerodynamic design of the body will allow for continued use in the future and provide the stable structure needed to compete in any race environment.

Both the front and the rear suspension were removed and replaced with a system that would fit more appropriately into the newly designed and fabricated body. The previous phase created a rear suspension that was only supported a single side of the tire, which created a large camber angle. During the second phase the team was able to create a doubly supported trailing arm suspension that corrected this issue. While selecting the appropriate spring for the suspension during the last phase, a member of the team forgot to account for two tires, resulting in springs which would be considered overkill for this application. When performing the redesign of the suspension all of these issues were fixed leaving the suspension in a fully completing and race-worthy state.

Since the overall weight of the vehicle changed with the fabrication of the carbon fiber body, the braking could also be modified for the new weight constraints of the vehicle. The overall weight of the vehicle with all components inside is about 507 lbs, with small error for inaccuracy in the scales. This weight is more comparable to a go-kart than a full size car and therefore the choice for brakes were specified for the given weight. Keeping the race as an ultimate goal the brake calipers were chosen to meet the minimum requirements for the race as well. The other issue with the braking system was an implemented hand brake for the regenerative braking. One of the main goals for this project was to find a means to integrate this with the mechanical brake, essentially simplifying the work for the driver, hit one pedal activate both. Through the use of creative fabrication the team produced a pedal cluster that accurately engages both braking system while pushing the same pedal. The completed braking system is something that the team can be proud of, something that should give a competitive edge to the solar car.

The power generation system did fall a little short in the overall progress department, largely because of the budget. Since the team began with a working car and knowing that overall success would be compared to the previous phase, it was paramount that the budget first goes towards the systems that would need to be improved to keep the car running. This unfortunately starved the power generation system, which needed to purchase the two most costly pieces of equipment, the MPPT and solar cells. As seen in the report above some of the analysis was performed for the design of the MPPT when it became apparent that the budget would not allow for the purchase of such a device. There was also heavy analysis and research performed into selecting the most appropriate solar cells for this application and preliminary purchases were made for this endeavor. The system has been left as expandable as possible, such as a relay designed to isolate the solar cells and MPPT from the rest of the electrical components in the event of dangerous operating conditions. This should facilitate future efforts to finally integrate the purchased solar cells, a designed MPPT, and additional solar cells if necessary to become a true solar car.

The remainder of the electrical system was improved from the previous phase by implementing a microcontroller and development board. While this phase of the project had limited uses of the board, it will help drastically when a team is attempting to improve the overall performance of the car since it will be capable of recording telemetry information on the vehicle. When implementing the board with a normally open relay, it was able to provide an extra level of protection for the motor controller (a $3000 piece of equipment), by keeping the relay open until the pre-charge circuit was completed. The other changes to the electrical system were the introduction of many, many fuses to help protect each piece of equipment along the way. For example, the DC/DC (100 V to 12 V) can handle a max load of 200 W, inline fuses were added both before and after the converter to ensure that the power would never exceed the maximum (2 A on the primary side and 20 A on the secondary side). The previous phase had 5 fuses integrated into their system and during this phase 10 more were added to ensure that no component is ever exposed to dangerous levels. The improvement which carries no numerical value, but is possibly one of the most useful is that every single connection to every single piece of equipment has quick connects on the ends. This means that every single device and every wire in the system can be removed without cutting a single wire. Since the electrical team found the old system from the previous phase rather unexpandable, one of the primary personal goals for this team was to guarantee that whoever took over the car after this phase could continue the expansion process without rebuilding a whole system.

Despite the fact that original scope of the project had to change many times overall as more information was gathered, the project as a whole was a success. The mechanical systems are completely sound and race worthy from this point forward. The electrical system in place works very well and is really only lacking the MPPT and solar panels to be fully considered as a solar car. The overall completion of this car has forced every individual on this team to adapt too many new areas of study including materials, renewable energy, or aerodynamics and as a whole every engineer on the team has a much more comprehensive repertoire of skills to bring along for future endeavors.

8 Bibliography

Änderung, L. (2009, May 15). *Hochschule Bochum University of Applied Sciences*. Retrieved November 10, 2010, from http://www.hochschule-bochum.de/en/solarcar.html

Barrys Tyre & Exhaust Centre. (2010). *Wheel Alignment.* Retrieved October 29, 2010, from Barrys Tyre & Exhaust Centre: http://www.barrystyre.co.uk/80610/info.php?p=5

Cady, F. M. (2008). *Software and Hardware Engineering: Assembly and C Programming for the Freescale HCS12 Microcontroller* (2nd ed.). New York: Oxford University Press.

CR Magnetics, Inc. (n.d.). *CR Magnetics: Products*. Retrieved October 29, 2010, from CR Magnetics: http://www.crmagnetics.com/products/CR8750-P96.aspx

Cyber, M. (1999, June 29). *Sunrayce 99*. Retrieved November 10, 2010, from http://www.lasersol.com/air_water/sunrayce_99/Sunrayce.html

Endless-Sphere. (n.d.). *Forums: Endless-Sphere*. Retrieved November 14, 2010, from Endless-Sphere Website: http://endless-sphere.com/forums/viewtopic.php?f=14&t=13839&start=0

Evolve Electrics: TBS Electronics E-Xpert Pro. (n.d.). Retrieved January 2010, from Evolve Electrics: http://evolveelectrics.com/E-Xpert%20Pro.html

Kruschandl, N. (2005). *Solar Car Anatomy* . Retrieved November 10, 2010, from http://www.speedace.info/solar_car_anatomy.htm

Kularatna, N. (1998). *Power Electronics Design Handbook: Low-Power Components and Applications.* Boston: Newnes.

Longhurst, C. (2010, October 11). *The Suspension Bible*. Retrieved October 15, 2010, from http://www.carbibles.com/suspension_bible.html

Penmethsa, H. V. (2004, June 4). *FEA on Vehicle Suspension System.* Retrieved February 2, 2011, from http://penmethsa.blogspot.com/2009/06/blog-post.html

Renewable Energy Access. (2006, December 7). *Renewable Energy World News Articles*. Retrieved April 5th, 2011, from Renewable Energy World: http://www.renewableenergyworld.com/rea/news/article/2006/12/solar-cell-breaks-the-40-efficiency-barrier-46765

Shiota, L. (2010, Spetember 26). *Car Suspension Types.* Retrieved November 13, 2010, from eHow: http://www.ehow.com/list_7233942_car-suspension-types.html

Tecknowledgey. (2002). *Tecknowledgey Products*. Retrieved April 5, 2011, from Tecknowledgey.com: http://www.tecknowledgey.com/catalog/product_info.php?cPath=57_60&products_id=528&osCsid=034a06df926b782db1a0e05

Temple, R. W. (1969, Spetember). *Popular Mechanics.* Retrieved November 13, 2010, from Google Books: http://books.google.co.uk/books?id=M9gDAAAAMBAJ&lpg=PP1&pg=PA129#v=onepage&q&f=false

Tesla Motors. (n.d.). *Telsa Motor Efficiency*. Retrieved April 5, 2011, from Tesla Motors Website: http://www.teslamotors.com/goelectric/efficiency

Wan, M. (2000). *Suspension Geometry.* Retrieved November 12, 2010, from AutoZine Technical School: http://www.autozine.org/technical_school/suspension/tech_suspension1.htm

9 Appendix

9.1 User Manual

Disclaimer: It is very important that the user of the vehicle not make contact with any but the mentioned electrical components. While the electrical components are all isolated from the user and covered by various insulations, it is still possible to come into contact with potential lethal amounts of current. The designer and builders of this car have taken great measures to ensure this will not happen, but nothing is fool proof. This being said, no changes should be made to the system without training in electrical safety and a thorough understanding of the overall system.

The following include the steps that should be undertaken to begin driving the vehicle. Only one person may occupy the vehicle at any given time, hence the reason there is only one seat, however it should be noted that at least three individuals will be required to ensure safest operation of the vehicle.

1. Place blocks or stops in front and behind one of the wheels to ensure the vehicle does not move while interacting with the bottom shell.
2. Remove the top carbon fiber shell from the vehicle and place to the side for now.
3. Ensure that the main contacter is in the off position before touching any of the components in the vehicle.
4. Once the contactor is off, verify that all wires appear to be securely connected and there are no extraneous or unplugged wires floating around in the bottom shell.
5. Ensure that the vehicle is free of all debris that could potentially get caught in moving parts or short circuit electrical elements.
6. One individual may take a seat in the vehicle and should buckle themselves into 5-point seat belt. The seat and seat belt can be seen in Figure 9.1.1 below.

Figure 9.1.1 – The driver seat and seat belt

7. Place the large three position switch on the driver's ride hand side to 'N' or neutral position. As seen below in Figure 9.1.2.

Figure 9.1.2 – Three switches used by the driver to start the car

8. Once the driver is securely fastened into the seat one of the other two individuals may flip the main contactor switch, at this point the power is active in the system and nothing else should be touched without specific direction.
9. The driver should now flip the green switch, labeled '1', on their right to the ON position.
10. The driver should see the display on the state of charge device immediately to their right turn on, if the display does not come on then the main contactor should immediately be flipped to the off position and the driver exit the vehicle. An engineer should be consulted to determine the issue before attempting to operate the vehicle again. The state of charge display can be viewed in Figure 9.1.3.

Figure 9.1.3 – The state of charge display for the driver

11. If the state of charge meter has turned on correctly then the black switch, labeled '2', should be turned to the ON position.
12. At this point the motor is charging, which will take approximately 10 seconds. The driver will know when the pre charge is completed because of the clicking sound from the relay activating.
13. The two other individuals should then carefully place the top shell onto the bottom shell, ensuring that the top shell is completely encased by the metal strip along the sides of the bottom shell.

14. Verify that the top shell is placed securely on, the driver is buckled into the seat belt properly, and the driver has protective eye equipment to keep out dust particles.
15. The stops can now be removed from around the tire.
16. Now the driver may use the large three position switch on their right, labeled ‘F N R’, which will place the motor into forward, neutral, or reverse. The driver should select the desired direction of motion.
17. The driver should now use the throttle, braking, and steering to navigate to any desired location. The operation of the vehicle is similar to any commercial car and should be driven using the same road safety protocols.
18. Once the driver is finished driving around, the other two individuals should again place the stops around one of the tires.
19. The driver should place the large three position switch on their right, labeled ‘F N R’, to the ‘N’ position.
20. The driver should turn the other two switches, labeled ‘1’ and ‘2’, to the off position.
21. The other two individuals may remove the top shell and again place it to the side.
22. The main contactor should be placed in the OFF position.
23. The driver may now remove the seat belt and exit the car.
24. The top shell may again be placed on top of the vehicle, ensuring that the top shell is once again inside the metal strip on the bottom shell.

9.2 Complete Test Reports

Test Plan – Solar Car Team '11

TEST ITEM (TITLE): 12K Carbon Fiber, 3K Carbon Fiber

TEST CASE #: BD-001 (ex: BS-001)

TEST DATE/TIME: 02/04/2011 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: TEST ■ RE-TEST ☐

The objective of this test is to determine the ultimate tensile strength of the carbon fiber we will be using in the vehicle. The 12K carbon fiber will be used on the bottom half of the car for higher strength. The 3K carbon fiber will be used on the top because of the light weight properties. A sample strip of each will be cut and used in a tensile testing machine.

EXPECTED RESULTS:

The carbon fiber composite will exceed the minimum force requirements.

ACTUAL RESULTS:

The tensile test showed confirmed that the carbon fiber composite exceeded the minimum force requirements

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team '11

TEST ITEM (TITLE): Rack and Pinion Steering System

TEST CASE #: SS-001 (ex: BS-001)

TEST DATE/TIME: 3/14/11 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: TEST ■ RE-TEST ☐

The rack and pinion will need to be tested to see if it can provide accurate steering for the new body design. The rack and pinion should not lock up and it must provide a smooth transition from rotational to linear motion as the steering wheel is turned. The rack and pinion must also be able to push the tie rods so the effectively turn the wheels.

EXPECTED RESULTS:

The rack and Pinion will function properly and give the vehicle proper steering

ACTUAL RESULTS:

The rack and pinion gears meshed with no problems when engaged. The gear effectively turned the cars wheels

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team '11

TEST ITEM (TITLE): Brake Rotor

TEST CASE #: BS-001 (ex: BS-001)

TEST DATE/TIME: 3/31/11 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ☒ TEST ☐ RE-TEST

A custom brake rotor will be fabricated in order to better suit the light weight of the vehicle. As the brake pads apply a frictional force to the brake rotor, the energy is transformed into heat. The rotor will need to be tested to ensure it can effectively dissipate heat, to reduce the chance of rotor failure. This can be done by performing FEM analysis on the rotor in order to see its heat transfer capabilities.

EXPECTED RESULTS:

The brake rotor will perform exceptionally.

ACTUAL RESULTS:

The rotor effectively dissipated heat that was generated due to friction, and was able to bring the car to a complete stop.

STATUS: ☒ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team '11

TEST ITEM (TITLE): Master Cylinder

TEST CASE #: BS-002 (ex: BS-001)

TEST DATE/TIME: 3/31/11 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ☒ TEST ☐ RE-TEST

The purpose of the master cylinders is to translate the force from the brake pedal into hydraulic fluid pressure to push the pistons in the caliper assembly. Test must be performed to meet race regulations.

EXPECTED RESULTS:

The master Cylinder will be able to provide sufficient hydraulic pressure to the calipers.

ACTUAL RESULTS:

The master cylinder provided enough pressure to allow the calipers to clamp on the rotor effectively to lock the wheels up when needed

STATUS: ☒ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team '11

TEST ITEM (TITLE): Brake Lines

TEST CASE #: BS-003 (ex: BS-001)

TEST DATE/TIME: 3/31/11 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The brake lines transport the brake fluid from the master cylinder to the caliper assembly. The lines must be checked to ensure no leaks are present in the lines, which will decrease hydraulic pressure to the calipers

EXPECTED RESULTS:

The brake lines will have no leaks.

ACTUAL RESULTS:

Brake lines had no leaks

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team '11

TEST ITEM (TITLE): Brake Caliper Assembly

TEST CASE #: BS-004 (ex: BS-001)

TEST DATE/TIME: 4/1/11 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ☒ TEST ☐ RE-TEST

The brake caliper will apply a clamping force onto the brake rotor to generate friction force onto the rotor. The caliper contains a piston which pushes the brake pads onto the rotor. The pistons must be checked to ensure the piston does not lock up, which will cause frictional force to be constantly applied to the rotor. This may lead to rotor failure

EXPECTED RESULTS:

Caliper piston is able to apply pressure to the rotor and return freely to its original position.

ACTUAL RESULTS:

Caliper provides sufficient pressure on brake rotor to slow and stop the vehicle

STATUS: ☒ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team '11

TEST ITEM (TITLE): Upper Control Arm Structural Testing

TEST CASE #: SP-001 (ex: BS-001)

TEST DATE/TIME: 2/28/2011 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The objective of this test is to verify that the part can structurally hold under loading.
This part will be tested in SolidWorks using the Finite Element Method analysis built in the program.
The test will provide displacement and Von Misses stress analysis.

EXPECTED RESULTS:

The upper control arm will not have any deformation during the force analysis.

ACTUAL RESULTS:

The max von Misses was 3.9 psi, which is less than the yield strength of 10998.1 psi. Thus, the upper control arm won't yield.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Name	Type	Min	Location	Max	Location
Stress1	VON: von Mises Stress	0.00368022 psi Node: 8136	(1.60983 in, -0.4 in, 6.375 in)	3.88641 psi Node: 40	(2.75 in, 6.46114e-017 in, -5.75 in)
Displacement1	URES: Resultant Displacement	0 mm Node: 1	(7.25 in, 6.46114e-017 in, -0.25 in)	5.12564e-005 mm Node: 5772	(5.83558 in, -2.02047e-006 in, 3.66442 in)
Strain1	ESTRN: Equivalent Strain	5.88898e-010 Element: 2974	(1.58795 in, 0.359873 in, 6.39748 in)	3.63689e-007 Element: 3580	(2.78777 in, 0.252652 in, 5.94279 in)

Test Plan – Solar Car Team '11

TEST ITEM (TITLE): Lower Control Arm Structural Testing

TEST CASE #: SP-002 (ex: BS-001)

TEST DATE/TIME: 2/28/2011 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The objective of this test is to verify that the part can structurally hold under loading.
This part will be tested in SolidWorks using the Finite Element Method analysis built in the program.
The test will provide displacement and Von Misses stress analysis.

EXPECTED RESULTS:

The control arm will not have any deformation during the force analysis.

ACTUAL RESULTS:

This resulted in a max von misses stress of 6.1 psi, which is less than the yield strength of 10998.1 psi. Thus, the control arm won't yield.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Name	Type	Min	Location	Max	Location
Stress1	VON: von Mises Stress	0.00790766 psi Node: 11799	(5.78299 in, -1.26959e-006 in, 3.347e-010 in)	6.09054 psi Node: 1	(7.75 in, 4.44139e-017 in, -0.25 in)
Displacement1	URES: Resultant Displacement	0 mm Node: 1	(7.75 in, 4.44139e-017 in, -0.25 in)	5.66411e-005 mm Node: 11121	(6.04938 in, -2.23247e-006 in, -3.91976 in)
Strain1	ESTRN: Equivalent Strain	8.66793e-010 Element: 1553	(1.60526 in, 0.363641 in, 6.39233 in)	4.97674e-007 Element: 1079	(7.68158 in, 0.149986 in, 0.197157 in)

Test Plan – Solar Car Team '11

Test Plan– Solar Car Team ‘11

TEST ITEM (TITLE): Upright Structural Testing

TEST CASE #: SP-003 (ex: BS-001)

TEST DATE/TIME: 3/3/2011 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The objective of this test is to verify that the part can structurally hold under loading.
This part will be tested in SolidWorks using the Finite Element Method analysis built-in the program.
The test will provide displacement and Von Misses stress analysis.

EXPECTED RESULTS:

The upright arm will not have any deformation during the force analysis.

ACTUAL RESULTS:

A max von Misses stress of 422.2 psi. This resulting stress is lower than the yield strength of aluminum 2024 of 10998.1 psi, thus the upright won’t yield under this load.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Name	Type	Min	Location	Max	Location
Stress1	VON: von Mises Stress	0.00350275 psi Node: 10834	(1.47308 in, 8.09689 in, -0.050692 in)	420.298 psi Node: 12960	(-0.234932 in, -0.085508 in, -5.50321e-007 in)
Displacement1	URES: Resultant Displacement	0 mm Node: 77	(2.75 in, 3.25 in, 0.5 in)	0.000255489 mm Node: 13231	(3.19152 in, 0.160703 in, -7.37576e-007 in)
Strain1	ESTRN: Equivalent Strain	1.73435e-010 Element: 4585	(1.41674 in, 7.8932 in, 0.0412874 in)	2.96203e-005 Element: 1947	(-0.249385 in, -0.0901947 in, 0.0416661 in)

Test Plan – Solar Car Team '11

TEST ITEM (TITLE): Front Suspension (Left and Right) Simulation

TEST CASE #: SP-004 (ex: BS-001)

TEST DATE/TIME: 3/3/2011 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The objective of this test is to verify the camber and caster angle changes on the front suspension to ensure optimum performance.
This system will be simulated in MSC ADAMS/Car using the parallel travel suspension simulation analysis built-in the program.
The test will provide camber angle vs. wheel travel, and the caster angle vs. wheel travel plots.

EXPECTED RESULTS:

The camber and caster angle change over the specified wheel travel of 2 inches is small.

ACTUAL RESULTS:

The camber and caster angle change over the 2 inch travel is minimal. Caster angle, shown in Figure 4.7, experiences a change of 0.0456° ranging from -0.0606° to -0.0150°.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Continue tuning until desired characteristics are achieved.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): Rear Suspension

TEST CASE #: SP-005 (ex: BS-001)

TEST DATE/TIME: 3/31/2011 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The objective of this test is to verify that the system can structurally hold under loading.
With the system installed on the car, the suspension behavior will be observed and ensure it behaves as expected.

EXPECTED RESULTS:

The rear suspension will hold the weight of the motor and car.

ACTUAL RESULTS:

The rear suspension holds the weight of the car and produces enough bound and rebound wheel travel.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

Adjust preload on shock if weight of the car changes or a different suspension behavior is desired.

COMMENTS:

Test Plan – Solar Car Team ’11

TEST ITEM (TITLE): Test regenerative braking signal when the regenerative braking handle is asserted

TEST CASE #: PGS-001 (ex: BS-001)

TEST DATE/TIME: 4/1/2011 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The regenerative braking system is to be asserted through a handle connected to a potentiometer. The signal is connected to the potentiometer to create a scaling effect from 0-5 V max; this signal needs to be tested using a digital multimeter, because each voltage on the scale equates to a desired signal for the given range. The analog signal will be connected to the microcontroller that will signal the motor controller on the amount of braking throttle force. The program code for this process will also be tested. The regenerative braking system as a whole will be tested for its functionality upon complete assembly on the body of the car.

EXPECTED RESULTS:

The regenerative signal is detected by the microcontroller when the regenerative braking handle is asserted.

ACTUAL RESULTS:

When the tire is raised off the ground and the regenerative braking is implemented the tire will stop almost immediately.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team ’11

TEST ITEM (TITLE): Test regenerative braking system charges the battery

TEST CASE #: PGS-002 (ex: BS-001)

TEST DATE/TIME: 4/1/2011 (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The regenerative braking system is to be asserted through a handle connected to a potentiometer. The analog signal is received by a microcontroller that will be programmed to assert the digital regenerative braking signal to the motor controller. The kinetic energy received by the motor due to the motion of the vehicle is expected to charge the battery system. The state of charge device will be used to test and thus, measure the magnitude of the current received by the battery system upon assertion of the regenerative braking system. The program code for this process will also be tested. The regenerative braking system as a whole will be tested for its functionality upon complete assembly on the body of the car. The car can also be loaded on a jack and the motor throttled to a certain limit, then the regenerative braking system can be applied; the state of charge system will detect the magnitude of current generated due to the braking force on the motor.

EXPECTED RESULTS:

The regenerative braking system provides some amperage to the battery system.

ACTUAL RESULTS:

The state of charge meter was capable of recording a ‘negative’ current or power flow back into the batteries when the regenerative braking handle was asserted.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): MCU Power Test

TEST CASE #: CS-001 (ex: BS-001)

TEST DATE/TIME: 02/10/2011 - 12:10PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

MCU board is hooked up to 9V, 1A power supply and turned on to ensure functionality. The MCU will be set to program mode and then set to run mode to make sure that the MCU starts up properly in both configurations.

EXPECTED RESULTS:

Because there is no power switch on the MCU, it should turn on as soon as the power supply is plugged in, and should cycle power when the reset button is pressed. When the boot switch is in debug mode, the power should come on, and the LEDs should light up in a sequence from left to right. When in run mode the LEDs should light up left to right and then right to left.

ACTUAL RESULTS:

When the power was supplied to the MCU, the speaker beeps and the LEDs flashed from left to right. The Boot mode was switched to run mode and the reset button was pressed. When the reset button was pressed, the speaker beeped again and the LEDs flashed from left to right and then from right to left.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The MCU can also be powered by a 9V battery connected to the external power terminals.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): SW-192 Relays Test

TEST CASE #: CS-002 (ex: BS-001)

TEST DATE/TIME: 02/16/2011 - 9:25AM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The relays will each be connected to a 12V power supply. The voltage between the contacts will be measured to make sure that the there is no continuity between the contacts. The power will then be supplied to the relay and the continuity will be tested again to make sure that there is continuity.

EXPECTED RESULTS:

The contacts should have continuity when the power is supplied and no continuity when the supply power is off.

ACTUAL RESULTS:

When the power was off, both poles had no continuity. When the power was turned on, both poles had continuity.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The relays each use about 850mA at 12V

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): 12V to 9V DC-DC Converter Test

TEST CASE #: CS-003 (ex: BS-001)

TEST DATE/TIME: 02/21/2011 - 1:10PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

Power will be supplied from a 12V Voltage source and the output voltage will be measured to check for proper function.

EXPECTED RESULTS:

The output voltage should read about 9V when the 12 is supplied

ACTUAL RESULTS:

When the 12V power was supplied the open circuit output voltage was 9.4V

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Uses about 40mA at 12V when open circuit.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): Fuse Box Test

TEST CASE #: CS-004 (ex: BS-001)

TEST DATE/TIME: 02/21/2011 - 2:43PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

12V is supplied from a power supply to the fuse box power terminals. The voltage ach of the supply terminals will be measured with and without a fuse.

EXPECTED RESULTS:

The output voltage of each of the terminals should read 12V when a fuse is installed and should be open circuit when a fuse is not installed

ACTUAL RESULTS:

Each of the terminals performed as expected.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): Potentiometers Test

TEST CASE #: CS-005 (ex: BS-001)

TEST DATE/TIME: 02/21/2011 - 3:00PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The resistance of each of the potentiometers will be measured to ensure that they are about 5kohm. Then the input resistance will be measured as the potentiometer is adjusted. Then a 5v source will be connected to ensure that the input reads the correct voltage when the potentiometers are adjusted

EXPECTED RESULTS:

The resistance of each potentiometer should measure 5kohm resistance. The voltage should be 5V initially and decrease to 0V as the potentiometer is adjusted

ACTUAL RESULTS:

Each of the potentiometers performed as expected. The regen potentiometer read 4.88kohms and the throttle potentiometer read 5.26kohms

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The throttle and regen inputs of the motor controller use a differential buffer on the input signal, so the potentiometers should read 5V in the resting position and about 0V at full throttle

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): Motor Controller Power Test

TEST CASE #: CS-006 (ex: BS-001)

TEST DATE/TIME: 02/16/2011 - 5:00PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The motor controller will be hooked up to the batteries directly using a contactor and precharge circuit to ensure that it is operational.

EXPECTED RESULTS:

The fans should spin for a few seconds when the power is supplied and the regen and throttle reference voltages should read 5V. The supply voltage should be about 96V.

ACTUAL RESULTS:

The battery voltage measured 96.3V. When the contactor switch was flipped the motor controller fans began to spin. The regen and throttles reference voltages measured 5V

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The motor controller takes about 15 seconds to begin to discharge after the power has been turned off

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): BMS Power Test

TEST CASE #: CS-007 (ex: BS-001)

TEST DATE/TIME: 02/17/2011 - 12:10PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

50V will be supplied from a power supply to the BMS to ensure that it is operational.

EXPECTED RESULTS:

When the power is supplied to the BMS, the 4 signal LEDs should light up, indicating that the BMS is on. The lights on the main contactor relay and the throttle relay should light up, indicating that they are closed.

ACTUAL RESULTS:

When the power was supplied to the BMS, the signal LEDs were flashing and the Relay indicators were off.

STATUS: ☐ PASSED ■ FAILED

FAILURE CAUSE(S):

The 50V supplied to the BMS was probably not enough to power it. The manual states the it can be supplied with voltages from 35-350V, but 50V does not work.

SUGGESTED SOLUTION(S):

Increases the supply voltage

COMMENTS:

Further tests may need to be performed in the future to find the minimum operating voltage of the BMS. This information may be needed if the number of batteries needs to be changed.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): MCU/12V to 9V DC-DC Integration Test

TEST CASE #: CS-008 (ex: BS-001)

TEST DATE/TIME: 02/27/2011 - 8:35PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The 9V to 12V DC-DC converter will be powered by a 12V power supply and used to power the MCU.

EXPECTED RESULTS:

When the 12V is supplied, the MCU should power on

ACTUAL RESULTS:

When the power was supplied, the MCU came on and started up normally.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The MCU uses about 150mA at 12V when powered on.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): MCU/Relay/Fuse Box Integration Test

TEST CASE #: CS-009 (ex: BS-001)

TEST DATE/TIME: 02/27/2011 - 10:45PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The motor relay and the MCU are powered through the fuse box which is supplied through a power supply.

EXPECTED RESULTS:

when the switch is turned on, the MCU should power on and turn on the motor relay after 10 seconds

ACTUAL RESULTS:

when the switch was turned on, the MCU came on and the motor relay came on after about 10.3 seconds

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The MCU and relay use about 980mA when powered on.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): Breakout Board Test

TEST CASE #: CS-010 (ex: BS-001)

TEST DATE/TIME: 02/24/2011 - 6:40PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

All of the wires from the breakout board are checked for continuity between each pin and its corresponding quick connect wiring terminal

EXPECTED RESULTS:

All of the pins with wires attached should have continuity.

ACTUAL RESULTS:

All connected wires have continuity.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): Speedometer Test

TEST CASE #: CS-011 (ex: BS-001)

TEST DATE/TIME: 10/22/2010 - 3:15PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The speedometer will be powered by a 12V power supply and the signal will be supplied by a function generator. The output of the function generator will be increased and the response recorded.

EXPECTED RESULTS:

The speed measured on the speedometer should increase as the frequency of the input signal is increased.

ACTUAL RESULTS:

The speedometer increases from 0 to120 mph as the input frequency increases from 0 to about 160Hz.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

A sine wave and square wave input seems to work better than a pulse input.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): 100V to 12V/Fuse Box Integration Test

TEST CASE #: CS-012 (ex: BS-001)

TEST DATE/TIME: 03/25/2011 - 4:20PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The 100V to 12V DC-DC converter will be powered by the batteries and the output measured.

EXPECTED RESULTS:

The output voltage at the fuse box input should be about 12V

ACTUAL RESULTS:

The voltage was 11.94V at the fuse box input

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): Full Control System Integration Test

TEST CASE #: CS-013 (ex: BS-001)

TEST DATE/TIME: 03/25/2011 - 6:00PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

All of major control components will be integrated and measured for functionality. A switch will turn on the BMS, then a second switch will turn on the MCU and begin the startup process. The test box will be used to simulate input from the potentiometers.

EXPECTED RESULTS:

When the BMS is turned on, the Main relay should close and the motor controller should come on and begin precharging. When the second switch is closed the MCU should turn on and should close the motor relay after about 10 seconds. Then the test boxshould be able to run the motor

ACTUAL RESULTS:

The system performed as expected.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The main contactor needs to be controlled by another switch besides the BMS power switch because the BMS will need to be on all of the time to monitor the status of the batteries.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): Dashboard Control Integration Test

TEST CASE #: CS-014 (ex: BS-001)

TEST DATE/TIME: 04/01/2011 - 4:30PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The will be started using the three control switches. The first switch turns the fuse box on. The second switch turns the MCU on and closes the main relay which begins the startup process. The third switch controls if the car is in neutral forward, or reverse.

EXPECTED RESULTS:

The fuse box should have no power until the switch is flipped. Once the switch is flipped, the fuse box should have 12V at its input. Once the second switch is flipped the car should start up. When the drive switch is in neutral the throttle should not respond. If the drive switch is in fwd, the car should move forward when the throttle is pressed and in reverse if the switch is

ACTUAL RESULTS:

The system performed as expected.

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The motor is mounted backward so as defined by the motor controller, reverse is actually forward, and forward is reverse.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): Battery Charger Integration Test

TEST CASE #: CS-015 (ex: BS-001)

TEST DATE/TIME: 03/31/2011 - 9:30AM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

The battery charger will be powered by the 208VAC outlet in the machine shop through the BMS AC relay and will charge the batteries.

EXPECTED RESULTS:

When the 208VAC power is supplied to the BMS, the BMS should detect the power supply and open the relay to provide power to the battery charger. The charger should power on and begin charging the batteries. The SOC meter should display a negative current while the batteries are charging.when the charging is complete the charger should turn off automatically.

ACTUAL RESULTS:

When the power was supplied to the BMS, the AC relay closed and the voltage into the charger was correct. After a few seconds, something inside the charger exploded and the AC power fuses were blown.

STATUS: ☐ PASSED ■ FAILED

FAILURE CAUSE(S):

The most likely cause is some kind of short circuit inside the charger, maybe from the carbon fiber dust that was in the car. The failure also could have something to do with the recommendation that the charger be plugged in for at least 2 hours once per month.

SUGGESTED SOLUTION(S):

Replace the battery charger.

COMMENTS:

A new battery charger will likely need to be purchased to replace the current one because the charger does not look repairable.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): State of Charge Meter Power Test

TEST CASE #: MS-001 (ex: BS-001)

TEST DATE/TIME: 02/27/2011 - 6:50PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

50V will be supplied from a power supply to the SOC prescaler to ensure that it is operational.

EXPECTED RESULTS:

When the power is supplied to the SOC meter, it should come on and display the input voltage on the screen (50V)

ACTUAL RESULTS:

When the power was supplied to the SOC meter, the power came on and the voltage read 49.95V

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The SOC Meter needs to be on all the time to retain SOC information. If it is power cycled, it will lose current SOC information and will have to be re-synchronized.

Test Plan – Solar Car Team ‘11

TEST ITEM (TITLE): State of Charge Meter Shunt Current Test

TEST CASE #: MS-002 (ex: BS-001)

TEST DATE/TIME: 02/27/2011 - 7:45PM (ex: 01/01/10 – 11:30 AM)

TEST CASE DESCRIPTION:

TEST TYPE: ■ TEST ☐ RE-TEST

Constant current will be passed through the shunt and the value will be read from the SOC to test the accuracy of the current gauge.

EXPECTED RESULTS:

The current displayed by the SOC meter should be the same as the output current of the power supply

ACTUAL RESULTS:

The current supplied was 0.5A. The current displayed by the SOC meter was 0.6A

STATUS: ■ PASSED ☐ FAILED

FAILURE CAUSE(S):

SUGGESTED SOLUTION(S):

COMMENTS:

The wires of the shunt to the SOC meter need to use a twisted pair of wires to minimize the interference for maximum accuracy. The SOC meter has a resolution of 0.1A. Ensure that the SOC meter is configured to uses the 400A,50mv shunt or the readings may be inaccurate.

9.3 Software

Attached is all the software written during the course of the project.

```
#ifndef _MAIN_ASM_H
#define _MAIN_ASM_H

#ifdef __cplusplus
    extern "C" { /* our assembly functions have C calling convention */
#endif

void asm_main(void);


#ifdef __cplusplus
    }
#endif

#endif /* _MAIN_ASM_H */
```

```
#include <hidef.h>          /* common defines and macros */
#include "derivative.h"     /* derivative-specific definitions */
#include "main_asm.h"       /* interface to the assembly module */
#include "LCD.h"
#include "delay.h"
#include "motor.h"
#include "keypad.h"


/*****************************************************************************
*******************
*
*  Default RAM Section (~16K): Put variables here.
*
*  RAM:  0x1000 TO 0x3FFF;
*
*
*
*****************************************************************************
*******************/

#pragma DATA_SEG DEFAULT
int  counter;
int  counter2;
char PRESSED_KEY;
char DISPLAY_CONTROL;
char FUNCTION_SET;



/*****************************************************************************
*******************
*
* Non-banked ROM Section (~16K): Put non-bankable (i.e. interrupt) routines here.
*
*  ROM_4000 0x4000 TO 0x7FFF;
*
*
*
*****************************************************************************
*******************/
#pragma CODE_SEG __FAR_SEG NON_BANKED




/*****************************************************************************
*******************
*
* Default Code Section: 14 16K Pages, 224K Total: Use for main parts of code and
constants
*
*  PAGE_30       0x308000 TO 0x30BFFF
*  PAGE_31       0x318000 TO 0x31BFFF
*  PAGE_32       0x328000 TO 0x32BFFF
*  PAGE_33       0x338000 TO 0x33BFFF
*  PAGE_34       0x348000 TO 0x34BFFF
*  PAGE_35       0x358000 TO 0x35BFFF
*  PAGE_36       0x368000 TO 0x36BFFF
*  PAGE_37       0x378000 TO 0x37BFFF
*  PAGE_38       0x388000 TO 0x38BFFF
```

```
*  PAGE_39       0x398000 TO 0x39BFFF
*  PAGE_3A       0x3A8000 TO 0x3ABFFF
*  PAGE_3B       0x3B8000 TO 0x3BBFFF
*  PAGE_3C       0x3C8000 TO 0x3CBFFF
*  PAGE_3D       0x3D8000 TO 0x3DBFFF
*
*********************************************************************************
******************/
#pragma CODE_SEG DEFAULT

//<function declarations>


/*********************************************************************************
******************
**
**  Main Function
**
**
*********************************************************************************
******************/
void main(void) {

  //startup code
        asm_main(); /* call the assembly function */

        LCD_justify_center();
  LCD_write_line("SOLAR CAR 2010",1);
  delay_X_ms(3000); //delay for 3s after intro message
  LCD_write_line("PRECHARGING....",1);
  delay_X_ms(7000); //delay for 7s after precharge message
  LCD_write_line("MOTOR READY",1);
  delay_X_ms(5000);
  init_speedometer();


  //main menu
  LCD_write_line("MAIN MENU",1);


  for(;;) {_FEED_COP();} /* feeds the dog forever */

  /* please make sure that you never leave main */
}


//-------------------------------------------------------------------------------
--------------------
```

```
;*****************************************************************
;* This stationery serves as the framework for a                 *
;* user application. For a more comprehensive program that       *
;* demonstrates the more advanced functionality of this          *
;* processor, please see the demonstration applications          *
;* located in the examples subdirectory of the                   *
;* Freescale CodeWarrior for the HC12 Program directory          *
;*****************************************************************

; export symbols
            XDEF        asm_main

; import symbols
            XREF        init_sci
            XREF        motor_precharge_delay
            XREF        LCD_on
            XREF        init_LCD
            XREF        init_keypad


; Include derivative-specific definitions
                INCLUDE 'derivative.inc'


; variable/data section
MY_EXTENDED_RAM: SECTION
; Insert here your data definition.




; code section
ASM_MAIN:     SECTION

; this assembly routine is called by the C/C++ application
asm_main:

            CLI         ;enable interrupts
            CALL        init_LCD
            CALL        LCD_on
            CALL        init_sci
            CALL        motor_precharge_delay

            RTC         ; return to caller
```

```
/* ------------------------------------------------------------------------
** This software is in the public domain, furnished "as is", without technical
** support, and with no warranty, express or implied, as to its usefulness for
** any purpose.
**
** delay.h
** functions for time delays
**
** Author: James Barge
** ------------------------------------------------------------------------*/

#ifndef _DELAY_h
#define _DELAY_h

#ifdef IDENT_H
static const char* const DELAY_h_Id =
  "$Id$";
#endif

//<#include directives>
#include <hidef.h>          /* common defines and macros */
#include "derivative.h"     /* derivative-specific definitions */

//<declarations>


//<function prototypes>

//----Assembly functions-----------------------------------------------------
#ifdef __cplusplus
    extern "C" { /* assembly functions have C calling convention */
#endif

    void delay_X_ms (int);
    void delay_X_us (int);

#ifdef __cplusplus
    }
#endif

#endif /*_DELAY_h*/
```

```
; ------------------------------------------------------------------------
; This software is in the public domain, furnished "as is", without technical
; support, and with no warranty, express or implied, as to its usefulness for
; any purpose.
;
; motor.asm
; gathers data from the motor controller
;
; Author: James Barge
; ------------------------------------------------------------------------


; export symbols
            XDEF        delay_X_ms
            XDEF        delay_X_us


; Include derivative-specific definitions
                        INCLUDE 'derivative.inc'

; code section
DELAY:      SECTION

;; -- fileScope ---------------------------------------------------------------
;; ----------------------------------------------------------------------------
;; delay_1_ms()
;;
;; This function utilizes timer channel 2's output compare to delay the
;; processor by 1 millisecond.
;;
;; parameters: None
;;
;;
;;
;; -- implementation ----------------------------------------------------------
;; This code is written assuming the MCU clock speed is 4Mhz (8Mhz bus)
;; ----------------------------------------------------------------------------
delay_1_ms:

            PSHD
            BSET          TSCR1,mTSCR1_TEN          ;Enable the timer
            BSET          TIOS,mTIOS_IOS2           ;Enable output compare ch2
            LDD           TCNT
            ADDD          #4000                     ;for 1ms delay (@Bus = 4Mhz)
            STD           TC2
            LDAB          #mTFLG1_C2F
            STAB          TFLG1

    wait_C2F:

            BRCLR         TFLG1,mTFLG1_C2F,wait_C2F ;wait until flag is set

            PULD
            RTS


;; -- fileScope ---------------------------------------------------------------
;; ----------------------------------------------------------------------------
;; delay_1_us()
;;
;; This function uses NOPs to delay the processor by about 1 microsecond.
;;
;;
```

```
;; parameters: None
;;
;;
;;
;;
;; -- implementation ------------------------------------------------------
;; This code is written assuming the MCU clock speed is 4Mhz (8Mhz bus)
;; ------------------------------------------------------------------------
delay_1_us:

            NOP
            NOP
            NOP
            NOP
            RTS


;; -- global --------------------------------------------------------------
;;
;; delay_X_ms(int)
;;
;; This function utilizes timer channel 2's output compare to delay the
;; processor by a specified time in milliseconds.
;;
;; parameters:
;;    int X: length of time delay (in milliseconds) Register D
;;
;;
;; -- implementation ------------------------------------------------------
;; This code is written assuming the MCU clock speed is 4Mhz (8Mhz bus).
;; Max delay is 65.536 seconds
;; ------------------------------------------------------------------------

delay_X_ms:
            PSHD
            PSHX
            TFR         D,X

    next_ms:
            JSR         delay_1_ms
            DEX
            BNE         next_ms

            PULX
            PULD
            RTC




;; -- global --------------------------------------------------------------
;; delay_X_us()
;;
;; This function utilizes timer channel 2's output compare to delay the
;; processor by a specified time in milliseconds.
;;
;; parameters:
;;    int X: approximate length of time delay (in microseconds)
;;
;;
;; -- implementation ------------------------------------------------------
;; This code is written assuming the MCU clock speed is 4Mhz (8Mhz bus).
;; The length of the time delay is not very accurate, so don't use for high
;; precision timing. X is 16-bit, so longest delay is about 65ms
```

```
;; ---------------------------------------------------------------------------

delay_X_us:
            PSHD
            PSHX
            TFR         D,X

    next_us:
            JSR         delay_1_us
            DEX
            BNE         next_us

            PULX
            PULD
            RTC
```

```
// ------------------------------------------------------------------------
//
// LCD.H
// Handles communication with 16x2 LCD display of microcontroller board.
//
// Author: James Barge
// ------------------------------------------------------------------------
#ifndef _LCD_H
#define _LCD_H

#ifdef IDENT_H
static const char* const LCD_H_Id =
  "$Id$";
#endif

//<#include directives>
#include <hidef.h>          /* common defines and macros */
#include "derivative.h"     /* derivative-specific definitions */



//<declarations>

//----Assembly functions-------------------------------------------------------
#pragma CODE_SEG __FAR_SEG NON_BANKED
#ifdef __cplusplus
    extern "C" { /* assembly functions have C calling convention */
#endif


      void LCD_send_4bit(char);
      void LCD_send_8bit(char);
      void init_LCD(void);
      void LCD_on(void);
      void LCD_off(void);
      void LCD_clear(void);
      void LCD_home(void);
      void LCD_cursor(void);
      void LCD_no_cursor(void);
      void LCD_blink(void);
      void LCD_no_blink(void);
      void LCD_bit_mode_4(void);
      void LCD_bit_mode_8(void);
      void LCD_left_shift_cursor(void);
      void LCD_right_shift_cursor(void);
      void LCD_left_shift_display(void);
      void LCD_right_shift_display(void);
      void LCD_normal_font(void);
      void LCD_large_font(void);
      void LCD_two_line_mode(void);
      void LCD_write_char(char);
      void LCD_set_address(char);
      void LCD_justify_left(void);
      void LCD_justify_right(void);
      void LCD_justify_center(void);
      void LCD_send_command(char);

      void LCD_write_line(char[],short int);
      void LCD_write_hex(char);
      void LCD_write_message(char[]);

#ifdef __cplusplus
    }
```

```
#endif
//---------------------------------------------------------------------------



//<inline function definitions>

#endif /*_LCD_H*/
```

```
// ----------------------------------------------------------------------------
//
// LCD.cpp
// Handles communication with 16x2 LCD display of microcontroller board
// (4-bit Mode).
//
// Author: James Barge
// ----------------------------------------------------------------------------
#ifdef IDENT_C
static const char* const LCD_C_Id =
  "$Id$";
#endif

// -- module ------------------------------------------------------------------
//
// <The intended audience for this section is the client of this class.
//
// This documentation section is for a detailed overview of the contents
// of this file. Explain class cohesion. Give a brief overview of the
// functions. Do not include details that are best documented in function
// headers. If the functions are dependent on one another, describe the
// dependencies. For example, maybe the init() member must be called once
// and only once before any other member.
//
// The module section must not include any implementation details. Discuss
// only those members and elements that are visible to clients.>
//
// -- implementation ----------------------------------------------------------
//
// <The intended audience for this section is the programmer maintaining this
// class. Include relevant implementation details such as the use of protected
// and private members and file-scoped static variables. Discuss the
// relationships of non-public functions.>
//
// ----------------------------------------------------------------------------

//<#include directives>

#include "LCD.h"
#include "delay.h"

//<static member variable definitions>


//<static (file-scoped) variable definitions>
#pragma DATA_SEG DEFAULT
int  JUSTIFY_SET = 0;

//<static (file-scoped) function prototypes>




//<function definitions>
#pragma CODE_SEG __FAR_SEG NON_BANKED

// -- public ------------------------------------------------------------------
// -- fileScope ---------------------------------------------------------------
//
// LCD_write_line(char[],short int)
//
//  writes a character string to a line of LCD display. limit of 16 characters
//  writes the line with the specified justification
```

```
// -- implementation ------------------------------------------------------
//  If data is longer than 16 char, message will be truncated.
// ------------------------------------------------------------------------
void LCD_write_line(char Data[],short int line){

    short int j = 0; //index of data
    short int length = 0;
    short int fill = 0; //number of spaces to fill

    //Determine which line is to be written to
    if(line == 2){

        LCD_set_address(0x40);
    }else{
        LCD_set_address(0x00);
    }

    //find length of string

    while(Data[length] != 0x00){
        length++;
    }

    //check to see if length is longer than 16 char
    if(length >= 16){
        j = length-16;
        fill = 0;
    } else{
        fill = 16-length;
    }

    //write to line
    switch(JUSTIFY_SET){
      case 1: //justify left


          for(short int i=0;i<16;i++){    //fill the whole line
              //write data from left
              if(Data[j] != 0x00){
                  LCD_write_char(Data[j]);
                  j++;
              } else{
                //if data is < 16 chars, fill the rest
                LCD_write_char(' ');
              }
          }


      case 2: //justify right

          for(short int i=0;i<16;i++){  //fill the whole line

              //fill spaces before data (if any)
              if (fill != 0){
                  LCD_write_char(' ');
                  fill--;
              } else { //write data
                  LCD_write_char(Data[j]);
                  j++;
              }
          }
```

```
      case 4: //justify center

          fill = fill/2;       //fill half before, half after
          for(short int i=0;i<16;i++){

              if(fill != 0){
                  LCD_write_char(' ');
                  fill--;
              }else if(j<length){
                  LCD_write_char(Data[j]);
                  j++;
              } else{
                  LCD_write_char(' ');
              }
          }
    };
}

// -- public ---------------------------------------------------------------
// -- fileScope ------------------------------------------------------------
//
// LCD_write_message(char[],short int)
//
//  writes a scrolling message to the LCD display. limit of 24 chars
//
// -- implementation -------------------------------------------------------
//  scroll speed may need to be adjusted for readability. overlaps to second
//  line after 24 chars,need to find a way around this (if any).
// -------------------------------------------------------------------------
 void LCD_write_message(char Data[]){

    LCD_clear();
    int i = 0;
    int speed = 375;
    LCD_set_address(0x10);
    while(Data[i] != 0){
        LCD_write_char(Data[i]);
        i++;
        LCD_left_shift_display();
        delay_X_ms(speed);
    }
    for(int i=0;i<16;i++){
        LCD_left_shift_display();
        delay_X_ms(speed);
    }
}

// -- public ---------------------------------------------------------------
// -- fileScope ------------------------------------------------------------
//
// LCD_write_hex(char)
//
//  writes a scrolling message to the LCD display. limit of 24 chars
//
// -- implementation -------------------------------------------------------
//  scroll speed may need to be adjusted for readability. overlaps to second
//  line after 24 chars,need to find a way around this (if any).
// -------------------------------------------------------------------------
void LCD_write_hex(char Data){

    //
    volatile Byte part1;
    volatile Byte part2;
```

```
    part1 = Data & 0xF0;
    part1 = part1 >> 4;
    part2 = Data & 0x0F;

}
```

```
; ------------------------------------------------------------------------
; This software is in the public domain, furnished "as is", without technical
; support, and with no warranty, express or implied, as to its usefulness for
; any purpose.
;
; LCD.asm
; Handles communication with 16x2 LCD display of microcontroller board
; (4-bit Mode)
;
; Author: James Barge
; ------------------------------------------------------------------------


; export symbols

            XDEF        init_LCD
            XDEF        LCD_on
            XDEF        LCD_off
            XDEF        LCD_clear
            XDEF        LCD_home
            XDEF        LCD_cursor
            XDEF        LCD_no_cursor
            XDEF        LCD_blink
            XDEF        LCD_no_blink
            XDEF        LCD_bit_mode_4
            XDEF        LCD_bit_mode_8
            XDEF        LCD_left_shift_cursor
            XDEF        LCD_right_shift_cursor
            XDEF        LCD_left_shift_display
            XDEF        LCD_right_shift_display
            XDEF        LCD_normal_font
            XDEF        LCD_large_font
            XDEF        LCD_two_line_mode
            XDEF        LCD_write_char
            XDEF        LCD_set_address
            XDEF        LCD_justify_left
            XDEF        LCD_justify_right
            XDEF        LCD_justify_center
            XDEF        LCD_send_command

            XREF        delay_X_us,delay_X_ms,JUSTIFY_SET
            XREF        DISPLAY_CONTROL
            XREF        FUNCTION_SET
; Include derivative-specific definitions
                        INCLUDE 'derivative.inc'


; variable/data section
MY_EXTENDED_RAM: SECTION




; code section
LCD:     SECTION

;; -- public ----------------------------------------------------------------

;;
;; LCD_send_4bit(char)
;;
;;  Writes data to the lcd using 4-bit mode. char is in reg B
;;
```

```
;; -- implementation ------------------------------------------------------
;;
;;  PORTK bits 5-2 are used to send the data.
;; ------------------------------------------------------------------------
LCD_send_4bit:

            PSHD

            LSLB
            LSLB
            BCLR        PORTK,mPORTK_BIT2|mPORTK_BIT3|mPORTK_BIT4|mPORTK_BIT5;clear
data bits
            LDAA        PORTK
            ABA
            STAA        PORTK


            BSET        PORTK,mPORTK_BIT1
            LDD         #5
            CALL        delay_X_us
            BCLR        PORTK,mPORTK_BIT1

            PULD
            RTC

;; -- private -------------------------------------------------------------
;; -- fileScope -----------------------------------------------------------
;;
;; LCD_send_8bit(char)
;;
;;  Uses 4-bit mode to send 8-bits of data to the LCD.
;;
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
LCD_send_8bit:

            PSHB

            PSHB
            ANDB        #$F0;
            LSRB
            LSRB
            LSRB
            LSRB
            CALL        LCD_send_4bit
            PULB
            ANDB        #$0F
            CALL        LCD_send_4bit

            PULB
            RTS

;; -- public --------------------------------------------------------------
;; -- fileScope -----------------------------------------------------------
;;
;; LCD_write_char(char)
;;
;;  displays character on LCD display
;;
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
```

```
LCD_write_char:

            BSET        PORTK,mPORTK_BIT0
            JSR         LCD_send_8bit
            RTC

;; -- private -----------------------------------------------------------
;; -- fileScope ---------------------------------------------------------
;;
;; LCD_send_command(char)
;;
;;  sends data to the LCD in instruction mode.
;;
;; -- implementation ----------------------------------------------------
;;
;; ----------------------------------------------------------------------
LCD_send_command:

            BCLR        PORTK,mPORTK_BIT0
            JSR         LCD_send_8bit
            RTC



;; -- public ------------------------------------------------------------
;; -- fileScope ---------------------------------------------------------
;;
;; LCD_set_address(char)
;;
;;  changes the address counter value of LCD display. In one-line mode,
;;  address range is for 0x00 to 0x4F.in two-line mode the first line is from
;;  0x00 to 0x27, and the second line ranges from 0x40 to 0x67.
;;
;; -- implementation ----------------------------------------------------
;;    later make addresses outside range default to 0x00;
;; ----------------------------------------------------------------------
LCD_set_address:

            ORAB        #$80
            CALL        LCD_send_command
            RTC



;;--------commands-------------------------------------------------------


;; -- public ------------------------------------------------------------
;; -- fileScope ---------------------------------------------------------
;;
;;  LCD_clear()
;;
;;  clears the whole display and sets display data RAM's address 0
;;  in address counter.
;;
;; -- implementation ----------------------------------------------------
;;  Command takes 1.52ms to execute. MCU delays during this time to prevent
;;  trying to send another command while LCD is busy.
;; ----------------------------------------------------------------------
LCD_clear:

            PSHD
```

```
            LDAB        #$01
            CALL        LCD_send_command
            LDD         #2
            CALL        delay_X_ms

            PULD
            RTC

;; -- public --------------------------------------------------------------
;; -- fileScope -----------------------------------------------------------
;;
;;  LCD_home()
;;
;;  Sets display data RAM's address 0 in address counter and display returns
;;  to its original position. The cursor or blink goes to the left edge of
;;  the display (to the 1st line if 2 lines are displayed). The contents of
;;  the Display Data RAM do not change.
;;
;; -- implementation ------------------------------------------------------
;;  Command takes 1.52ms to execute. MCU delays during this time to prevent
;;  trying to send a command while LCD is busy.
;; ------------------------------------------------------------------------
LCD_home:

            PSHD

            LDAB        #$02
            CALL        LCD_send_command
            LDD         #2
            CALL        delay_X_ms

            PULD
            RTC

;;------------------------------------------------------------------------


;;/----------Display On/Off Control-----------------------------------------

;; -- public --------------------------------------------------------------
;; -- fileScope -----------------------------------------------------------
;;
;;  LCD_on()
;;
;;  Turns LCD display on.
;;
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
LCD_on:

            PSHB

            BSET        DISPLAY_CONTROL,$04
            LDAB        DISPLAY_CONTROL
            CALL        LCD_send_command

            PULB
            RTC

;; -- public --------------------------------------------------------------
;; -- fileScope -----------------------------------------------------------
;;
```

```
;;  LCD_off()
;;
;;  Turns LCD display off.
;;
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
LCD_off:

            PSHB

            BCLR        DISPLAY_CONTROL,$04
            LDAB        DISPLAY_CONTROL
            CALL        LCD_send_command

            PULB
            RTC

;; -- global --------------------------------------------------------------
;;
;;  LCD_cursor()
;;
;;  Turns cursor display on.
;;
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
LCD_cursor:

            PSHB

            BSET        DISPLAY_CONTROL,$02
            LDAB        DISPLAY_CONTROL
            CALL        LCD_send_command

            PULB
            RTC

;; -- global --------------------------------------------------------------
;;
;;  LCD_no_cursor()
;;
;;  Turns cursor display off.
;;
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
LCD_no_cursor:

            PSHB

            BCLR        DISPLAY_CONTROL,$02
            LDAB        DISPLAY_CONTROL
            CALL        LCD_send_command

            PULB
            RTC

;; -- global --------------------------------------------------------------
;;
;;  LCD_blink()
;;
;;  Turns cursor blinks on.
```

```
;;
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
LCD_blink:

            PSHB

            BSET        DISPLAY_CONTROL,$01
            LDAB        DISPLAY_CONTROL
            CALL        LCD_send_command

            PULB
            RTC


;; -- public --------------------------------------------------------------
;; -- fileScope -----------------------------------------------------------
;;
;;  LCD_no_blink()
;;
;;  Turns cursor blinks off.
;;
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
LCD_no_blink:

            PSHB

            BCLR        DISPLAY_CONTROL,$01
            LDAB        DISPLAY_CONTROL
            CALL        LCD_send_command

            PULB
            RTC

;;/----Cursor or Display Shift-----------------------------------------------

;; -- global --------------------------------------------------------------
;;
;;  LCD_left_shift_cursor()
;;
;;  Without changing DD RAM's daters, it can move cursor and shift display
;;  Shift cursor to the left
;;
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------

LCD_left_shift_cursor:

            PSHB

            LDAB        #$10
            CALL        LCD_send_command

            PULB
            RTC

;; -- public --------------------------------------------------------------
;; -- fileScope -----------------------------------------------------------
;;
```

```
;;  LCD_right_shift_cursor()
;;
;;  Without changing DD RAM's daters, it can move cursor and shift display
;;  Shift cursor to the right.
;;
;; -- implementation ---------------------------------------------------
;;
;; ---------------------------------------------------------------------------
LCD_right_shift_cursor:

            PSHB

            LDAB        #$14
            CALL        LCD_send_command

            PULB
            RTC


;; -- public -----------------------------------------------------------------
;; -- fileScope --------------------------------------------------------------
;;
;;  LCD_left_shift_display()
;;
;;  Without changing DD RAM's daters, it can move cursor and shift display
;;  Shift display to the left. Cursor follows the display shift.
;;
;; -- implementation ---------------------------------------------------
;;
;; ---------------------------------------------------------------------------
LCD_left_shift_display:

            PSHB

            LDAB        #$18
            CALL        LCD_send_command

            PULB
            RTC

;; -- public -----------------------------------------------------------------
;; -- fileScope --------------------------------------------------------------
;;
;;  LCD_right_shift_display()
;;
;;  Without changing DD RAM's daters, it can move cursor and shift display.
;;  Shift display to the left. Cursor follows the display shift.
;;
;; -- implementation ---------------------------------------------------
;;
;; ---------------------------------------------------------------------------
LCD_right_shift_display:

            PSHB

            LDAB        #$1C
            CALL        LCD_send_command

            PULB
            RTC


;;----Function Set-------------------------------------
```

```
;; -- public ------------------------------------------------------------
;; -- fileScope ---------------------------------------------------------
;;
;;  LCD_bit_mode_8()
;;
;;  Sets interface data length.
;;  Datas are transferred with 8-bit lengths (DB7 - DB0)
;;
;; -- implementation ----------------------------------------------------
;; Do not Use, board is hardwired for 4-bit operation.
;; ----------------------------------------------------------------------
LCD_bit_mode_8:

            PSHB

            BSET        FUNCTION_SET,$08
            LDAB        FUNCTION_SET
            CALL        LCD_send_command

            PULB
            RTC


;; -- public ------------------------------------------------------------
;; -- fileScope ---------------------------------------------------------
;;
;;  LCD_bit_mode_4
;;
;;  Sets interface data length.
;;  Datas are transferred with 4-bit lengths (DB7 - DB4).
;;
;; -- implementation ----------------------------------------------------
;;
;; ----------------------------------------------------------------------
LCD_bit_mode_4:

            PSHB

            BCLR        FUNCTION_SET,$08
            LDAB        FUNCTION_SET
            CALL        LCD_send_command

            PULB
            RTC

;; -- public ------------------------------------------------------------
;; -- fileScope ---------------------------------------------------------
;;
;;  LCD_normal_font()
;;
;;  Sets the number of the display lines.
;;  One-line display, 5 x 7 dots character font.
;;
;; -- implementation ----------------------------------------------------
;;
;; ----------------------------------------------------------------------
LCD_normal_font:

            PSHB

            BCLR        FUNCTION_SET,$02
            BCLR        FUNCTION_SET,$04
```

```
            LDAB        FUNCTION_SET
            CALL        LCD_send_command

            PULB
            RTC

;; -- public --------------------------------------------------------------
;; -- fileScope -----------------------------------------------------------
;;
;;  LCD_large_font()
;;
;;  Sets the number of the display lines.
;;  One-line display, 5 x 10 dots character font.
;;.
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
LCD_large_font:

            PSHB

            BSET        FUNCTION_SET,$02
            BCLR        FUNCTION_SET,$04
            LDAB        FUNCTION_SET
            CALL        LCD_send_command

            PULB
            RTC


;; -- public --------------------------------------------------------------
;; -- fileScope -----------------------------------------------------------
;;
;;  LCD_two_line_mode()
;;
;;  Sets the number of the display lines.
;;  Two-line display, 5 x 7 dots character font.
;;.
;; -- implementation ------------------------------------------------------
;;
;; ------------------------------------------------------------------------
LCD_two_line_mode:

            PSHB

            BCLR        FUNCTION_SET,$02
            BSET        FUNCTION_SET,$04
            LDAB        FUNCTION_SET
            CALL        LCD_send_command

            PULB
            RTC

;;-------------------------------------------------------------------------



;;-----------------------------------------------------------------

LCD_justify_left:

            MOVW        #$0001,JUSTIFY_SET
            RTC
```

```
LCD_justify_right:

            MOVW        #$0002,JUSTIFY_SET
            RTC


LCD_justify_center:

            MOVW        #$0004,JUSTIFY_SET
            RTC




;; -- global ----------------------------------------------------------------
;;
;; LCD__LCD()
;;
;; Constructor that is used to initialize the LCD display.
;;
;; -- implementation --------------------------------------------------------
;;
;;  Timing delays may need to be adjusted as they have not been optimized yet.
;; --------------------------------------------------------------------------
init_LCD:
            PSHD

            ;init I/O
            MOVB        #$FF,DDRK
            BCLR        PORTK,mPORTK_BIT1
            BCLR        PORTK,mPORTK_BIT0
            BCLR        PORTK,mPORTK_BIT7
            MOVB        #$08,DISPLAY_CONTROL
            MOVB        #$30,FUNCTION_SET

            ;;Startup sequence
            LDAB        #$03
            CALL        LCD_send_4bit
            LDD         #10
            CALL        delay_X_ms          ;wait for at leat 4.1ms
            LDAB        #$03
            CALL        LCD_send_4bit
            LDD         #1
            CALL        delay_X_ms          ;wait for at least 100us
            LDAB        #$03
            CALL        LCD_send_4bit
            LDAB        #$02
            CALL        LCD_send_4bit
            LDAB        #$28
            CALL        LCD_send_command    ;Fuction Set (4-bit),two-line mode
            LDAB        #$0C
            CALL        LCD_send_command    ;display on,cursor off,blink off
            LDAB        #$01
            CALL        LCD_send_command    ;clear display
            LDAB        #$07
            CALL        LCD_send_command    ;entry shift left mode
            LDD         #20
            CALL        delay_X_ms

            PULD
            RTC
```

```
/* ------------------------------------------------------------------------
** This software is in the public domain, furnished "as is", without technical
** support, and with no warranty, express or implied, as to its usefulness for
** any purpose.
**
** motor.h
** gathers data from the motor controller
**
** Author: James Barge
** ------------------------------------------------------------------------*/
#ifndef _motor_h
#define _motor_h

#ifdef IDENT_H
static const char* const motor_h_Id =
  "$Id$";
#endif

//<#include directives>
#include "LCD.h"

//<declarations>
#define  CR     0x0D
#define  LF     0x0A
//<function prototypes>

//----Assembly functions-------------------------------------------------------
#ifdef __cplusplus
    extern "C" { /* assembly functions have C calling convention */
#endif

void init_sci(void);
void sci_send_char(char);
void sci_send_message(char[]);
void motor_precharge_delay(void);
void enable_motor(void);
void init_speedometer(void);
void interrupt 9 TOC1_ISR (void);



#ifdef __cplusplus
    }
#endif

#endif /*_motor_h*/
```

```
;;-------------------------------------------------------------------------
;;This software is in the public domain, furnished "as is", without technical
;;support, and with no warranty, express or implied, as to its usefulness for
;;any purpose.
;;
;;motor.asm
;;gathers data from the motor controller
;;
;;Author: James Barge
;;-------------------------------------------------------------------------


;Include derivative-specific definitions
                        INCLUDE 'derivative.inc'

;export symbols
            XDEF        init_sci
            XDEF        sci_send_char
            XDEF        motor_precharge_delay
            XDEF        TOC1_ISR
            XDEF        TOC2_ISR
            XDEF        sci_send_message
            XDEF        init_speedometer

;import symbols
            XREF        counter
            XREF        counter2
            XREF        LCD_write_line
            XREF        delay_X_ms

;variable/data section
DEFAULT_RAM: SECTION

PULSE_COUNT: DS.W        1
TCNTPROD_HI: DS.W        1
TCNTPROD_LO: DS.W        1
KP:          DS.W        1


;----Drive States (values of RAM addr
0x96)-------------------------------------------------
DS_POWERUP              EQU     32      ;Initial state
DS_POWERUPEND           EQU     63      ;Power-up period over
DS_SHUTDOWN             EQU     64      ;Stopped and disabled
DS_DISABLECOAST         EQU     65      ;Disabled but not stopped
DS_INTERLOCK            EQU     66      ;Type 1 fault detected, waiting for disable
command
DS_INTERLOCKCOAST       EQU     67      ;Type 1 fault detected, waiting for disable
command, not stopped
DS_STOPPED              EQU     74      ;Enabled but not moving or throttling
DS_COASTING             EQU     75      ;Enabled and moving but not throttling
DS_NO_LONGER_THR        EQU     76      ;Leaving DS_THR mode
DS_NO_LONGER_BRK        EQU     77      ;Leaving DS_BRK mode
DS_THR                  EQU     78      ;Throttling
DS_BRK                  EQU     89      ;Braking
DS_PROGRAM              EQU     1       ;Shutdown with programming enabled



;-<externally referenced variable definitions>

;-----RAM
Values----------------------------------------------------------------------
```

```
;----serial inputs-----------------------------------------------
SI_DESIREDSPEED:        DS.B    2       ;(0x00) Serial speed in (deci-Hz)
SI_THRILIMIT:           DS.B    2       ;(0x01) Serial throttle limit in (deci-A)
SI_BRKILIMIT:           DS.B    2       ;(0x02) Serial regen current limit in (dA)
SI_KP:                  DS.B    2       ;(0x03) Proportional coefficient for speed
control
SI_KI:                  DS.B    2       ;(0x04) Integral coefficient for speed
control
SI_KT:                  DS.B    2       ;(0x05) Phase current to speed error speed
control coefficient
SI_MAXSPEEDERROR:       DS.B    2       ;(0x07) Speed error clamping value
SI_PHASEIRAMP:          DS.B    2       ;(0x09) Ramp rate for serial PhaseI input
deciA/(seconds/60)
SI_SPEEDRAMP:           DS.B    2       ;(0x0A) Ramp rate for serial speed input,
deci-hz electrical/(seconds/15)
SI_DESIREDPHASEI:       DS.B    2       ;(0x60) Serial phase current in (dA)
SI_MINFANSPEED:         DS.B    1       ;(0x90) Minimum fan speed (0-3)
SI_UL:                  DS.B    16      ;(0xF0-0xF3) Used as input registers


;-----Command Boolean
CB_DISABLE:             DS.B    1       ;(0xAD) Serial disable input
CB_THRENABLE:           DS.B    1       ;(0xAE) Serial throttle enable input

;----State Variables (Read Only)---------------------------------
SV_TARGETPHASEI:        DS.B    2       ;(0x10) Target current (dA)
SV_THERMALLIMITMOTOR:   DS.B    2       ;(0x11) Thermal motor current limit (dA)
SV_HEATSINKDERATING:    DS.B    2       ;(0x12) Heatsink thermal derating ratio
SV_MAXTHRI:             DS.B    2       ;(0x13) Maximum throttle current (dA)
SV_MAXRGNI:             DS.B    2       ;(0x14) Maximum regen current (dA)
SV_DRIVESTATE:          DS.B    1       ;(0x96) Operating status
SV_FAULT1LATCH:         DS.B    1       ;(0x98) Latched values of below
SV_FAULT1:              DS.B    1       ;(0x99) Bit-coded fault indications that
prevent operation
SV_FAULT2:              DS.B    1       ;(0x9A) Bit-coded fault indications of sensor
problems
SV_FAULT3:              DS.B    1       ;(0x9B) Bit-coded fault indications of
warnings
SV_FAULT4:              DS.B    1       ;(0x9C) Bit-coded fault indications of
current limiting
SV_FANSPEED:            DS.B    1       ;(0x9D) Actual fan speed setting
SV_FORWARD:             DS.B    1       ;(0xA1) Actual operating direction
SV_SPEEDCONTROL:        DS.B    1       ;(0xA2) When true, speed control


;-----Input Values Set By Arbitration
Logic-----------------------------------------------------
IN_DESIREDSPEED:        DS.B    2       ;(0x0B) Desired speed
IN_RGNILIMIT:           DS.B    2       ;(0x1C) Discrete regen current limit I (dA)
IN_DISABLE:             DS.B    1       ;(0x9E) Disable input, equal to [CB_DISABLE]
| [BI_DISABLE] | wrong direction
IN_THRENABLE:           DS.B    1       ;(0x9F) Throttle enable input, true when
[CB_THRENABLE] AND [BI_THRENABLE]
IN_DESIREDPHASEI:       DS.B    2       ;(0x61) Phase current in (dA)
IN_FORWARD:             DS.B    1       ;(0xA0) Input direction


;-----Analog Measurement Values(Read
Only)-----------------------------------------------
AM_SPEED:               DS.B    2       ;(0x0C) Actual speed (deci-Hz)
AM_SUPPLYV:             DS.B    2       ;(0x64) Measured supply voltage (dV)
AM_MOTORT:              DS.B    2       ;(0x65) Measured motor temp (degrees C * 10)
```

```
AM_HTSINKT:             DS.B    2       ;(0x66) Measured heatsink temp (degrees C *
10)
AM_SUPPLYI:             DS.B    2       ;(0x67) Measured logic supply current (mA)

;-----Analog Input Values(Read
Only)----------------------------------------------
AI_THR:                 DS.B    2       ;(0x17) Discrete throttle in
AI_RGN:                 DS.B    2       ;(0x18) Discrete regen in

;-----Boolean measurement value(Read
Only)----------------------------------------------
BM_OBSERVEDDIR:         DS.B    1       ;(0x97) Observed direction of rotation

;-----Boolean Input Values(Read
Only)----------------------------------------------------
BI_DISABLE:             DS.B    1       ;(0xAA) State of digital disable input
BI_THRENABLE:           DS.B    1       ;(0xAB) State of throttle enable input
BI_FORWARD:             DS.B    1       ;(0xAC) State of forward input


;-----Other-----------------------------------------------------------------------------
--------
PRODUCT:                DS.B    4       ;(0xF8) Returns product string
BUILD:                  DS.B    4       ;(0xF9) Returns software build string
BUILDDATE:              DS.B    4       ;(0xFA) Returns build date string



;-----ROM
Values---------------------------------------------------------------------------

;-----Factory Settings(Read
Only)-------------------------------------------------------------
FS_ABSMINV:             DS.B    2       ;(0x02) Absolute minimum voltage for
operation (dV)
FS_ABSMAXVGUARD:        DS.B    2       ;(0x09) Voltage at absmaxthrI1 set point (dV)
FS_ABSMAXV:             DS.B    2       ;(0x0A) Absolute maximum voltage for
operation (dV)
FS_MINGUARDDELTA:       DS.B    2       ;(0x0B) Minimum difference between minV and
minVguard, also maxV
FS_SCHTSINKT:           DS.B    2       ;(0x15) Scale value for heatsink temperature
FS_ABSMAXTHRI0:         DS.B    2       ;(0x3A) Factory set maximum value for
[CG_MAXTHRI] (dA)
FS_ABSMAXTHRI1:         DS.B    2       ;(0x3B) Factory set maximum value for
[CG_MAXTHRI] (dA)
FS_ABSMAXRGNI0:         DS.B    2       ;(0x3C) Factory set maximum value for
[CG_MAXRGNI] (dA)
FS_ABSMAXRGNI1:         DS.B    2       ;(0x3D) Factory set maximum value for
[CG_MAXRGNI] (dA)
FS_HSINKITCOEFF:        DS.B    2       ;(0x3F) I^2t coefficient for estimating motor
temp
FS_OFSUPPLYV:           DS.B    2       ;(0x60) Offset value for supply voltage
FS_OFSUPPLYI:           DS.B    2       ;(0x61) Offset value for supply current
FS_HSINKTIMEC:          DS.B    1       ;(0xBA) Thermal time constant coefficient for
heatsink


;------Configuration
Values-----------------------------------------------------------------------
CG_BAUDRATE:            DS.B    2       ;(0x00) I/O baud rate
CG_MINV:                DS.B    2       ;(0x03) Voltage at which max throttle current
is zero (dV)
CG_MINVGUARD:           DS.B    2       ;(0x04) Voltage at which max throttle current
```

```
limiting starts (dV)
CG_MAXVRGNGUARD:        DS.B    2       ;(0x05) High voltage cut-off start point for
regen
CG_MAXVRGN:             DS.B    2       ;(0x06) Maximum voltage for regen
CG_MAXVGUARD:           DS.B    2       ;(0x07) Voltage at which max phase current
limiting begins due to over voltage (dV)
CG_MAXV:                DS.B    2       ;(0x08) Voltage at which phase current is
zero (dV)
CG_MINFREQ:             DS.B    2       ;(0x0C) Minimum commutation frequency for
speed control
CG_SCTHR_SPEED:         DS.B    2       ;(0x16) Scale value for throttle input into
speed (speed control)
CG_SCTHR_TORQUE:        DS.B    2       ;(0x17) Scale value for throttle input into
amps (torque control)
CG_SCRGN_TORQUE:        DS.B    2       ;(0x18) Scale value for regen input into amps
CG_SCMOTORT:            DS.B    2       ;(0x1A) Scale value for motor temperature
CG_MAXMOTORI:           DS.B    2       ;(0x1D) Maximum motor current, throttle or
regen (deci-Amps)
CG_SPEEDTHRESHOLD:      DS.B    2       ;(0x21) Safe speed for changing motor
direction
CG_MINSUPPLYI:          DS.B    2       ;(0x22) Minimum supply current when fans are
off
CG_MAXSUPPLYI:          DS.B    2       ;(0x23) Maximum supply current when fans are
off
CG_MINFANSUPPLYI:       DS.B    2       ;(0x24) Minimum supply current when fans are
on
CG_MAXFANSUPPLYI:       DS.B    2       ;(0x25) Maximum supply current when fans are
on
CG_FANI:                DS.B    8       ;(0x2C-0x2F) Current thresholds for fan
control
CG_MAXTHRI0:            DS.B    2       ;(0x36) Maximum throttle current (dA)
CG_MAXTHRI1:            DS.B    2       ;(0x37) Maximum throttle current (dA)
CG_MAXRGNI0:            DS.B    2       ;(0x38) Maximum regen current (dA)
CG_MAXRGNI1:            DS.B    2       ;(0x39) Maximum regen current (dA)
CG_MOTORITCOEFF:        DS.B    2       ;(0x3E) I^2t coefficient for estimating
heatsink temp
CG_OFMOTORT:            DS.B    2       ;(0x64) Offset value for motor temp
CG_FANTEMP:             DS.B    6       ;(0x65-0x67) Temperature transition points
for fan control
CG_DEFAULT_MOTORT:      DS.B    2       ;(0x71) Assumed motor temp when sensor fails
CG_TLIMTMTR:            DS.B    6       ;(0x72-0x75) Motor Temperature transition
points (deci-Celsius)
CG_TLIMIMTR:            DS.B    4       ;(0x7C-0x7D) 0-256 % of current at the
corresponding Temp. Implied denominator of 256
CG_ECHO:                DS.B    1       ;(0x90) When true, echo characters as they
are received
CG_TEXTERRORS:          DS.B    1       ;(0x91) When true, send text messages for
errors, else send two digit codes
CG_LINEFEEDS:           DS.B    1       ;(0x92) When true, use CR-LF combinations at
end of lines
CG_MAXSCIIDLE:          DS.B    1       ;(0x93) Maximum idle time for serial
interface watchdog fault in tenths of a second, 0 disables
CG_60DEGREEHALLS:       DS.B    1       ;(0x95) When true, assume hall-effect sensor
are 60 electrical degrees apart
CG_INVERTDIR:           DS.B    1       ;(0x97) When true, reverse definition of
forward
CG_ENDISCRETE_THR:      DS.B    1       ;(0x99) When true, use discrete throttle and
regen inputs
CG_ENDISCRETE_DIR:      DS.B    1       ;(0x9B) When true, use discrete direction
input
CG_ENDISCRETE_THRENABLE:DS.B    1       ;(0x9C) When true, use discrete throttle
enable input
CG_ENDISCRETE_DISABLE:  DS.B    1       ;(0x9D) When true, use discrete disable input
```

```
CG_THRDEADBAND:         DS.B    1       ;(0x9E) Offset (in counts) of throttle input
CG_RGNDEADBAND:         DS.B    1       ;(0x9F) Offset (in counts) of regen input
CG_GAPDEADBAND:         DS.B    1       ;(0xA0) Offset (in counts) of gap input (not
used)
CG_RTSUPPLYV:           DS.B    1       ;(0xA1) Filtering level for supply voltage
(0:none to 4:max)
CG_RTSUPPLYI:           DS.B    1       ;(0xA2) Filtering level for supply current
measurement
CG_RTHSINKT:            DS.B    1       ;(0xA4) Filtering level for heatsink temp
measurement
CG_RTHTR:               DS.B    1       ;(0xA5) Filtering level for throttle input
CG_RTRGN:               DS.B    1       ;(0xA6) Filtering level for regen input
CG_RTMOTORT:            DS.B    1       ;(0xA8) Filtering level for motor temp
measurement
CG_SOFTSTARTN:          DS.B    1       ;(0xA9) Speed of softstart operation
(0:fastest ramp to 7:slowest ramp)
CG_NAUTORESETS:         DS.B    1       ;(0xAA) Number of automatic reset attempts in
four seconds
CG_MOTORTIMEC:          DS.B    1       ;(0xB9) Thermal time constant coefficient for
motor
CG_AISPDTOPWMFREQMULT:  DS.B    1       ;(0xBB) Sets threshold for detecting max
torque production
CG_SPDNUMERATOR:        DS.B    4       ;(0xF0) Numerator used for speed calculation

;-----Default Values----------------------------------------------
DF_KI:                  DS.B    2       ;(0x0D) Default value for SI_KI
DF_KP:                  DS.B    2       ;(0x0E) Default value for SI_KP
DF_KT:                  DS.B    2       ;(0x0F) Default value for SI_KT
DF_PHASEIRAMP:          DS.B    2       ;(0x1E) Default value for SI_PHASEIRAMP
DF_SPEEDRAMP:           DS.B    2       ;(0x1F) Default value for SI_SPEEDRAMP
DF_MAXSPDERROR:         DS.B    2       ;(0x11) Clamping value for speed error in
speed control
DF_SPEEDCONTROL:        DS.B    1       ;(0x98) When true, power-up in speed control
mode

;-----Scale
Values----------------------------------------------------------------------------
SC_SUPPLYV:             DS.B    2       ;(0x12) Scale value for supply voltage
SC_SUPPLYI:             DS.B    2       ;(0x13) Scale value for supply current


;code section
MOTOR:     SECTION

TCNT_MAX:    DC.W        65535
TCNT_MIN:    DC.W        390
K:           DC.W        91

;;-- global ----------------------------------------------------------------
;
;void init_sci(void)
;;
;Initializes SCI1 to interface with motor controller. Sets baud rate to 9600,
;8 data bits, 1 stop bit, no flow control, no parity.
;
;;
;;-- implementation --------------------------------------------------------
;Inputs:  None
;Outputs: None
;Registers Modified: CCR
;Bus Speed is 4Mhz.
;
;;-----------------------------------------------------------------------------
```

```
init_sci:

            PSHD

            ;enable TX,RX, no interrupts
            BCLR        SCI1CR1,mSCI1CR1_M     ;8 bit data
            BCLR        SCI1CR1,mSCI1CR1_PE    ;No parity
            BSET        SCI1CR2,mSCI1CR2_TE    ;Transmitter enable
            BSET        SCI1CR2,mSCI1CR2_RE    ;Reciever enable
            ;set baud rate
            LDD         #26                    ;4Mhz/(16x26)=9600
            STD         SCI1BDH

            PULD
            RTC                                ;return to caller


init_speedometer:

            PSHD
            PSHY

            ;evaluate TCNT_MIN*TCNT_MAX and store (32-bit multiply)
            LDD         TCNT_MAX
            LDY         TCNT_MIN
            EMUL
            STY         TCNTPROD_HI
            STD         TCNTPROD_LO


            ;pulse accumulator A init
            BCLR        PACTL,mPACTL_PAMOD     ;event counter mode
            BSET        PACTL,mPACTL_PAEN      ;pulse accumulator enable
            BSET        DDRP,mDDRP_DDRP5       ;enable lights
            BSET        DDRP,mDDRP_DDRP4
            MOVW        #0,PACN32              ;init pulse accumulator


            ;timer CH2 init
            MOVW        #20,counter2           ;for 200ms delay
            BSET        TIOS,mTIOS_IOS2        ;Enable CH2 output compare
            MOVW        TCNT,TC2               ;transfer timer count to output
compare
            LDD         TC2
            ADDD        #40000                 ;for 10ms delay (@Bus = 4Mhz)
            STD         TC2
            BSET        TFLG1,mTFLG1_C2F       ;clear Timer CH1 Output compare flag
            BSET        TIE,mTIE_C2I           ;enable Timer CH2 interrupts


            ;PWM CH01 init
            BSET        PWMCTL,mPWMCTL_CON01   ;16-bit PWM01
            MOVB        #6,PWMPRCLK            ;divide clock by 64
            BSET        PWMPOL,mPWMPOL_PPOL1   ;pwm1 polarity high
            BSET        PWME,mPWME_PWME1       ;enable PWM Channel 1

            PULY
            PULD
            RTC


;;-- global ----------------------------------------------------------------
```

```
;
;void sci_send_char(char)
;;
;Sends a char to SCI1.
;
;
;;
;;-- implementation ----------------------------------------------------------
;Inputs:  B = char data to send
;Outputs: None
;Registers Modified: CCR
;Bus Speed is 4Mhz.
;
;;-----------------------------------------------------------------------------
sci_send_char:

    send_wait:
            ;wait until transmit data register is empty
            BRCLR       SCI1SR1,mSCI1SR1_TDRE,send_wait
            ;output the data
            STAB        SCI1DRL
            RTC


;;-- global -----------------------------------------------------------------
;
;void sci_send_message(char[])
;
;sends a string through SCI1.
;
;;-- implementation ----------------------------------------------------------
;Inputs:  pointer to data (in register D)
;Outputs: None
;Registers Modified: D,CCR
;
;;-----------------------------------------------------------------------------

sci_send_message:
            PSHX
            TFR         D,X
    next_char:
            LDAB        1,X+
            CMPB        #0
            BEQ         eom
            CALL        sci_send_char
            BRA         next_char
    eom:
            PULX
            RTC


;;-- global -----------------------------------------------------------------
;
;void motor_precharge_delay(void)
;;
;Start motor precharge delay timer. delays for 10 seconds, then generates
;interrupt to enable motor controller relay
;
;;
;;-- implementation ----------------------------------------------------------
;Inputs:  None
;Outputs: None
;Registers Modified: CCR
```

```
;
;;-------------------------------------------------------------------------

motor_precharge_delay:

            PSHD

            BSET        TSCR1,mTSCR1_TEN      ;Enable the timer
            BSET        TIOS,mTIOS_IOS1       ;Enable output compare
            MOVW        #10000,counter        ;for 10s delay
            MOVW        TCNT,TC1              ;transfer current time to output
compare
            LDD         TC1                   ;update timer compare
            ADDD        #4000                 ;for 1ms delay (@Bus = 4Mhz)
            STD         TC1
            BSET        TFLG1,mTFLG1_C1F      ;clear Timer CH1 Output compare flag
            BSET        TIE,mTIE_C1I          ;enable CH1 timer interrupts

            PULD
            RTC

;;-- interrupt ------------------------------------------------------------
;
;void interrupt 9 TOC1_ISR(void)
;
;Turns on motor controller relay after set period of time.
;
;;-- implementation -------------------------------------------------------
;Inputs:  None
;Outputs: None
;Registers Modified: CCR
;
;;-------------------------------------------------------------------------

NON_BANKED  SECTION

TOC1_ISR:

            PSHD
            PSHX

            ;decrement counter
            LDX         counter
            DEX                               ;decrement counter
            STX         counter

            ;add 4000 to TC1
            LDD         TC1
            ADDD        #4000                 ;for 1ms delay (@Bus = 4Mhz)
            STD         TC1

            ;if counter = 0,start motor
            LDD         counter
            BEQ         start_motor

            ;else exit
            PULX
            PULD
            BSET        TFLG1,mTFLG1_C1F      ;clear Timer CH1 Output compare flag

            RTI

  start_motor:
```

```
            BCLR        TIOS,mTIOS_IOS1       ;disable CH1 output compare
            BSET        DDRE,mDDRE_BIT2       ;Set PE2 as Output Port
                BSET        PORTE,mPORTE_BIT2     ;turn on relay at PE2
            PULX
            PULD
            RTI


;;-- interrupt ------------------------------------------------------------
;
;void interrupt 10 TOC2_ISR(void)
;
;Updates the PWM signal to the speedometer every 200ms.
;
;;-- implementation -------------------------------------------------------
;Inputs:  None
;Outputs: None
;Registers Modified: CCR
;
;;-------------------------------------------------------------------------

TOC2_ISR:


            PSHD
            PSHX
            PSHY

            ;decrement counter
            LDX         counter2
            DEX                               ;decrement counter
            STX         counter2

            ;add 4000 to TC1
            LDD         TC2
            ADDD        #40000                ;for 10ms delay (@Bus = 4Mhz)
            STD         TC2

            ;if counter = 0,start motor
            LDD         counter2
            BEQ         update_speed

            ;else exit
            PULY
            PULX
            PULD
            BSET        TFLG1,mTFLG1_C2F      ;clear Timer CH1 Output compare flag

            RTI


update_speed:



            ;check if stopped
            LDX         PACN32
            BEQ         stopped
            STX         PULSE_COUNT           ;store current pulse count
```

```
            ;update pwm

            ;evaluate KP
            LDD         K
            LDY         PULSE_COUNT
            EMUL
            STD         KP

            ;divide  TCNTPROD by KP, store period count
            LDY         TCNTPROD_HI
            LDD         TCNTPROD_LO
            LDX         KP
            EDIV                            ;32/16-bit Division. Y has TCNT, put
in D for duty cycle division
            BVS         stopped             ;if division overflows, moter is
close to stopped
            TFR         Y,D
            STD         PWMPER01

            ;evaluate duty cycle
            LDX         #2
            IDIV                            ;divide by 2 for 50% duty
cycle,quotient is in X
            STX         PWMDTY01                ;

            ;exit
            ;flash green light
            BSET        PTP,mPTP_PTP5
            LDD         #3
            CALL        delay_X_ms
            BCLR        PTP,mPTP_PTP5
            MOVW        #20,counter2        ;reset counter
            MOVW        #0,PACN32           ;reset pulse count
            BSET        TFLG1,mTFLG1_C2F    ;clear Timer CH1 Output compare flag
            PULY
            PULX
            PULD
            RTI

stopped:
            ;flash red light
            BSET        PTP,mPTP_PTP4
            LDD         #3
            CALL        delay_X_ms
            BCLR        PTP,mPTP_PTP4

            ;update pwm
            MOVW        #0,PWMDTY01         ;set frequency to 0

            ;exit
            MOVW        #20,counter2        ;reset counter
            MOVW        #0,PACN32           ;reset pulse count
            BSET        TFLG1,mTFLG1_C2F    ;clear Timer CH1 Output compare flag
            PULY
            PULX
            PULD
            RTI
```

9.4 Data Sheets

tbs electronics

e-xpert pro
High precision battery monitor

TECHNICAL DATASHEET

Description

The e-xpert pro selectively displays voltage, charge- and discharge current, consumed amphours, remaining battery capacity and time remaining of your battery bank. Using a clear backlit LC Display and an intuitive user interface, all parameters can be recalled with just a button press. A second battery input is also provided to monitor an auxiliary battery.

Defining the amount of energy available in a battery is a complex task since battery age, discharge current and temperature all influence the actual battery capacity. High performance measuring circuits, along with complex software algorithms, are used to exactly determine the remaining battery capacity. A new shunt selection feature enables the e-xpert pro to measure currents up to 10.000Amps.

The e-xpert pro is equipped with an internal programmable alarm relay, to run a generator when needed or to turn off devices when the battery voltage exceeds programmable boundaries.

The e-xpert pro comes standard with a 500 Amp shunt and a very clear installation and operating instruction manual.

Features

- Read your battery bank like a fuel gauge
- Provides critical information about the status of your battery bank
- Displays voltage, current, consumed amphours, remaining battery capacity and time remaining
- Two battery inputs
- Large backlit LC Display
- Quick nut mounting construction
- Fully programmable alarm relay
- Shunt selection capability enables flexible system integration
- Communication/expansion port
- Stores a wide range of history events in internal memory
- Splash proof frontpanel
- 500 Amp shunt included
- CE and e-mark certified
- 24 month warranty

Applications

- Recreational vehicles
- Solar power systems
- Industrial systems
- Forklifts
- Mobile entertainment systems
- Service vehicles
- Marine applications
- Remote homes

Accessories

- Temperature sensor
- Professional connection kit
- Quick connection kit
- Isolated RS-232 interface
- Isolated USB interface
- Alarm relay output expander
- 1:5 voltage prescaler
- Windows XP/Vista/7 software

Technical specifications

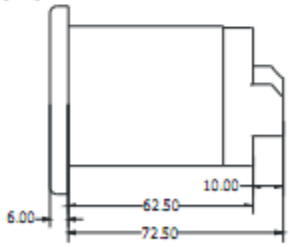
Parameter		e-xpert *pro*
Supply voltage range		9..35VDC
Supply current[1] :	@Vin=24VDC	7mA
	@Vin=12VDC	9mA
Input voltage range (auxiliary battery)		2..35VDC
Input voltage range (main battery)		0..35VDC
Input current range[2]		-9999..+9999A
Battery capacity range		20..9990Ah
Operating temperature range		-20..+50°C
Readout resolution :	voltage (0..35V)	± 0.01V
	current (0..200A)	± 0.1A
	current (200..9999A)	± 1A
	amphours (0..200Ah)	± 0.1Ah
	amphours (200..9990Ah)	± 1Ah
	state-of-charge (0..100%)	± 0.1%
	time remaining (0..24hrs)	± 1minute
	time remaining (24..240hrs)	± 1hr
	temperature (-20..+50°C)[3]	± 0.5°C
Voltage measurement accuracy		± 0.3%
Current measurement accuracy		± 0.4%
Dimensions :	frontpanel	Ø64mm
	body diameter	Ø52mm
	total depth	79mm
	weight	95gr
Shunt dimensions :	footprint	45 x 87mm
	height	17mm (base) / 35mm (high current screws)
	weight	145gr
Protection class		IP20 (frontpanel only IP65)
Complies with the following standards		EN61000-6-3 (EN55022), EN61000-6-2 (EN61000-2/3/4, EN61000-4-3), LVD 73/23/EEC (EN60335-1), e4-95/54/EC, RoHS 2002/95/EC

Note: the given specifications are subject to change without notice.

1) Measured with backlight and alarm relay turned off.
2) Depends on selected shunt. With standard delivered 500A/50mV shunt (350A continuous), the range is limited to -600..+600A.
3) Only available when optional temperature sensor is connected.

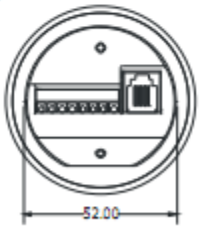
Dimensions

Sideview :

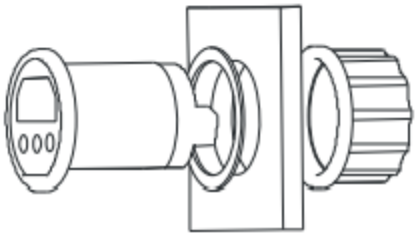


Frontview :

Rearview :



Mounting sequence :



SCM150-XXX Axial Flux, Brushless PM Motor

The New Generation Motors (NGM) SCM150 motor is ideal for small, direct drive, experimental electric vehicles where efficiency is key. The SCM150 motor can be driven as a 3-phase AC synchronous or DC brushless electric motor with advanced capabilities and superior efficiency. Standard features include:

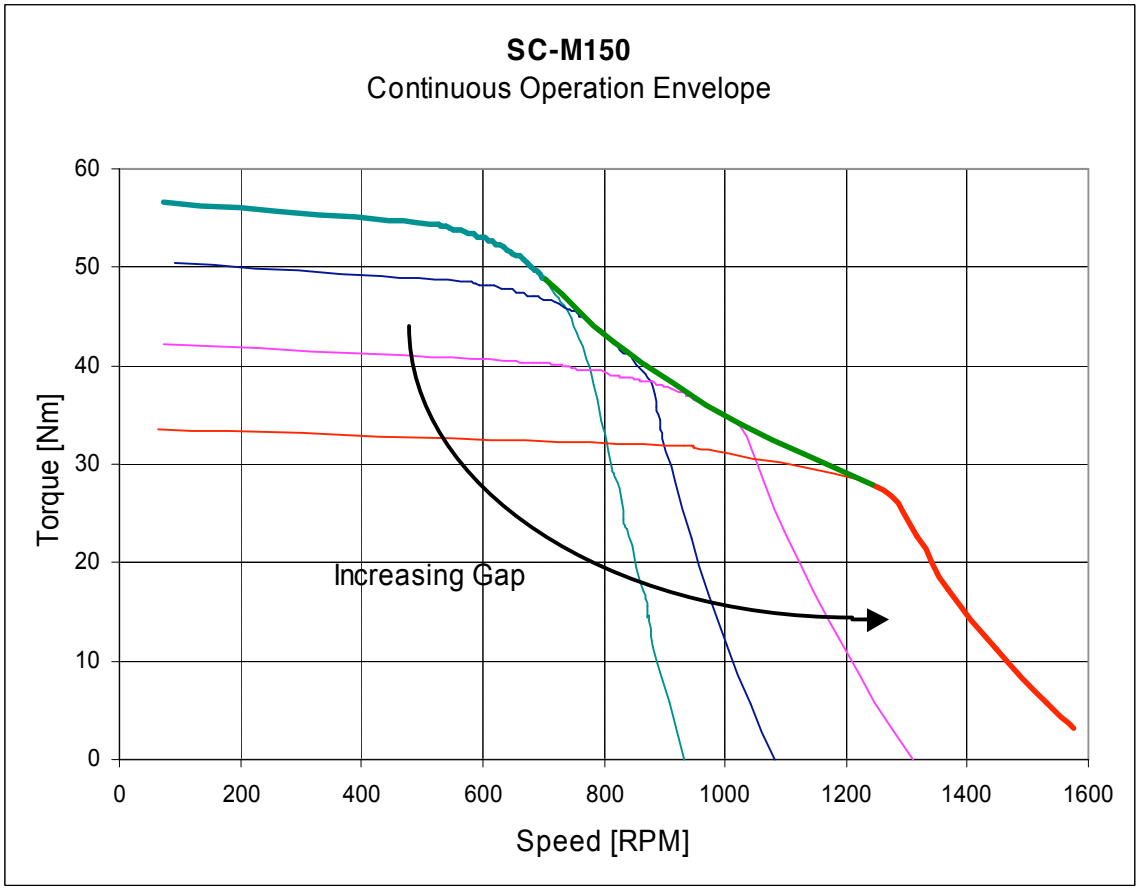
- *High Power Rare Earth Permanent Magnets*
- *Variable gap mechanism allows for torque constant to be changed on the fly.*
- *Fits NGM solar car wheel, NGM PN 300-000115.*
- *Ultra high efficiency*
- *Open motor design for improved thermal performance and reduced weight.*
- *Field proven -- Used by top 5 finishers in Sunrayce 99 and ASC 2001*

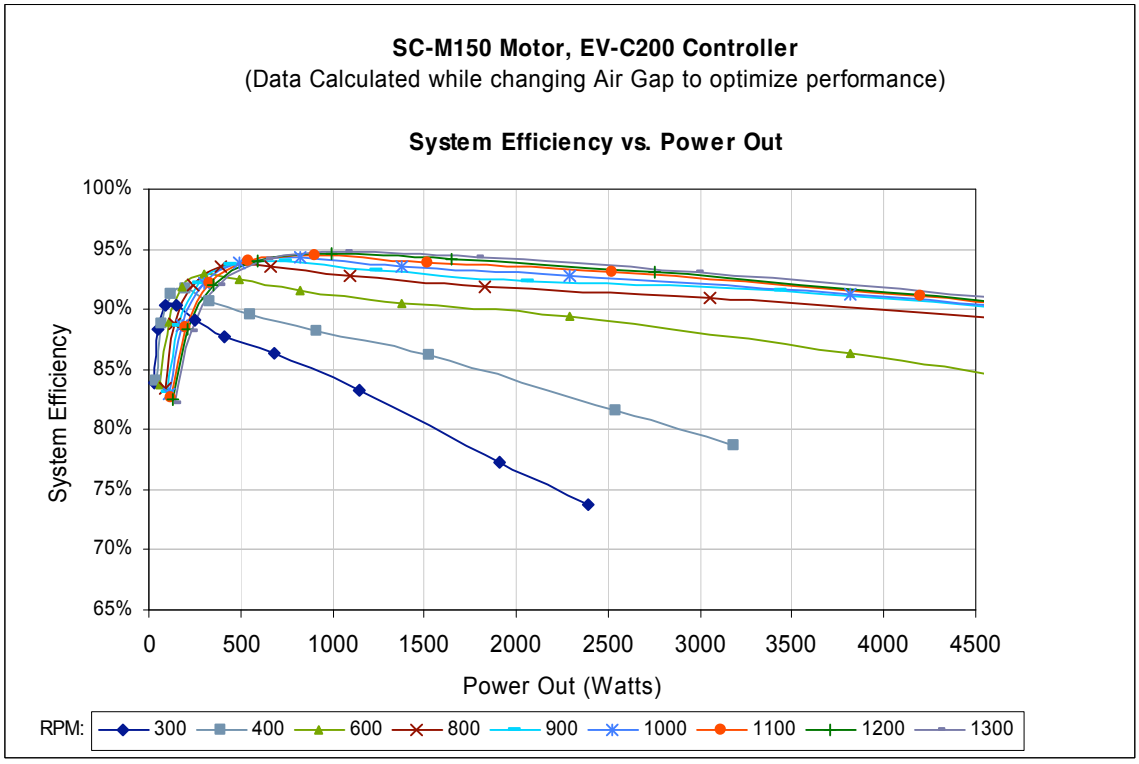
❋ **Specifications *(Preliminary)***

		SCM150
Peak Power	kW	7.5
Continuous Power @V_{nom}	kW	3.75
Speed @ Peak Power	RPM	1300
No-Load Speed	RPM	1700
Peak Torque @ I_{max} (T_p)	Nm	135
DC Bus Voltage Nominal (V_{nom})	V	96 & 48
Weight	kg	20

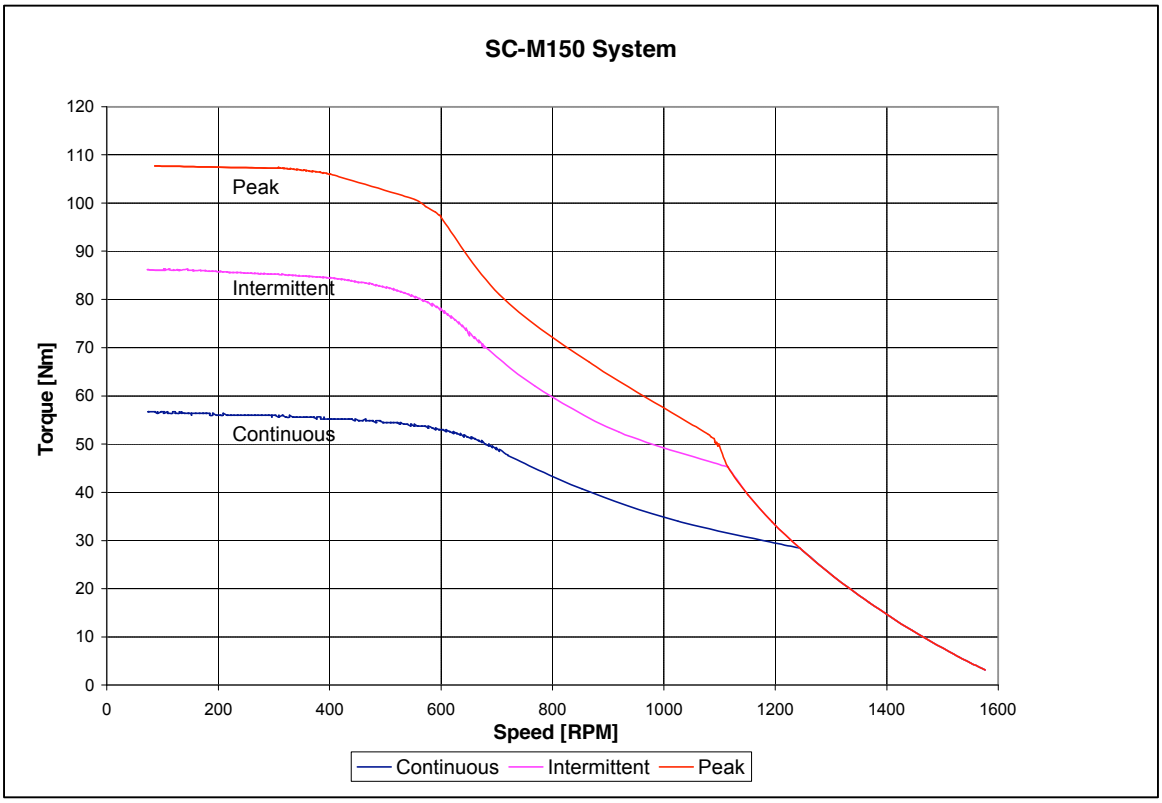
For more information, contact:

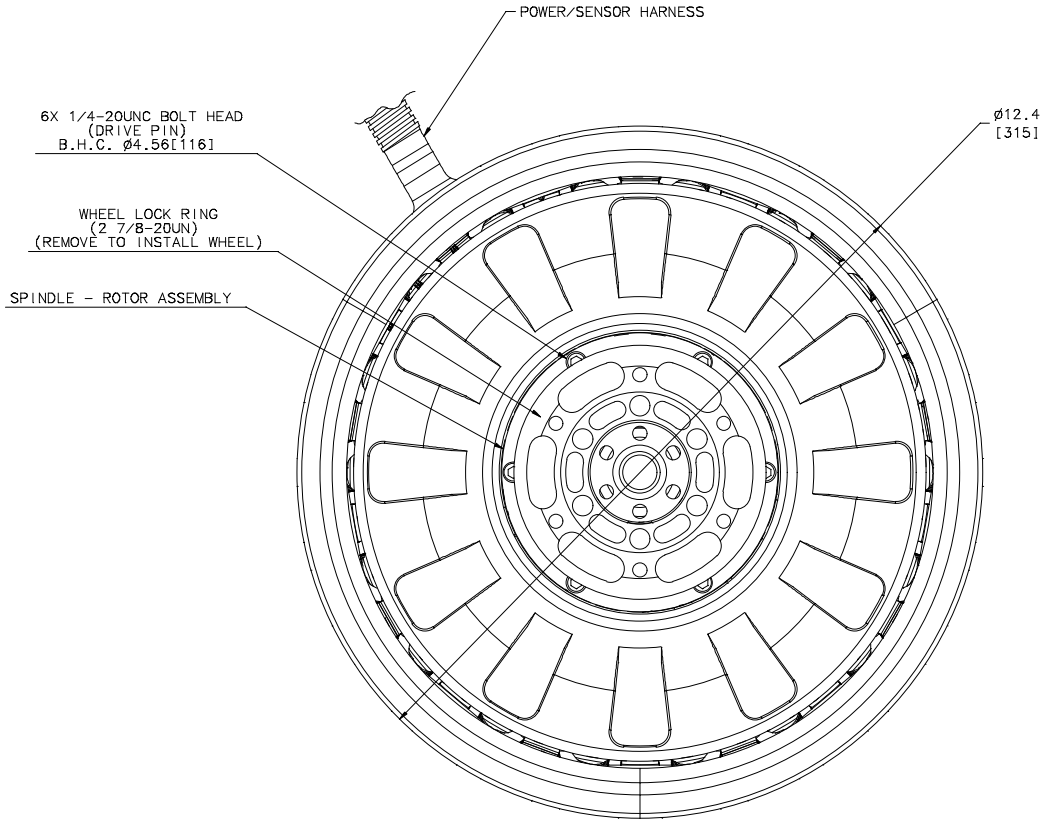
New Generation Motors Corporation
44645 Guilford Drive, Suite 201
Ashburn, Virginia 20147
(703) 858-0036 (Voice)
(703) 858-0602 (Fax)
Email: info@ngmcorp.com



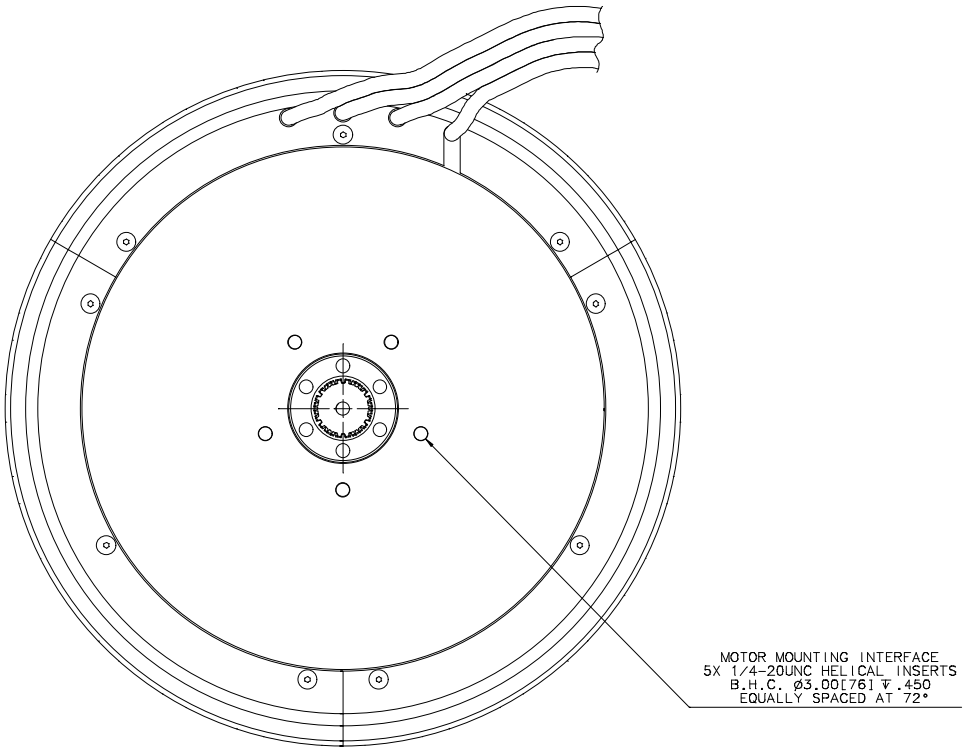


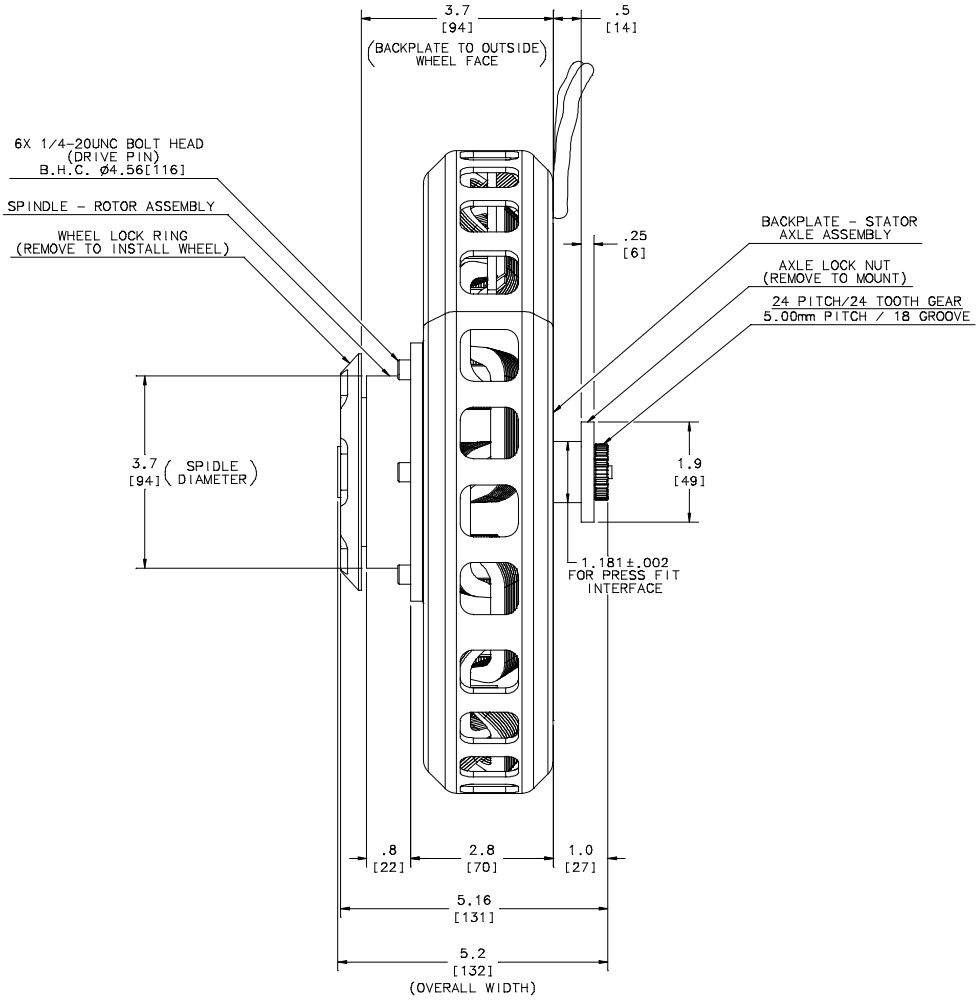
 Specifications Subject to Change Without Notice





 Specifications Subject to Change Without Notice





 Specifications Subject to Change Without Notice

NEW GENERATION MOTORS CORPORATION

Washington, D.C. U.S.A

NGM-EV-C200 series Controller
OPERATING MANUAL
Version 1.10D

Table of Contents

Appendices

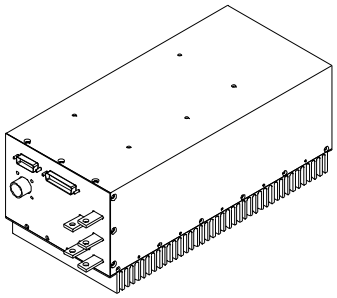
1. Specifications and features

The NGM-EV-C200-XX2 controller can power and control 3-phase DC brushless electric motors with advanced capabilities and superior efficiency. The controller has programmable logic to optimize and match it with nearly any 3-phase brushless permanent magnet DC motor. Basic features include:

◊ Selectable speed or torque control

◊ Selectable serial or discrete control interface

◊ Motor Current Limiting (MCL) logic senses battery voltage and motor & controller temperatures to limit current input or output



ISOMETRIC VIEW
(REFERENCE ONLY)

◊ Internal power supply for cooling fans activated by the controller's on-board thermal sensors for full thermal and power optimization

Specifications (w/o fans & connectors)

	EV-C200-042	**EVC-C200-092**
Peak current (amps)	260	150
Nominal bus voltages (volts)	42-54	66-108
Maximum operating voltage (volts)	68	135
Minimum operating voltage (volts)	30	50
Maximum voltage (volts)	75	160
Input capacitance (*mf*)	39,600	12,000
Height (in.)	5.29	5.29
Width (in.)	6.13	6.13
Length (in.)	13.06	13.06
Weight (lbs.)	10.8	10.8

Key features of the EV-C200-XX2 controller:

- ❋ Ultra high efficiency
- ❋ Compact design
- ❋ Synchronous switching
- ❋ Hysteresis control
- ❋ Regenerative braking
- ❋ Torque control
- ❋ Speed control
- ❋ Variable fan speed
- ❋ Reverse and brake light control
- ❋ Full I/O isolation
- ❋ Digital and analog inputs mutually isolated
- ❋ Isolated speed pulse output
- ❋ Configurable analog inputs
- ❋ Multiple hall sensor placement recognition
- ❋ Active discharge circuit
- ❋ Built in safety features
  - Continuous self diagnostics
  - FET drive under voltage lockout
  - Extreme over/under voltage protection
  - Motor interface connection verification
  - Phase lead connection verification
  - Thermal limiting protection
  - Over- and under- voltage current limiting with soft shutdown
  - Abrupt start-up inhibition
  - Stator short detection
- ❋ Data available
  - Voltage measurement
  - Speed measurement
  - Motor temperature sense
  - PWM frequency measurement
  - Logic supply current measurement
  - Drive state
  - Heatsink Temperature sense
  - Logic supply current measurement fault

Front panel interface information of the EV-C200-XX2 controller

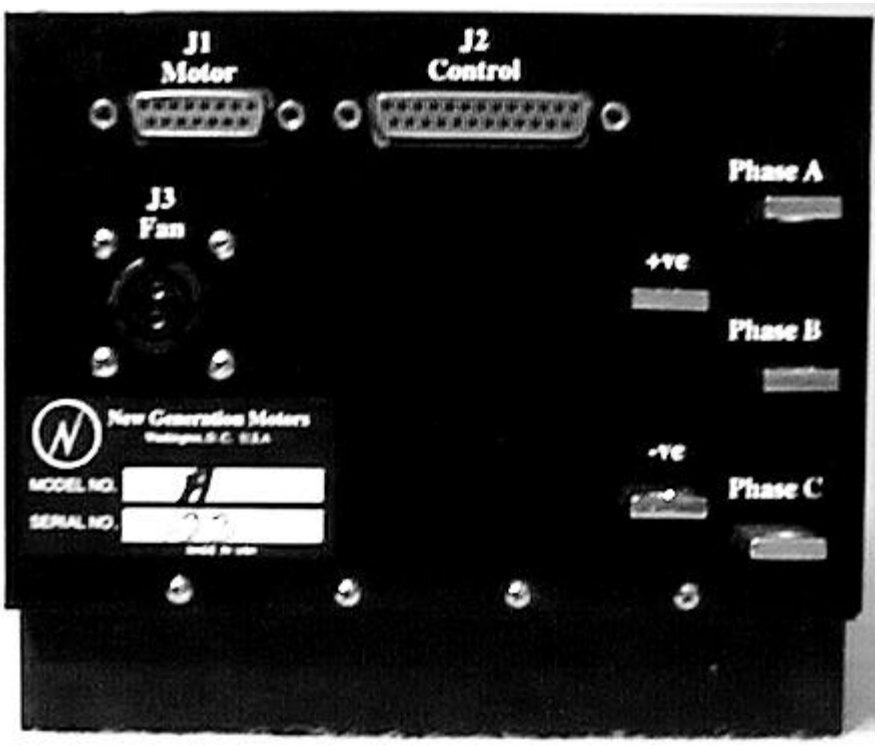


Fig. 1

Controller connections:

J1 – Motor communication link,
15 pin female D-Sub. connector

J2 – Control signals (vehicle)
25 pin female D-Sub. connector

J3 – Fan power for cooling
AMP Series 1 CPC 11-4, reversed sex
(*mating connector provided*)

+ve & -ve – Positive & negative power
bus bar with ¼ in. diameter through hole

Phase A,B,C – Phase lead connections for the motor
bus bar with ¼ in. diameter through hole

2. Parts list:

Qty.	Item
1	NGM-EV-C200-092 controller
1	NGM-EV-C200-092 controller Manual
1	CPC Series 1 11-4 plug
2	Series 1 pins
2	Muffin fans
5	¼ in. 20 UNC low head bolts
10	¼ in. flat washers
5	¼ in. 20 UNC narrow lock nuts
5	Rubber boots

NOTE: If you have not received all of the above items, please contact NGM.

3. Mechanical installation

3.1 Physical mounting

The Controller should be mounted by a method that minimizes the vibration and protects it from the elements during operation. High impact loads or excessive moisture and dirt could shorten the life span of the controller. There are several 4-40 UNC screw holes on the side of the controller that may be used for mounting. *Do not remove* any of the existing screws.

There are five types of connections that must be performed before operation of the controller:

- motor phase
- motor sense
- control
- fan power
- power

It is recommended that they be performed in the order as listed.

> ***Safety* Note**: The controller can retain a charge due to its high capacitance. Check the voltage before servicing the controller. **DO NOT** short the positive and negative buses together

.

3.2 Motor phase connection

This unit has three phase bus bars located on the right hand side; phase A, phase B and phase C. These phases must be properly connected to the corresponding phases of the motor. These connections must be made with *no less than* AWG 6 gage (4.11 mm) wire, although AWG 4 (5.18mm) is preferred. The connections can be made using properly sized ring terminals for the corresponding wire width and inner diameter of 0.25 in. Low head bolts, ¼ in. UNC no longer than 0.625in. should be used. They must be securely fastened with lock nuts and washers. Rubber boots should then be placed over each connection point to ensure no shorts between phases (a set of hardware is provided). *Visually check the spacing between connections and ensure the leads can not be rotated.* There should be a minimum of 3/16in. between connection points. Great care should be taken in applying proper strain relief for these cables. Additionally, ensure there exists enough slack in the cables for movement, especially for those connected to “in the wheel” motors.

In combination with NGM-SC-M100 & NGM-SC-M150 motors, RED corresponds to Phase A, GREEN to Phase B and BLACK to Phase C.

3.3 Motor sense connection

The motor sense connection requires a 15 pin D-sub male to be inserted into *J1* on the front of the controller and secured tightly. Take care to strain-relieve this cable properly on both ends to prevent any damage. (See Appendix A for pin out information)

Most NGM motors have a pre-installed cable for connection to the controller. However the NGM-SC-M100 motor, requires the *rotor retrofit package* to have been installed. For further information, contact NGM. Once the retrofit package is installed, the connection is similar to the others.

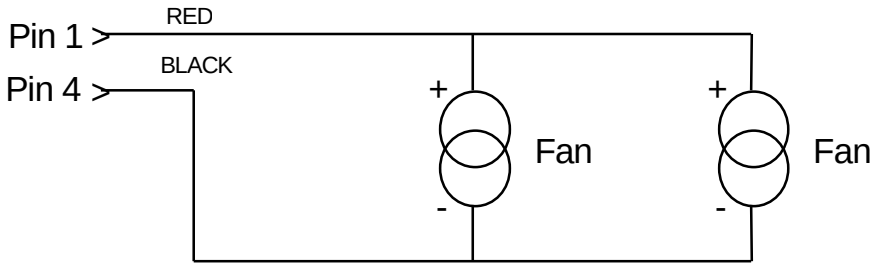
NOTE: *The retrofit package only allows use of the Hall and temperature sensors.*

3.4 Control input connection

The control cable must be plugged into *J2*, a DB25F connector. See appendix A for pin information.

3.5 Fan connection

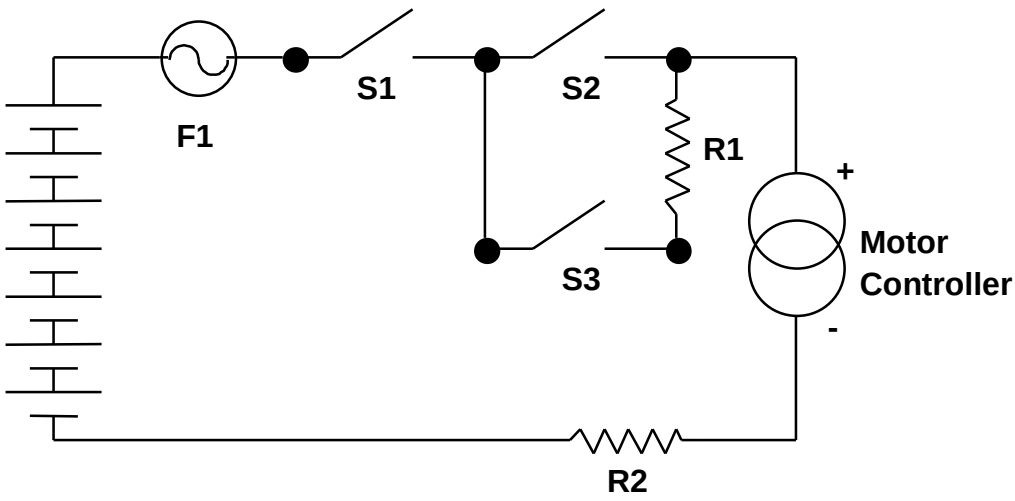
Fan power should be connected to *J3*. A series 1 CPC Amp 11-4 plug and two pins are provided with the controller. Splice the ground of each fan wire (Black) into one single wire long enough to reach the front panel of the controller. Do the same with the positive (Red) wires of each fan. Crimp the pins (CPC, series 1) on to the end of the positive and negative leads of this cable. The positive must be placed into position 1 of the plug and the negative into position 4 (See Fig. 3). Then mate the plug to *J3* on the controller's front panel.



Fan power circuit (Fig. 3)

3.6 Power connection

A pre-charge circuit *(see Fig. 4) must* be used to connect the motor controller to the power system. Resistor R1 and switch S3 form a "pre-charge" for the motor controller. The input capacitance of the controller is very high, large in-rush currents will eventually destroy the controller and switch S2. R1 should have a resistance such that the current through it at turn-on is at most 30A. Resistor R2 is an optional high current shunt for measuring the motor controller current. The DC ratings of all components must exceed the maximum bus voltage.



Pre-Charge circuit schematic (Fig. 4)

Low head bolts ¼ in. UNC with a lock nut and washer (provided) should be used to connect to the positive and negative posts of the controller. A *minimum* of AWG 6 gage (4.11 mm) or larger should be used (AWG 4 (5.18mm) preferred). Visually check the spacing between connections to ensure that the leads can not be rotated. After connection, each post should have a rubber boot covering it. Take care to strain-relieve each wire properly to ensure that no damage is done by the force on the connections.

4. Communication formats

The motor controller can be controlled by either discrete or serial communication. All communication is conducted through connector *J2* on the front panel. See Appendix A for pin out diagram. From the factory the controller's default setting is "discrete interface."

4.1 Via Discrete interface

The term "discrete interface" refers to all of the I/O except for the serial interface lines.

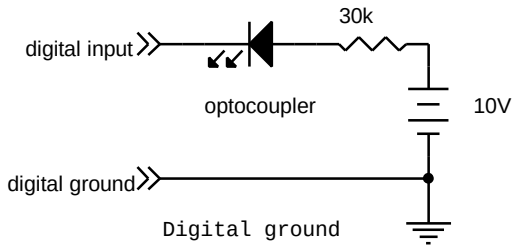
4.1.1 Digital inputs

Forward/Reverse**:** The *for/rev* input and its corresponding *gnd,* use pins 5 and 18, respectively, on *J2*. Forward corresponds to open circuit and reverse to closed. It is recommended that the direction signal be wired directly to a switch for maximum safety and reliability. The [CG_INVERTDIR] register is used to define whether forward is clockwise rotation or counter-clockwise rotation. The direction may differ depending upon the type of motor being used. When [CG_INVERTDIR] is False, forward is defined as a counterclockwise rotation when looking at the rotor of the NGM single-stator motors.

Enable: The *enable* input signal (pin 3 on *J2*) must be connected to *gnd* (pin 16, on *J2*) for the controller to enable. An open circuit immediately disables all torque production, and reduces the controller's quiescent power consumption. As part of an extra safety feature the controller will not enable unless there is a closed circuit between pins 2 and 10 on *J1* of the controller. This is normally performed inside of the motor acting as a motor sense circuit.

Throttle enable: The *threnable* signal (pin 4 on *J2*) must be connected to *gnd* (pin 17 on *J2*) for the controller to produce accelerating torque. When open-circuited, the maximum throttle current is set to zero, but the controller can still operate in regen. It is suggested that this input be wired to a switch on the brake pedal.

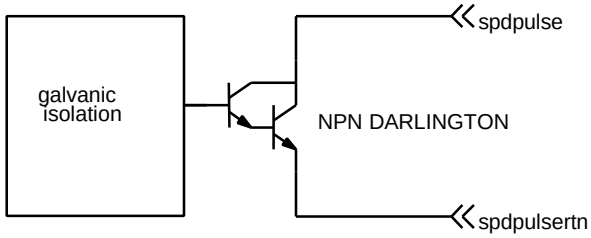
Electrical equivalent of *direction*, *enable,* and *throttle enable* inputs



(Fig. 5)

4.1.2 Digital outputs

Speed pulse: Speed *pulse* (pins 6 and 19 on *J2*) is an isolated open-collector, low drive, pulse stream output that is proportional to the commutation rate, and thus the rotational velocity. The output changes state every two consecutive commutations (i.e. never after forward and backward movement), producing a 50% duty cycle. The electrical equivalent of the speed pulse output is shown in Fig. 6.



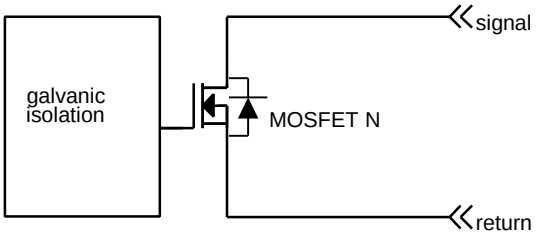
Speed pulse schematic (Fig. 6)

The output frequency f_{out} equals (3*P)/4 * f_{motor}, where P equals the motor pole count and f_{motor} equals the motor's revolutions per second.

Reverse detection: *Reverse* (pins 8 and 21 on *J2)* can be used as an activating switch that corresponds to the controller operating in the reverse direction. It is on (conducting) when the input direction is reverse (i.e. when [IN_FORWARD] = 0), regardless of the actual direction the motor is spinning or the state of [SV_FORWARD] (the direction in which the controller is operating).

Regenerative braking detection*: Brake* (pins 7 and 20 on *J2*) can be used as an activating switch that corresponds to the controller when in a "braking" mode. It is on (conducting) when [IN_DESIREDPHASEI] is negative, even if the drive state is not in regen mode [DS_RGN]. This includes any case where [SV_MAXRGNI] is zero.

The electrical equivalent of the reverse and brake outputs is:

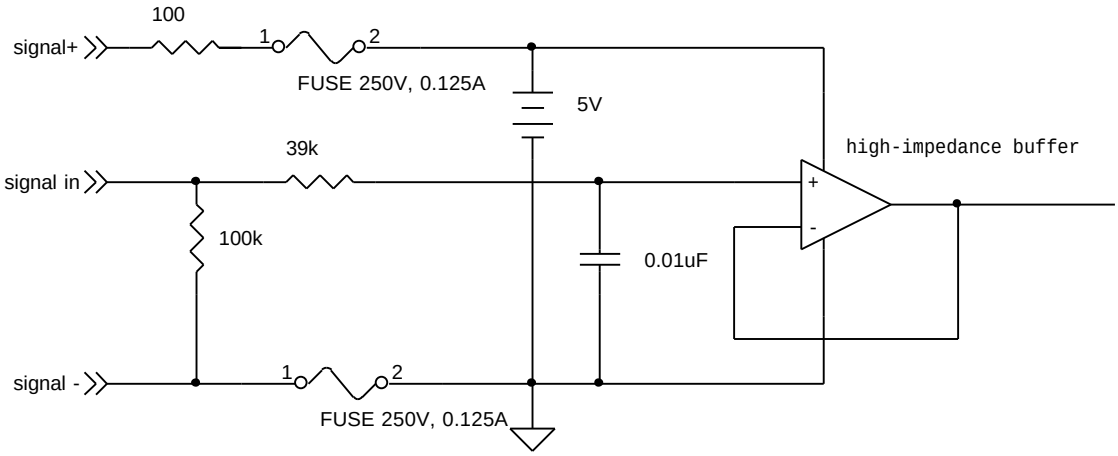


(Fig. 6)

NOTE: The D connector *reverse* and *brake* pins can sustain a maximum of 20V and 100mA.

4.1.3 Analog inputs

There are two analog inputs, *throttle* and *regen*. Each input has an associated +5V reference and ground, labeled sig+ and sig- respectively, where "sig" is either throttle or regen. The signals can be coupled with potentiometers, having a resistance at 4k to 20k Ohms, to create a scaling effect from 0-5 V max. Each voltage on the scale then equates to a desired signal for the given range. The signals are user defined including the gains as well as maximum and minimum desired values of the scale. An electrical equivalent of the circuit is shown in Fig. 7.



Throttle & regen internal circuit schematic (Fig.7)

Filtering

There are two types of filtering on the analog inputs. First, the maximum rate of change (in bits/sample) is limited to 32 for [AM_THR] and [AM_RGN]. Second, all of the analog inputs are capable of exponential filtering. The level of filtering is set by [RT_(INPUT NAME)], which may vary from 0 (no filtering) to 4 (maximum filtering, slowest response). The formulas for these are:

0: Value = newvalue
1: Value = ½(oldvalue) + ½(newvalue)
2: Value = ¾(oldvalue) + ¼(newvalue)
3: Value = 7/8(oldvalue) + 1/8(newvalue)
4: Value = 15/16(oldvalue) + 1/16(newvalue)

The analog inputs are sampled 10 times per second.

Deadbands

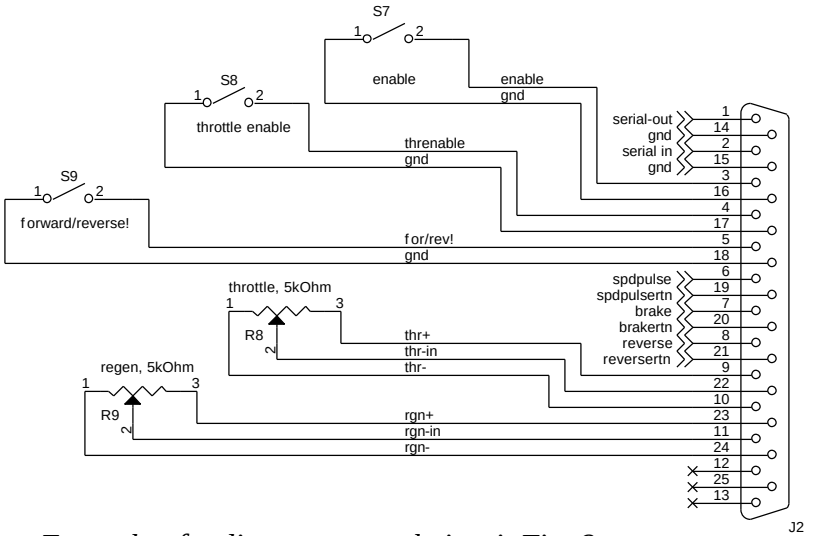
The deadbands are similar to offsets, but any input less than the deadband is set to zero. The deadband value range is from 0 to 255, corresponding to the controller's eight-bit A-D converter.

The following pseudo-code illustrates how [AM_THR] and [AM_RGN] are computed:

```
X = Output from A to D converter // 0-255 counts

if ((X – oldXvalue) > 32 counts) then
        X = oldXvalue + 32
elseif ((X – oldXvalue) < 32 counts) then
        X = oldXvalue – 32

if (X < CG_XDEADBAND) then
        X = 0

oldXvalue = X

AI_X = (AI_X * (2^RT_X – 1) + X) / 2^RT_X

if (X equals 0) and ((AI_X < CG_XDEADBAND * 16)) then
        AI_X = 0
```

Enables

DF_SPEEDCONTROL	Sets speed control as default by setting [SV_SPEEDCONTROL] at power-up.
CG_ENDISCRETE_THR	Enables the discrete throttle & regen inputs for either torque or speed control. Otherwise, the serial input registers are used.
CG_ENDISCRETE_DIR	When set, the discrete direction input is used, otherwise the serial input register is used.
CG_ENDISCRETE_THRENABLE	When set, both the discrete and serial throttle enable inputs are used. Otherwise, only the serial input register is used.



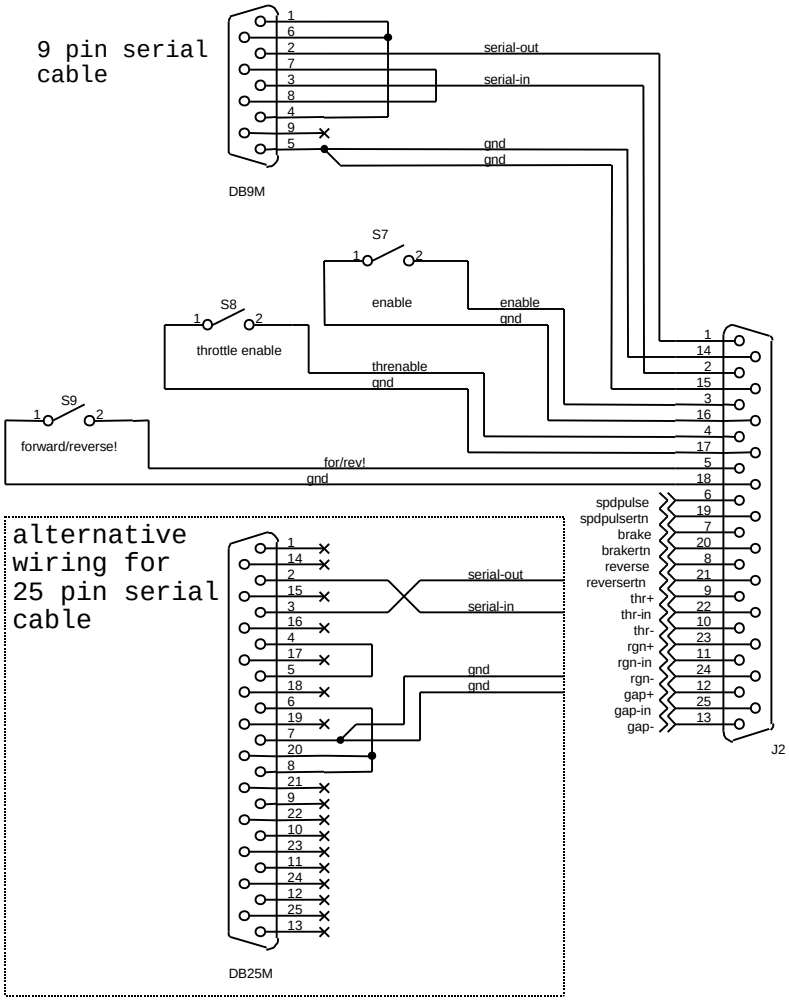
Example of a discrete control circuit Fig. 8

4.2 Via Serial interface

Serial I/O is performed using an RS-232 three-wire interface. All communication is performed through pins (1, 14) and (2,15) serial out and in, respectively. The serial interface serves three functions:

1). Provides control inputs to the controller
2). Receives measurements and status information from the controller
3). Configures the controller for specific applications and settings

A sample schematic of wiring the controller to a 9 pin serial port or an alternative 25 pin.



4.2.1 RS-232 settings & syntax

Default serial settings:			
Flow Control	None *fixed*	Baud rate	9600 *(factory default)*
Data Bits	8 *fixed*	Echo	enabled *(factory default)*
Stop bits	1 *fixed*		

There are three types of serial input messages, commands, queries, and assignments. All are terminated with either a carriage return or carriage return and line feed combination. They take the following forms:

Command:
Commands are used for setting forward or reverse, enabling, entering and exiting program mode, and similar operations.

XX! (CR)

Where **XX** is the hexadecimal command number and **(CR)** is a carriage return character.

The controller replies with either **Ok** (**CR)** or an error message.

Query:
Queries may be made of either RAM variables (inputs, measurements and the like) or EEPROM variables (configuration values). The operation character determines which memory area is accessed as follows:

From RAM:

XX? (CR)

From EEPROM:

XX>(CR)

In both cases **XX** is the memory location to be returned. The controller replies with a text string of the decimal value of the register.

Assignment:
Assignments may also be of either RAM or EEPROM registers. Note that it is necessary to enter the program mode to make assignments to EEPROM register. This allows the controller the time to program the memory location and prevents accidental changes.

To RAM:

XX=#(CR)

To EEPROM:

XX<#(CR)

In the example, # stands for the text string of the decimal value to be assigned. For unsigned long registers, this string may be ten digits long. Note that for the most, part range checking is not performed. The controller responds with either **Ok (CR)** or an error message.

4.2.2 Serial commands

Command	Name	Description
00!	Enable	Clears [CB_DISABLE]
01!	Disable	Sets [CB_DISABLE] disabling the controllers output
02!	Throttle enable	Sets [CB_THRENABLE]
03!	Throttle disable	Clears [CB_THRENABLE], forcing throttle input to zero
04!	Reverse	If [CG_ENDISCRETE_DIR] is false, clears [IN_FORWARD]
05!	Forward	If [CG_ENDISCRETE_DIR] is false, sets [IN_FORWARD]
06!	Torque control	Clears [SV_SPEEDCONTROL]
07!	Speed control	Sets [SV_SPEEDCONTROL]
08!	Coast	Forces desired current to zero Sets [IN_DESIREDPHASEI] to zero
0A!	program	Sets controller in PROGRAM drive state
0B!	operate	Exits program drivestates
FA!	Reset microcontroller	Performs a hard reset (similar to a power cycle)

4.2.3 EEPROM Registers

00_H	CG_BAUDRATE	I/O baud rate
90	CG_ECHO	When true, echo characters as they are received
91	CG_TEXTERRORS	When true, send text messages for errors, else send two digit codes. *See Appendix E*
92	CG_LINEFEEDS	When true, use CR-LF combinations at end of lines
93	CG_MAXSCIIDLE	Maximum idle time for serial interface watchdog fault in tenths of a second, 0 disables

5. Control Modes

5.1 Overview

There are two modes that can control the output of the controller; torque and speed control. The *factory default* is set to torque control.

Torque control is the method of controlling the output phase current directly. The phase current is roughly proportional to the torque output of the motor. The relationship will differ from motor to motor and must be determined by the user, especially when the motor's parameters are constantly changing, e.g. the air gap in the NGM-SC motors.

Speed control varies the phase current as a function of the difference between the input desired speed and the actual speed. At the most basic level, it is controlling the motor's electrical frequency, which is directly proportional to the rotational velocity.

5.2 Torque control:

RAM Registers:

10	SV_TARGETPHASEI	Target current (dA)
13	SV_MAXTHRI	Maximum throttle current (dA)
14	SV_MAXPHASEIRGN	Maximum regen current (dA)
1A	SV_HYSTERESIS	Hysteresis level (counts)
61	IN_DESIREDPHASEI	Phase current in (dA)
96	SV_DRIVESTATE	Operating status
A9	SV_STEP	Current commutation step

The controller does not implement torque control per-se. Instead, it operates by controlling the motor phase current that is proportional to the torque. This proportionality is a function of the motor coupling (which varies in adjustable gap motors). The input to the phase current control function is [IN_DESIREDPHASEI]. This value can be set by either the discrete interface or the serial interface. The maximum phase current allowed is determined by the minimum value of the following functions:

1. Motor thermal limiting *(user defined)*
2. Controller thermal limiting *(factory set)*
3. Low supply voltage limit *(user defined)*
4. High supply voltage limit *(user defined)*
5. Soft start limits *(user defined)*
6. Base maximum phase current (throttling or regen). *(user defined)*

In addition, the maximum throttling current is zero when [IN_THRENABLE] is FALSE (set by [CB_THRENABLE] and/or [BI_THRENABLE]). (See Section 7 MCL logic for more information).

5.2.1 Via discrete interface

In the discrete torque control, the desired phase current (which is proportional to torque) is determined by the difference between the throttle and regen analog inputs. The inputs could be references across potentiometers. When the difference is greater than zero, driving torque is produced. When the difference is less than zero, regen is applied. When the difference is equal to zero, the motor coasts. These inputs have independent deadband and scale settings. The deadbands are similar to offsets, but any input less than the deadband is set to zero.

To Engage:

[CG_ENDISCRETE_THR] set to TRUE
[SV_SPEEDCONTROL] set to FALSE

Associated variables:

Desired phase current	= [IN_DESIREDPHASEI]
Analog throttle input	= [AM_THR], on 0-4080 (counts*16) scale

Analog regen input	= [AM_RGN], on 0-4080 (counts*16) scale
Deadband values:	range is 0-255 counts
Throttle	= [CG_THRDEADBAND]
Regen	= [CG_RGNDEADBAND]
Scales:	
Torque	= [CG_SCTHR_TORQUE]
Regen	= [CG_SCRGN_TORQUE]

Calculation:

The controller uses the following formula to calculate the desired phase current:

IN_DESIREDPHASEI = (AM_THR * CG_SCTHR_TORQUE – AM_RGN * CG_SCRGN_TORQUE) / 256
Where;

AM_THR = [(input voltage)* 51 – CG_THRDEADBAND]*16
AM_RGN = [(input voltage)* 51 – CG_RGNDEADBAND]*16

Example: IF CG_THRDEADBAND = 127
CG_SCTHR_TORQUE = 100
AM_RGN = 0

AM_THR = (5 * 51 – 127) * 16 = 2048
(V * counts/V – counts) * count/count

IN_DESIREDPHASEI = (2048 * 100 – (0 * CG_TOAMPSRGN) / 256= 800 dA

The configuration registers are factory set to provide full-scale output phase current over an input voltage range of 0.12 to 5.0 V. If desired, these registers may be altered to accommodate a narrower voltage range. Reducing the voltage range in this fashion also reduces the resolution. Therefore, it is recommended that the input voltage range be at least 1.2 V (1/4 of full range).

Calculation: To find scale and deadband settings:

Desired phase current: I (in deci-A)
Throttle input voltage:
Low Lt, recommend a min value of 0.12
High Ht
regen input voltage:
Low Lr, recommend a min value of 0.12
High Hr

CG_THRDEADBAND = 51 * Lt

CG_RGNDEADBAND = 51 * Lr

CG_SCTHR_TORQUE = (I / 255) * (Ht – Lt) / 5 * 16 * 1.05
= (16.8/1275)* I * (Ht – Lt)

CG_SCRGN_TORQUE = (I / 255) (Hr – Lr) / 5 * 16 * 1.05
=(16.8/1275)* I * (Hr – Lr)

These settings will allow full regen and throttle current when the two controls are at full travel. It may be desirable to double [CG_SCTHR_TORQUE] and [CG_SCRGN_TORQUE] to reach full throttle or full regen when the difference between the two controls is only half travel.

5.2.2 Via serial interface

In the serial torque control mode, the desired phase current value is set by assigning a value to [SI_DESIREDPHASEI] in deci-amps. A negative phase current corresponds to regen.

To Engage:

[CG_ENDISCRETE_THR]	set to FALSE
[SV_SPEEDCONTROL]	set to FALSE

Action:
Assign Desired phase current to [SI_DESIREDPHASEI] (deci-Amps)

5.3 Speed Control

The speed control function operates as an outer control loop by controlling [IN_DESIREDPHASEI]. It is implemented with a P^2I control loop with the following parameters:

RAM REGISTERS	NAME	DESCRIPTION
03	SI_KP	Coefficient of the proportional error squared term
04	SI_Ki	Coefficient of the integrated error term
07	SI_MAXSPEEDERROR	Limits the maximum speed error, dampens response to large step changes in the desired speed.
05	SI_Kt	Coefficient multiplied with the phase current and subtracted from the speed error. When non-zero, it allows speed to be decreased at high load, and increased at negative loads (such as a steep hill).

All speed control parameters are stored in RAM registers with power-up values stored in EEPROM. These coefficients are stored in RAM to allow real time adjustment to find the best performance in a given application. The values can then be stored in the EEPROM registers as defaults.

The integrated error term is subjected to the same MCL logic applied to [IN_DESIREDPHASEI].

Speed control is set when [SV_SPEEDCONTROL] is TRUE. The default value is FALSE. "Speed" is in the units of deci-Hz and is the measurement of the electrical frequency. The conversion to RPM is ([AI_SPEED] * 12) / P. P is equal to the motor pole count. If the motor being used had 12 poles, [AI_SPEED] would equal RPM.

The desired speed used in the *PI* control loop can be made a function of the applied torque with [SI_Kt]([CG_Kt]). Basically:

Speed error = Input speed – Actual speed – [SV_PHASEI]*[SI_Kt]

5.3.1 Via discrete interface:

The throttle input sets the desired speed using a 0-5V scale with 0 V =0 speed and 5V equal to the maximum full-scale value allowed by the controller's configuration. The regen input sets the maximum braking torque using a 0-5V scale also.

To Engage:

[CG_ENDISCRETE_THR]	set to TRUE
[SV_SPEEDCONTROL]	set to TRUE

Associated registers:

Desired speed (user def. Units)	= [IN_DESIREDPHASEI]
Analog throttle input voltage	= [AM_THR], on 0-5V scale
Analog regen input voltage	= [AM_RGN], on 0-5V scale
Deadband values:	range is 0-255 (counts)
Throttle	= [CG_THRDEADBAND]
Regen	= [CG_RGNDEADBAND]
Scales:	
Speed	= [CG_SCTHR_SPEED] (<speed unit> / 16 * (counts))
Regen	= [CG_SCRGN_TORQUE]

Calculation: To find scale and deadband settings

Maximum desired speed: S
Maximum regen phase: I (in deci-A)
Throttle input voltage
Low Lt, recommended a minimum value of 0.12 V
High Ht
regen input voltage
Low Lr, recommended a minimum value of 0.12 V
High Hr

CG_THRDEADBAND = 51 * Lt

CG_RGNDEADBAND = 51 * Lr

CG_K_AD_SPD = (S / 255) * (Ht – Lt) / 5 * 16 * 1.05
= (16.8/1275) * S * (Ht – Lt)

CG_SCRGN_TORQUE = (I / 255)* (Hr – Lr) / 5 * 16 * 1.05
= (16.8/1275) * I * (Hr – Lr)

The input speed is stored in [SI_DESIREDSPEEED].

5.3.2 Via serial interface

In the serial mode, the speed is set by assigning a value to [SI_DESIREDSPEED].

To engage:
[CG_ENDISCRETE_THR] set to FALSE
[SV_SPEEDCONTROL] set to TRUE

Action:
Assign desired value to [SI_DESIREDSPEED].

6. Controller fault detection

The controller has continuous self-diagnostics. If a fault occurs, it is recorded into its corresponding register and the controller reacts accordingly. The faults are divided into four types, corresponding with four 8-bit registers, [SV_FAULT1] through [SV_FAULT4].

Type 1: Faults that immediately disable the controller and prevent operation
Type 2: Sensor problems
Type 3: Warnings
Type 4: Conditions that lead to a reduction in the output torque.

Type 1 faults disable the controller, with the effect of clearing the fault registers. There is a fifth register, [SV_FAULT1LATCH], which corresponds one-to-one with [SV_FAULT1]. Bits in [SV_FAULT1LATCH] are set whenever a fault occurs. They are only cleared when the controller is re-enabled. In this way, [SV_FAULT1LATCH] stores the condition that caused a shutdown.

The register stores each fault in its own distinct bit location. To read which fault has occurred, use a mask to determine which bit in the register is set to one.

EXAMPLE:
8 bit REGISTER

$$SV\_FAULT1 = 02_H = 0000\ 0010_B$$

If [SV_FAULT1] was read and it showed this. The fault would be Type 1 and correspond to [FA1_OVERVOLT], supply voltage is greater than [CG_ABSMAXV]

Type 1 faults:

Mask	Fault	Description
1_H	FA1_UNDERVOLT	Supply voltage is less than[CG_ABSMINV].
2	FA1_OVERVOLT	Supply voltage is greater than [CG_ABSMAXV].
4	FA1_NOFETDR	Supply voltage to gate drives low, internal fault or low supply voltage spike.
8	FA1_NOPHASELEADS	Most likely a phase is not connected
10	FA1_INVALIDHALLS	A HALL effect input is invalid. Most likely a cable is not connected
20	FA1_LOSTCOMM	Serial I/O watchdog has tripped due to inactivity on serial input.

Type 2 faults:

Mask	Fault	Description
1_H	FA2_MOTORT	Signal from motor temp sensor is < -50C or > 150C.
2	FA2_HEATSINKT	Signal from heatsink temp sensor is < -50C or > 150C.
4	FA2_SUPPLYI	Logic supply current measurement is less than [CG_MINSUPPLYI].

Type 3 faults:

Mask	Fault	Description
1_H	FA3_FAN	The logic supply current is outside the fans MINSUPPLYI and MAXSUPPLYI range. The fan may be disconnected or jammed
2	FA3_STATORSHORT	A stator short to ground or a phase has been detected.
4	FA3_MAXTORQUE	Motor has reached maximum throttle or regen current for the current speed.
10	FA3_SOFTSTART	Controller is soft-starting because [IN_DESIREDPHASEI] was != 0 when controller was enabled.
20	FA3_OBDIRBACKWARDS	The observed direction of rotation is opposite the input direction. Controller is coasting.

The Fault 4 register

Mask	Fault	Description
1_H	FA4_MOTORTLIM	Current limit is due to motor temperature.
2	FA4_HEATSINKTLIM	Current limit is due to heatsink temperature.
4	FA4_UNDERVOLT	Supply voltage being less than [CG_MINVGUARD].
8	FA4_OVERVOLT	Supply voltage being greater than [CG_MAXVGUARD].
10	FA4_ABSLIM	Desired current is greater than either [CG_MAXTHRI] or [CG_MAXRGNI].
20	FA4_SOFTLIMIT	Desired current is greater than either [SV_THRPHASEILIM] or [CG_RGNIPHASEILIM].
40	FA4_THRDISABLED	Throttle current is zero because throttle enable input (either discrete or serial) is FALSE.
80	FA4_BRAKEPHASEILIM	When in discrete speed control, target regen current is greater than limit set by regen input.

7. Motor Current Limiting logic (MCL)

MCL can cause the controller to shut down or limit its output. The parameters that can activate the MCL logic to take place are as follows:

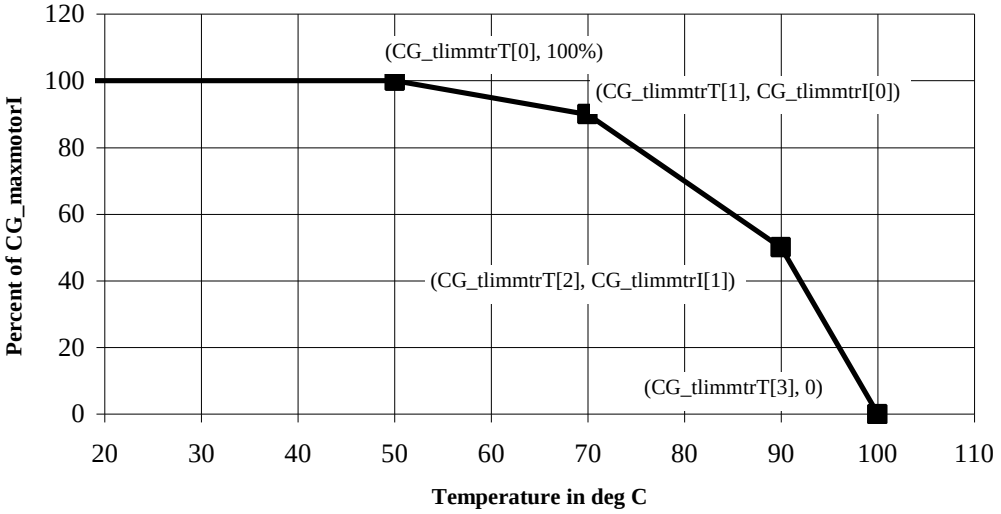
1. Motor temperature
2. Controller temperature
3. Under & over voltage
4. Absolute maximum limits
5. Soft starting
6. Soft maximum limits
7. Throttle enable
8. Discrete Regen limit
9. Observed Direction backwards

These parameters set the maximum phase current levels in throttle and regen according to some pre-set algorithms. Should one of these parameters go out of its predetermined range, causing the desired motor phase current to be greater than the calculated maximum, the MCL logic would take over and limit the controller's output until the point of shut-down.

7.1 Motor thermal limits

Thermal protection is achieved by limiting the phase current as a function of the estimated motor temperature, [AM_MOTORTEST]. The absolute maximum motor current is [CG_MAXMOTORI] and all thermal derating is in proportion to this value. Note that this is completely independent of the motor controller limit. Thus, a 300A capable controller can be safely used with a 100A maximum motor and vice-versa.

The thermal derating is based on a piecewise-linear function as shown below. The values for [CG_TLIMMTRT] are stored in deci-degrees C (consistent with [AM_MOTORTEST]). The [CG_TLIMMTRI] values are stored with an implied denominator of 256, such that 256 = 100% of [CG_MAXMOTORI], 128 = 50% of [CG_MAXMOTORI], etc…



7.2 Controller thermal limits

If the controller reaches a temperature above its set limit it will shut down. Before reaching this temperature, the controller performs current limiting operations to reduce the amount of heat generated internally.

7.3 Under and over voltage

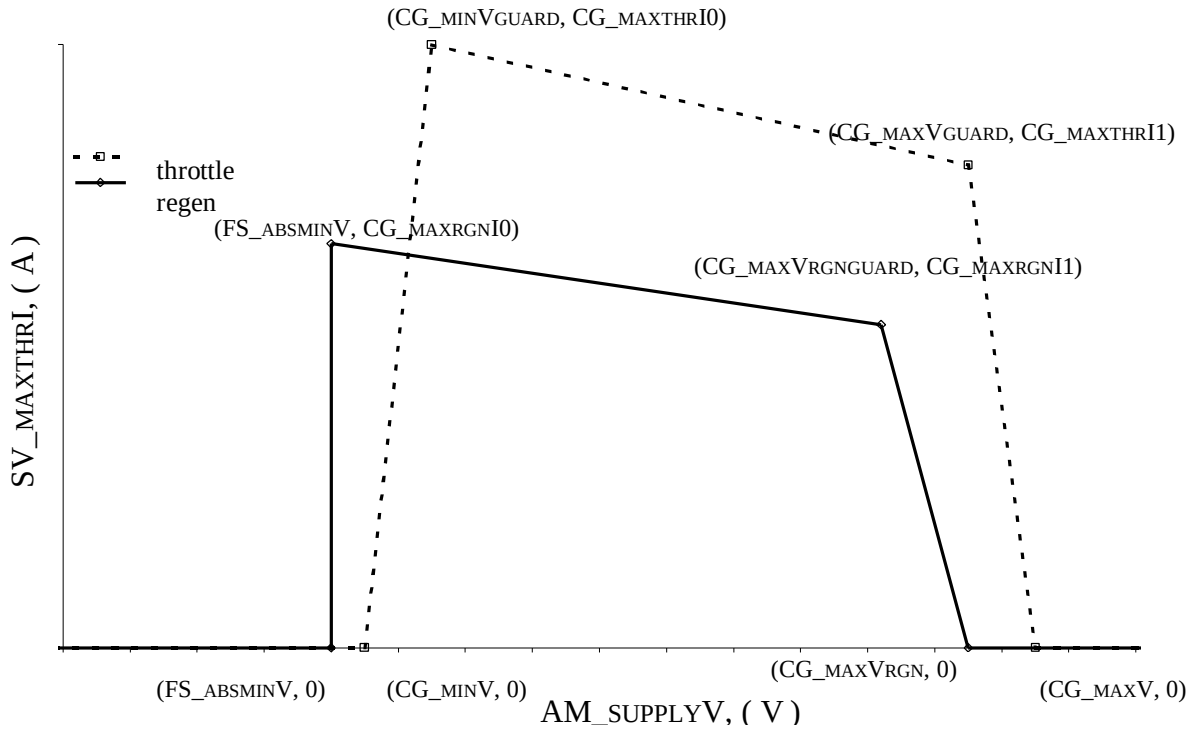
When [AM_SUPPLYV] is less than [CG_MINVGUARD], the maximum throttle current is linearly derated from [CG_MAXTHRI0] at [CG_MINVGUARD] to zero at [CG_MINV] and below. When the controller is limiting the phase current, the resulting [FA4_UNDERVOLT] is set. Note that there is no under-voltage limiting of the regen current until the supply voltage falls to [FS_ABSMINV].

When [AM_SUPPLYV] is greater than [CG_MAXVGUARD], the maximum throttle current is linearly derated from [CG_MAXTHRI1] at [CG_MAXVGUARD] to zero at [CG_MAXV] and above. Similarly, when [AM_SUPPLYV] is greater than [CG_MAXVRGNGUARD], the maximum regen current is linearly derated from [CG_MAXRGNI1] at [CG_MAXVRGNGUARD] to zero at [CG_MAXVRGN] and above. As a result of the controller limiting the phase current, [FA4_OVERVOLT] is set.

7.4 Abslolute maximum

When [AM_SUPPLYV] is between [CG_MINVGUARD] and [CG_MAXVGUARD], the maximum throttle current is limited to a value between [CG_MAXTHRI0] and [CG_MAXTHRI1]. When [AM_SUPPLYV] is between [FS_ABSMINV] and [CG_MAXVGUARDRGN], the maximum regen current is limited to a value between [CG_MAXRGNI0] and [CG_MAXRGNI1]. When the controller is limiting the phase current, [FA4_ABSLIMIT] is set.

The following piece-wise linear graph shows the complete relationship between supply voltage and phase current for both throttle and regens.



7.5 Soft-start limit

When [IN_DESIREDPHASEI] is a non-zero and the controller transitions from disabled to enabled, a soft-start mechanism is employed. During this period, the actual phase current is a fraction of [IN_DESIREDPHASEI]. This fraction is increased linearly until it equals one. The length of this period is set by [CG_SOFTSTARTN] and equals $2^{CG\_softstartN}/62$ seconds. Thus for [CG_SOFTSTARTN] = 6, the soft start period is approximately 1 second. The maximum value for [CG_SOFTSTARTN] is 7.

7.6 Soft maximums (phase current adjustment)

The "soft maximums" are two serial input registers, [SI_THRILIMIT] and [SI_RGNILIMIT], that allow the maximum phase current to be adjusted "on-the-fly," if these limits are less than all other limits. This can be useful for supervisory control of the discrete throttle and regen inputs or for limiting the phase current while in speed control to prevent low efficiency accelerating and braking. When [SV_TARGETPHASEI] is limited by these constraints, the [FA4_SOFTLIMITS] bit is set in the [SV_FAULT4] register.

7.7 Throttle Enable

The state of the throttle enable [BI_THRENABLE] must register TRUE for the controller to produce any accelerating torque. If FALSE the controller sets the maximum throttling current to 0. The controller can still operate in regen mode. See sections 4.1.1 & 4.2.2

7.8 Discrete Regen limit

While operating in Discrete speed control, the controller will use the regen limit set by the Regen "pot". "pot" referring to the control method of the input signal to the controller.

7.9 Observed Direction backwards

Observed direction backwards is a Fault 3 offense. If the controller should detect that the direction of rotation of the motor is opposite of what is desired, it will go into a *coast* mode. While in coast mode the phase current is set to 0. The threshold is Factory set to 20 deci-hertz of reverse rotation. To translate this into RPM see section 5.3.

8. Controller Cooling

The NGM-EV-C200-XX2 motor controllers are supplied with two high efficiency muffin fans for cooling the controller. These should be hooked up at *all times* during operation of the controller. This will help the controller run cooler and more efficiently. The fans are powered and controlled by an internal temperature sensor built into the controller.

Note: If the controller notices excessive heat, it will slowly decrease its output to reduce heat generated. At this point the temperature would have to decrease before full operation can continue.

NGM Warranty

New Generation Motors Corporation warrants that its NGM-EV-C200 series motor controller will be free from defects in title, materials, and manufacturing workmanship for one (1) year. If an NGM-EV-C200 series motor controller is found to be defective, then, as your sole remedy and as the manufacturer's only obligation, New Generation Motors Corporation will repair or replace the product. This warranty is exclusive and is limited to the NGM-EV-C200 motor controller.

This warranty *shall not apply* to NGM-EV-C200 series motor controllers that have been subjected to abuse, misuse, abnormal electrical or environmental conditions, or any condition other than what can be considered normal use (including, and not limited to, opening of the controller for any purpose).

Warranty Disclaimers

New Generation Motors Corporation makes no other warranties, express, implied, or otherwise, regarding NGM-EV-C200 series motor controllers, and specifically disclaims any warranty for merchantability or fitness for a particular purpose.

The exclusion of implied warranties is not permitted in some States and countries thus exclusions specified herein may not apply to you. This warranty provides you with specific legal rights. There may be other rights that you have which vary from State to State.

Limitation of Liability

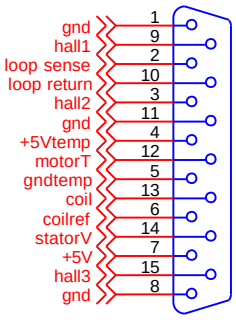
The liability of New Generation Motors Corporation arising from this warranty and sale shall be limited to the replacement of defective parts. In no event shall New Generation Motors Corporation be liable for costs of procurement of substitute products or services, or for any lost profits, or for any consequential, incidental, direct or indirect damages, however caused and on any theory of liability, arising from this warranty and sale. These limitations shall apply not withstanding any failure of essential purpose of any limited remedy.

Appendix A

J1 Pin Out

If another motor is being used, the following connections are possible. As a minimum the 3 hall sensors and the loop back of pins 2 and 10 must be connected for operation.

1. Ground of hall effect sensor for phase A
2. Cable connection sense loop
3. Hall effect signal of phase B
4. +5V for a temperature sensor
5. Ground for the temperature sensor
6. Reference for coil detection circuit
7. +5V for hall effect sensors
8. Ground of hall effect sensor for phase C
9. Hall effect signal of phase A
10. Cable connection sense loop
11. Ground of hall effect sensor for phase B
12. Signal from temperature sensor
13. Coil detection circuit
14. Stator voltage sense
15. Hall effect signal of phase

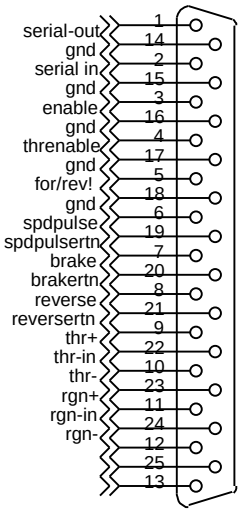


NOTE: Any attempt to adapt or modify signals could nullify any existing warranty. Please consult NGM prior to any such attempts.

J2 Pin Out explanation

PIN #

1. Serial out-The serial communication line for data leaving the controller.
2. Serial in- The serial communication line for data going into the controller.
3. Enable- When it is shorted to pin 16 the controller is *Enabled* for operation.
4. Throttle enable- when it is shorted to pin 17 the controller will act upon the throttle signal given.
5. Forward or reverse- depending whether it is shorted to pin 18 will determine the direction the controller operates the motor in.
6. Speed pulse- Sends out a TTL signal that is taken off of the hall sensor information that can be translated into the speed of operation.
7. Brake- +5V signal is given off when the controller goes into regenerative mode. This signal can be used to trigger brake lights.
8. Reverse- +5V signal is given off when the controller goes into reverse mode. This signal can be used to trigger reverse lights.
9. Throttle positive reference
10. Throttle negative reference
11. Regen in
12. Do not connect
13. Do not connect
14. Serial out ground- the ground line for the serial out communication.
15. Serial in ground- the ground line for the serial in communication.
16. Enable ground- the ground line for the *enable* circuit.
17. Throttle enable ground- the ground line for the *throttle enable* circuit.
18. For/rev ground- the ground line for the *forward and reverse* circuit.
19. Speed pulse return- The return line for the *speed pulse* signal.
20. Brake Return- The return line for the *braking* signal.
21. Reverse Return- The return line for the *reverse* signal.
22. Throttle-in –
23. Regenerative braking positive reference
24. Regenerative braking negative reference
25. Do not connect



Appendix B: Explanation of Register names

B.1 Prefixes

The manual frequently uses names that refer to specific register values, i.e. [SV_FAULT1]. To facilitate understanding, they have been arranged by their prefix's to let the reader know how they can be used. The following table lists the prefixes and what they mean:

EEPROM Prefixes:

FS_	Factory setting. Read only
CG_	Configuration value, user writeable
DF_	Default value for like-named RAM register, user writeable
FD_	Factory default value for like-named RAM register, read-only

RAM Prefixes:

SI_	Serial input register, user writeable
CB_	Command Boolean, set or cleared with command inputs
SV_	State variable, read-only
IN_	Input value set by arbitration logic, read-only
AM_	Analog measurement value, read-only
AI_	Analog input value from discrete interface, read-only
BM_	Boolean measurement value, read-only
BI_	Boolean input value from discrete interface, read-only
PM_	Performance metric, read-only

B.2 Suffixes

The suffixes are not as easily recognizable as the prefixes. However they tend to be abbreviations or full words to dictate their relationship. Some common suffixes are:

General Suffixes:	General meaning
THR	Having to do with the throttle or acceleration
RGN	Having to do with regenerative braking, or negative PHASE current
I	Current
SPD or SPEED	Having to do with speed control or RPM
PH	Having to do with the Phases, out or in
SUPPLY	Having to do with the Supply voltage or current

A register name is recognized by being encased in [].

Example:

[SV_FAULT1] => The ***SV*** as stated in the table denotes that it is a state variable. The second portion, *FAULT1* by recognition is a fault, error, etc.

Appendix C: RAM Registers

RAM Variable	Name	Description
Unsigned Integers		
00_H	SI_DESIREDSPEED	Serial speed in (deci-Hz)
01	SI_THRILIMIT	Serial throttle current limit in (deci-A)
02	SI_BRKILIMIT	Serial regen current limit in (dA)
03	SI_KP	Proportional coefficient for speed control
04	SI_KI	Integral coefficient for speed control
05	SI_KT	Phase current to speed error speed control coefficient
07	SI_MAXSPEEDERROR	Speed error clamping value
09	SI_PHASEIRAMP	Ramp rate for serial phaseI input, deciA/(seconds/60)
0A	SI_SPEEDRAMP	Ramp rate for serial speed input, deci-hz electrical/(seconds/15)
0B	IN_DESIREDSPEED	Desired speed
0C	AM_SPEED	Actual speed (deci-Hz)
10	SV_TARGETPHASEI	Target current (dA)
11	SV_THERMALLIMITMOTOR	Thermal motor current limit (dA)
12	SV_HEATSINKDERATING	Heatsink thermal derating ratio
13	SV_MAXTHRI	Maximum throttle current (dA)
14	SV_MAXRGNI	Maximum regen current (dA)
17	AI_THR	Discrete throttle in
18	AM_RGN	Discrete regen in
1C	IN_RGNILIMIT	Discrete regen current limit I (dA)
Signed Integers		
60	SI_DESIREDPHASEI	Serial phase current in (dA)
61	IN_DESIREDPHASEI	Phase current in (dA)
64	AM_SUPPLYV	Measured supply voltage (dV)
65	AM_MOTORT	Measured motor temp (degrees C * 10)
66	AM_HTSINKT	Measured heatsink temp (degrees C * 10)
67	AM_SUPPLYI	Measured logic supply current (mA)
Unsigned bytes and Boolean		
90	SI_MINFANSPEED	Minimum fan speed (0-3)
96	SV_DRIVESTATE	Operating status
97	BM_OBSERVEDDIR	Observed direction of rotation
98	SV_FAULT1LATCH	Latched values of below
99	SV_FAULT1	Bit-coded fault indications that prevent operation
9A	SV_FAULT2	Bit-coded fault indications of sensor problems
9B	SV_FAULT3	Bit-coded fault indications of warnings
9C	SV_FAULT4	Bit-coded fault indications of current limiting
9D	SV_FANSPEED	Actual fan speed setting
9E	IN_DISABLE	Disable input, equal to [CB_DISABLE] | [BI_DISABLE] | wrong direction

RAM Variable	Name	Description
9F	IN_THRENABLE	Throttle enable input, true when [CB_THRENABLE] AND [BI_THRENABLE]
A0	IN_FORWARD	Input direction
A1	SV_FORWARD	Actual operating direction
A2	SV_SPEEDCONTROL	When true, speed control
AA	BI_DISABLE	State of digital disable input
AB	BI_THRENABLE	State of throttle enable input
AC	BI_FORWARD	State of forward input
AD	CB_DISABLE	Serial disable input
AE	CB_THRENABLE	Serial throttle enable input
Unsigned Long Integers		
F0-F3	SI_UL[4]	Used as input registers
F8	PRODUCT	Returns product string
F9	BUILD	Returns software build string
FA	BUILDDATE	Returns build date string

Power up Values and Access Limits

RAM Variable	Name	Default at start-up	access/range
Unsigned Integers			
00_H	SI_DESIREDSPEED	0	read/write
01	SI_THRILIMIT	65535	read/write
02	SI_BRKILIMIT	65535	read/write
03	SI_KP	DF_KP	read/write
04	SI_KI	DF_KI	read/write
05	SI_KT	DF_KT	read/write
07	SI_MAXSPEEDERROR	DF_MASSPDERROR	read/write
09	SI_PHASEIRAMP	CG_PHIRAMP	read/write
0A	SI_SPEEDRAMP	CG_SPDRAMP	read/write
0B	IN_DESIREDSPEED	0	read only
0C	AM_SPEED	0	read only
10	SV_TARGETPHASEI	0	read only
11	SV_THERMALLIMITMOTOR	-	read only
12	SV_THERMALLIMITRGN	-	Read only
13	SV_MAXTHRI	-	Read only
14	SV_MAXRGNI	-	Read only
17	AI_THR	-	Read only
18	AM_RGN	-	Read only
1C	IN_RGNILIMIT	65535	Read only
1D	SV_LAST	-	Read only

RAM Variable	Name	Default at start-up	Access/range
Signed Integers			
60	SI_DESIREDPHASEI	0	Read/write
61	IN_DESIREDPHASEI	0	Read only
64	AM_SUPPLYV	-	Read only
65	AM_MOTORT	-	Read only
66	AM_HTSINKT	-	Read only
67	AM_SUPPLYI	-	Read only
68	AM_HSINKTEST	-	Read only
69	AM_MOTORTEST	-	Read only
Unsigned bytes and Boolean			
90	SI_MINFANSPEED	0	Read/write
96	SV_DRIVESTATE	-	Read only
97	BM_OBSERVEDDIR	1	Read only
98	SV_FAULT1LATCH	-	Read only
99	SV_FAULT1	-	Read only
9A	SV_FAULT2	-	Read only
9B	SV_FAULT3	-	Read only
9C	SV_FAULT4	-	Read only
9D	SV_FANSPEED	-	Read only
9E	IN_DISABLE	-	Read only
9F	IN_THRENABLE	-	Read only
A0	IN_FORWARD	-	Read only
A1	SV_FORWARD	-	Read only
A2	SV_SPEEDCONTROL	-	Read only
AA	BI_DISABLE	-	Read only
AB	BI_THRENABLE	-	Read only
AC	BI_FORWARD	-	Read only
AD	CB_DISABLE	(maxSCIidle > 0) OR (NOT(CG_ENDISCRETE_DISABLE))	Read only
AE	CB_THRENABLE	1	Read only
Unsigned Long Integers			
F0-F3	SI_UL[4]	-	Read/write
F8	PRODUCT	"EVC-200"	Read only
F9	BUILD	"Build xx"	Read only
FA	BUILDDATE	"YYYYMMDD"	Read only

Appendix D: EEPROM Registers

EEPROM Variable	Name	Description
Unsigned Integers		
00_H	**CG_BAUDRATE**	**I/O baud rate**
02	FS_ABSMINV	Absolute minimum voltage for operation (dV)
03	CG_MINV	Voltage at which max throttle current is zero (dV)
04	CG_MINVGUARD	Voltage at which max throttle current limiting starts (dV)
05	CG_MAXVRGNGUARD	High voltage cut-off start point for regen
06	CG_MAXVRGN	Maximum voltage for regen
07	CG_MAXVGUARD	Voltage at which max phase current limiting begins due to over voltage (dV)
08	CG_MAXV	Voltage at which phase current is zero (dV)
09	FS_ABSMAXVGUARD	Voltage at absmaxthrI1 set point (dV)
0A	FS_ABSMAXV	Absolute maximum voltage for operation (dV)
0B	FS_MINGUARDDELTA	Minimum difference between minV and minVguard, also maxV
0C	CG_MINFREQ	Minimum commutation frequency for speed control
0D	DF_KI	Default value for SI_Ki
0E	DF_KP	Default value for SI_Kp
0F	DF_KT	Default value for SI_Kt
11	DF_MAXSPDERROR	Clamping value for speed error in speed control
12	SC_SUPPLYV	Scale value for supply voltage
13	SC_SUPPLYI	Scale value for supply current
15	FS_SCHTSINKT	Scale value for heatsink temperature
16	CG_SCTHR_SPEED	Scale value for throttle input into speed (speed control)
17	CG_SCTHR_TORQUE	Scale value for throttle input into amps (torque control)
18	CG_SCRGN_TORQUE	Scale value for regen input into amps
1A	CG_SCMOTORT	Scale value for motor temperature
1D	CG_MAXMOTORI	Maximum motor current, throttle or regen (deci-Amps)
1E	DF_PHASEIRAMP	Default value for SI_PHASEIRAMP
1F	DF_SPEEDRAMP	Default value for SI_SPEEDRAMP
21	CG_SPEEDTHRESHOLD	Safe speed for changing motor direction
22	CG_MINSUPPLYI	Minimum supply current when fans are off
23	CG_MAXSUPPLYI	Maximum supply current when fans are off
24	CG_MINFANSUPPLYI	Minimum supply current when fans are on
25	CG_MAXFANSUPPLYI	Maximum supply current when fans are on

EEPROM Variable	Name	Description
2C-2F	CG_FANI[4]	Current thresholds for fan control
36	CG_MAXTHRI0	Maximum throttle current (dA)
37	CG_MAXTHRI1	
38	CG_MAXRGNI0	Maximum regen current (dA)
39	CG_MAXRGNI1	
3A	FS_ABSMAXTHRI0	Factory set maximum value for [CG_maxthrI] (dA)
3B	FS_ABSMAXTHRI1	
3C	FS_ABSMAXRGNI0	Factory set maximum value for [CG_maxrgnI] (dA)
3D	FS_ABSMAXRGNI1	
3E	CG_MOTORITCOEFF	I^2t coefficient for estimating heatsink temp
3F	FS_HSINKITCOEFF	I^2t coefficient for estimating motor temp
Signed Integers		
60	FS_OFSUPPLYV	Offset value for supply voltage
61	FS_OFSUPPLYI	Offset value for supply current
64	CG_OFMOTORT	Offset value for motor temp
65-67	CG_FANTEMP[3]	Temperature transition points for fan control
71	CG_DEFAULT_MOTORT	Assumed motor temp when sensor fails
72-75	CG_TLIMTMTR[3]	Motor Temperature transition points (deci-Celsius)
7C-7D	CG_TLIMIMTR[2]	0-256 % of current at the corresponding Temp. Implied denominator of 256
Unsigned bytes and Boolean		
90	CG_ECHO	When true, echo characters as they are received
91	CG_TEXTERRORS	When true, send text messages for errors, else send two digit codes
92	CG_LINEFEEDS	When true, use CR-LF combinations at end of lines
93	CG_MAXSCIIDLE	Maximum idle time for serial interface watchdog fault in tenths of a second, 0 disables
95	CG_60DEGREEHALLS	When true, assume hall-effect sensor are 60 electrical degrees apart
97	CG_INVERTDIR	When true, reverse definition of forward
98	DF_SPEEDCONTROL	When true, power-up in speed control mode
99	CG_ENDISCRETE_THR	When true, use discrete throttle and regen inputs
9B	CG_ENDISCRETE_DIR	When true, use discrete direction input
9C	CG_ENDISCRETE_THRENABLE	When true, use discrete throttle enable input
9D	CG_ENDISCRETE_DISABLE	When true, use discrete disable input
9E	CG_THRDEADBAND	Offset (in counts) of throttle input
9F	CG_RGNDEADBAND	Offset (in counts) of regen input
A0	CG_GAPDEADBAND	Offset (in counts) of gap input (not used)
A1	CG_RTSUPPLYV	Filtering level for supply voltage (0:none to 4:max)
A2	CG_RTSUPPLYI	Filtering level for supply current measurement
A4	CG_RTHSINKT	Filtering level for heatsink temp measurement
A5	CG_RTHTR	Filtering level for throttle input
A6	CG_RTRGN	Filtering level for regen input

EEPROM Variable	Name	Description
A8	CG_RTMOTORT	Filtering level for motor temp measurement
A9	CG_SOFTSTARTN	Speed of softstart operation (0:fastest ramp to 7:slowest ramp)
AA	CG_NAUTORESETS	Number of automatic reset attempts in four seconds
B9	CG_MOTORTIMEC	Thermal time constant coefficient for motor
BA	FS_HSINKTIMEC	Thermal time constant coefficient for heatsink
BB	CG_AISPDTOPWMFREQMULT	Sets threshold for detecting max torque production
UNSIGNED LONGS		
F0	CG_SPDNUMERATOR	Numerator used for speed calculation

Factory Settings and Access Limits

EEPROM Registers	Name	Access/range	Factory Setting EV-C200-XX2	
			-042	-092
Unsigned Integers				
00_H	CG_BAUDRATE	Read/write	9600	9600
02	FS_ABSMINV	Read only	250	400
03	CG_MINV	Between [ABSOLUTEMINV] and [MINVOLTAGEGUARD]-[MINGUARDDELTA]	280	450
04	CG_MINVGUARD	Between [MINVOLTAGE]+[MINGUARDDELTA] and [MAXVOLTAGEGUARD]	350	550
05	CG_MAXVRGNGUARD	Read/write	630	1220
06	CG_MAXVRGN	Read/write	680	1350
07	CG_MAXVGUARD	Between [MINVOLTAGEGUARd] and [MAXVOLTAGE]-[MINGUARDDELTA]	630	1350
08	CG_MAXV	Between [MAXVOLTAGEGUARD]+[MINGUARDDELTA] and [ABSOLUTEMAXVOLTAGE]	680	1450
09	FS_ABSMAXVGUARD	Read only	630	1350
0A	FS_ABSMAXV	Read only	680	1500
0B	FS_MINGUARDDELTA	Read only	50	50
0C	CG_MINFREQ	Read/write	0	0
0D	DF_KI	Read/write	2600	1500
0E	DF_KP	Read/write	867	150
0F	DF_KT	Read/write	0	0
10	DF_KS	Read/write	0	0
11	DF_MAXSPDERROR	Read/write	60	60

EEPROM Registers	Name	Access/range	Factory Setting EV-C200-XX2	
			-042	-092
12	SC_SUPPLYV	Read only	61	150
13	SC_SUPPLYI	Read only	206	206
15	FS_SCHTSINKT	Read only	139	139
16	CG_SCTHR_SPEED	Read/write	80	80
17	CG_SCTHR_TORQUE	Read/write	207	120
18	CG_SCRGN_TORQUE	Read/write	207	120
1A	CG_SCMOTORT	Read only	139	139
1B	FS_SCPWMFREQ	Read only	122	122
1D	CG_MAXMOTORI	Limit to 0 to 4095	3600	1500
1E	DF_PHASEIRAMP	Read/write	65535	65535
1F	DF_SPEEDRAMP	Read/write	65535	65535
21	CG_SPEEDTHRESHOLD	Read/write	20	20
22	CG_MINSUPPLYI	Read/write	40	40
23	CG_MAXSUPPYI	Read/write	250	250
24	CG_MINFANSUPPLYI	Read/write	200	200
25	CG_MAXFANSUPPLYI	Read/write	950	950
2C-2F	CG_FANI[4]	Read/write	1200,1600, 2000,2400	800,1000, 1200,1400
36	CG_MAXTHRI0	<= absmaxthrI	3010	1770
37	CG_MAXTHRI1		2600	1500
38	CG_MAXRGNI0	<= absmaxrgnI	2060	1200
39	CG_MAXRGNI1		1700	1020
3A	FS_ABSMAXTHRI0	Read only	3100	1800
3B	FS_ABSMAXTHRI1		2600	1500
3C	FS_ABSMAXRGNI0	Read only	2060	1200
3D	FS_ABSMAXRGNI1		1720	1000
3E	CG_MOTORITCOEFF	Read/write	19	73
3F	FS_HSINKITCOEFF	Read only	41	121
Signed Integers				
60	FS_OFSUPPLYV	Read only	7	7
61	FS_OFSUPPLYI	Read only	0	0
63	FS_OFHSINKT	Read only	-611	-611
64	CG_OFMOTORT	Read only	-611	-611
65-67	CG_FANTEMP[3]	Read/write	350,400,450	350,400,450
68-6F	reserved	Read only	0	0
70	FS_DEFAULT_HSINKT	Read only	750	750
71	CG_DEFAULT_MOTORT	Read/write	750	750
72-75	CG_TLIMTMTR[3]	Read/write	500,770,920, 1000	750,850,1000 ,1100
77-7B	FS_TLIMHSINK[5]	Read only	450,680,840, 900,32767	300,500,650, 750,32767
7C-7D	CG_TLIMIMTR[2]	0-256	182,105	230,128
7E-7F	FS_TLIMIHSINK[2]	Read only	179,92	179,92

EEPROM Registers	Name	Description	Factory Setting EV-C200-XX2	
			-042	-092
Unsigned bytes and Boolean				
90	CG_ECHO	Coerced to 0-1	1	1
91	CG_TEXTERRORS	Coerced to 0-1	1	1
92	CG_LINEFEEDS	Coerced to 0-1	1	1
93	CG_MAXSCIIDLE	Read/write	0	0
95	CG_60DEGREEHALLS	Coerced to 0-1	1	1
97	CG_INVERTDIR	Coerced to 0-1	0	0
98	DF_SPEEDCONTROL	Coerced to 0-1	0	0
99	CG_ENDISCRETE_THR	Coerced to 0-1	1	1
9B	CG_ENDISCRETE_DIR	Coerced to 0-1	1	1
9C	CG_ENDISCRETE_THRENABLE	Coerced to 0-1	1	1
9D	CG_ENDISCRETE_DISABLE	Coerced to 0-1	1	1
9E	CG_THRDEADBAND	Read/write	8	8
9F	CG_RGNDEADBAND	Read/write	8	8
A0	CG_GAPDEADBAND	Read/write	8	8
A1	CG_RTSUPPLYV	0-4	2	2
A2	CG_RTSUPPLYI	0-4	4	4
A4	CG_RTHSINKT	0-4	4	4
A5	CG_RTTHR	0-4	1	1
A6	CG_RTRGN	0-4	1	1
A8	CG_RTMOTORT	0-4	4	4
A9	CG_SOFTSTARTN	0-7	6	6
AA	CG_NAUTORESETS	0-64	0	0
B9	CG_MOTORTIMEC	Read/write	2	2
BA	FS_HSINKTIMEC	Read only	15	15
BB	CG_AISPDTOPWMFREQMULT	0-4	0	0

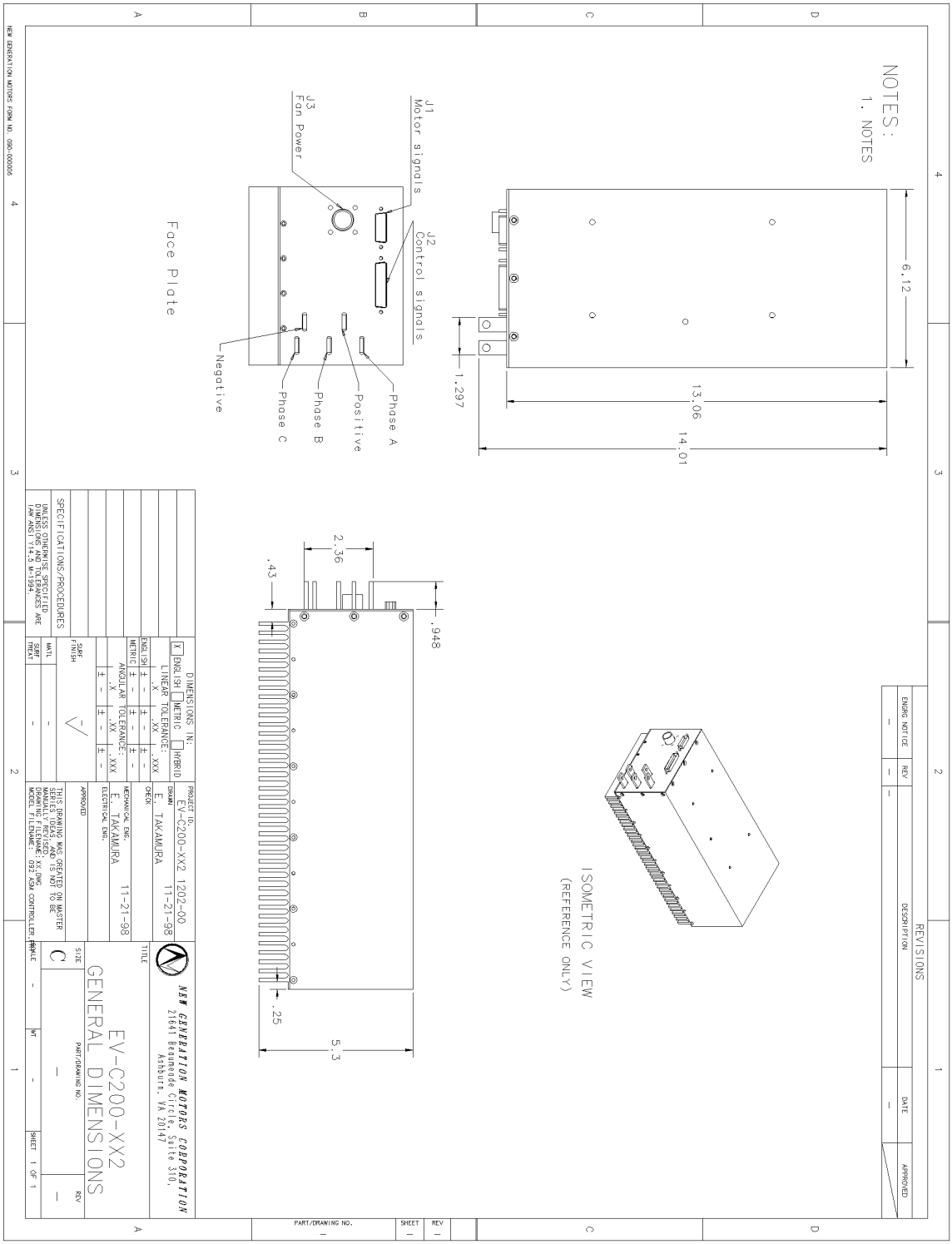
Appendix E: Drive States

Value	Name	Description
32_D	DS_POWERUP	Initial state
33-62		Powering up
63	DS_POWERUPEND	Power-up period over
64	DS_SHUTDOWN	Stopped and disabled
65	DS_DISABLECOAST	Disabled but not stopped
66	DS_INTERLOCK	Type 1 fault detected, waiting for disable command
67	DS_INTERLOCKCOAST	Type 1 fault detected, waiting for disable command, not stopped
74	DS_STOPPED	Enabled but not moving or throttling
75	DS_COASTING	Enabled and moving but not throttling
76	DS_NO_LONGER_THR	Leaving DS_thr mode
77	DS_NO_LONGER_BRK	Leaving DS_brk mode
78	DS_THR	Throttling
79	DS_BRK	Braking
1	DS_PROGRAM	Shutdown with programming enabled

Appendix F: Error Messages and Codes

Number	Message	Description
#00	Ok	Normal completion of a command or set
#02	Bad Command	Command character not !,<,>,=, or ?
#03	SCI Overflow	Input buffer overflow, over 15 characters in input
#04	Bad Input	First two characters not hex digits or input less than 3 characters
#05	Command Failed	Not in the correct mode for command, such as: sending a forward command when discrete direction enabled, sending a speed control signal when in speed control, sending a program control signal when drivestate isn't shutdown
#06	Not PGM Mode	Command requires that the drivestate be program, or attempting to write to an EEPROM variable while drivestate is not program
#09	Read Only	Attempting to write to a read-only variable
#0A	Out of Range	Attempting to write a value to a limited-write setting that is outside the factory limits. This includes attempts to violate the absminV < minV < minVguard < maxVguard < maxV < maxVguard relationship. Changes to these values must be made in the correct order
#0D	Bad Address	Attempting to access an address that does not exist
#0E	MaxthrI0 too hi	When programming, can't set [CG_MINVGUARD] because [CG_MAXTHRI0] is too high
#0F	MaxthrI1 too hi	When programming, can't set [CG_MAXVGUARD] because [CG_MAXTHRI1] is too high
#10	MaxrgnI1too hi	When programming, can't set [CG_MAXVRGNGUARD] because [CG_MAXRGNI1] is too high

Appendix G: Basic Dimensions



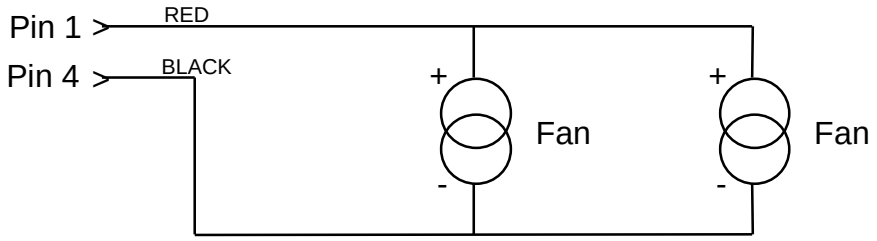
Appendix H: Discrete control Quick Start Guide

The Quick Start Guide is not intended as a replacement to the NGM-EV-C200 series Controller OPERATING MANUAL. The entire NGM-EV-C200 series Controller OPERATING MANUAL must be read before operating the controller.

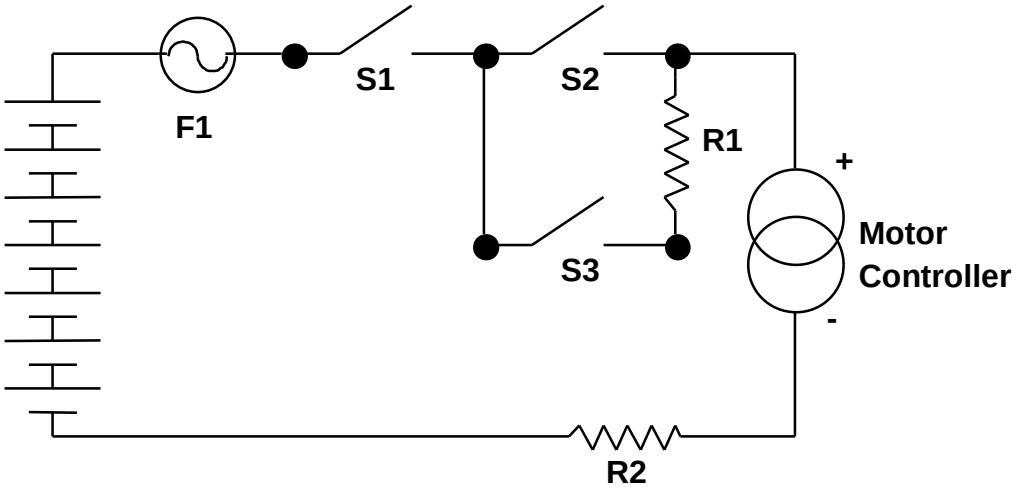
The controller is shipped with its power up defaults set to ***discrete*** control in ***torque control*** mode.

Connections:

1. Motor Phase leads. *No less than AWG 6 gage (4.11mm) wire,* AWG 4 (5.18mm) or larger is preferable.
2. Motor Sense, Connector J1.
3. Control, Connector J2.
4. Fan Power, Connector J3. Below is the schematic for connecting the fans.

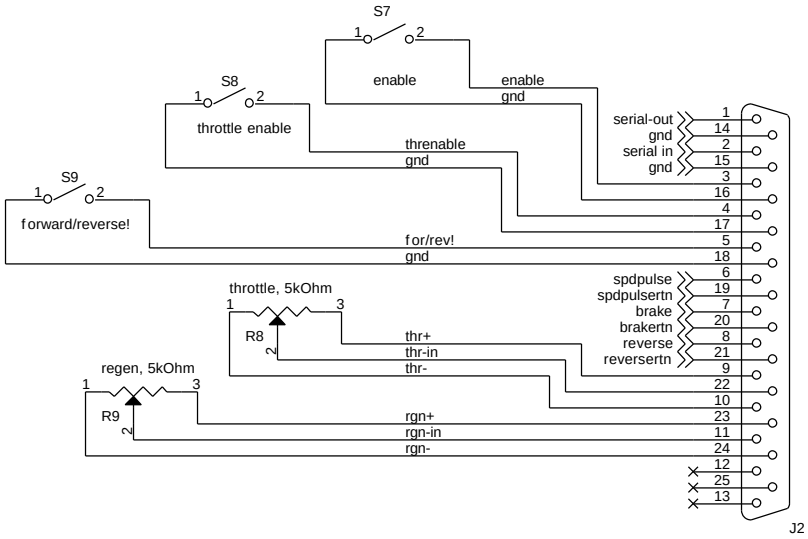


5. Power. *No less than AWG 6 gage (4.11mm),* AWG 4 (5.18) or larger is preferred. A ***Pre-charge circuit must be in place*** to protect the controller. Schematic below.



R1 should have a resistance such that the current through it at turn-on is at most 30A. Resistor R2 is an optional 100 A shunt for measuring the motor controller current.

6. *Example:* Connection for discrete control.



Operation:

1. There are five inputs that must be communicated to the controller as a minimum; Enable, Throttle- Enable, Direction, Throttle, and Regen.

> ***Forward/Reverse*:** Forward corresponds to open circuit and reverse to *closed*. It is recommended that the direction signal be wired directly to a switch for maximum safety and reliability
>
> ***Enable:*** An open circuit immediately disables all torque production.
>
> ***Throttle enable:*** When open-circuited, the maximum throttle current is set to zero (i.e. it can not produce accelerating torque, but the controller can still operate in regen. It is suggested that this input be wired to a switch on the brake pedal).

2. When in discrete torque control, the desired phase current, which is proportional to torque, is determined by the difference between the throttle and regen analog inputs. When this is greater than zero, driving torque is produced. When less than zero, regen is applied. When equal to zero, the motor coasts. The inputs could be references across potentiometers having resistance values ~4k-20k.

Installing the BMS Master Control Unit BMS-MCU-4C

Introduction

The MCU-4C control unit is designed to operate with EV Power cell modules to facilitate automatic battery management of large format LiFePO4 batteries. It is microcontroller based and has a number of features which make it well suited to larger battery installations.

It is primarily designed for electric vehicle application where an onboard charger is employed.

Features

Features:
- 4 signal input channels for cell module (voltage) or thermistor (temperature) monitoring.
- Low power consumption when idle.
- Throttle cutback control relay output.
- Warning light/buzzer output relay.
- Main contactor control relay.
- Onboard AC/DC relays to control charger. Can switch up to 3 phases at 20A/phase.
- Automatically detects when AC charger input is connected to engage onboard charger relays.
- All relay outputs fully programmable via BASIC programming language.
- 12V-72V or 96V-350V models available.

Tools & Materials

The following tools and materials will be required:

- Side cutters
- Wire stripping tool.
- philips screwdrivers
- small soldering iron and resin core solder
- Multimeter
- light duty speaker cable (Jaycar part number WB1702)

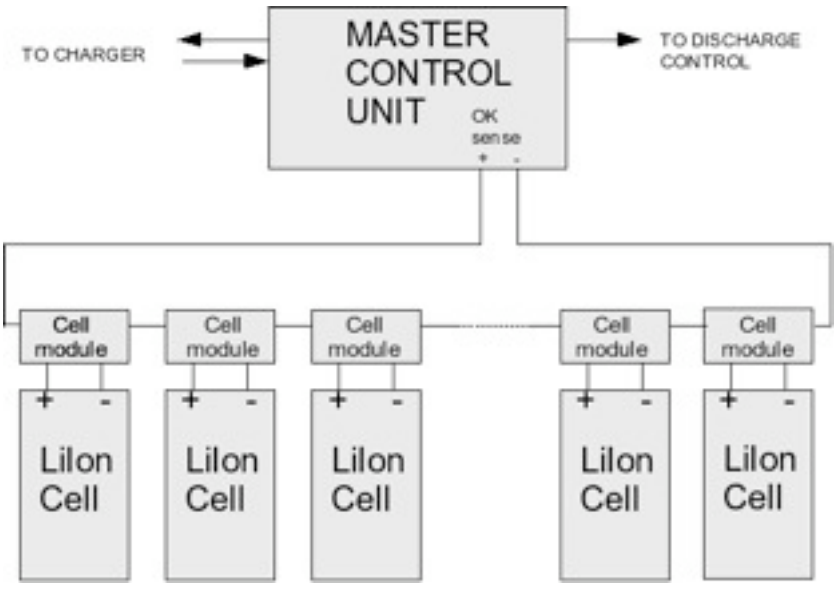
Disclaimer

Before commencing any operations with an electric vehicle or other high voltage DC system you should consider if you have the relevant experience. 48VDC and greater voltages can be lethal. Voltages discussed herein such as 240VAC and 350VDC are most certainly lethal if accidentally touched. Accidental short circuits of power systems will result in damage to tools and equipment and can result in fires.

If you are not completely confident working with hazardous voltages please find someone who is. EV Works will assume no responsibly whatsoever for damage, injury or death arising from the use of the supplied equipment or instructions.

Cell Modules

It is assumed that the LFP cells, interconnects and cells modules are already installed and signal wires daisy-chain connected together. If not, please refer to the relevant instructions for this.



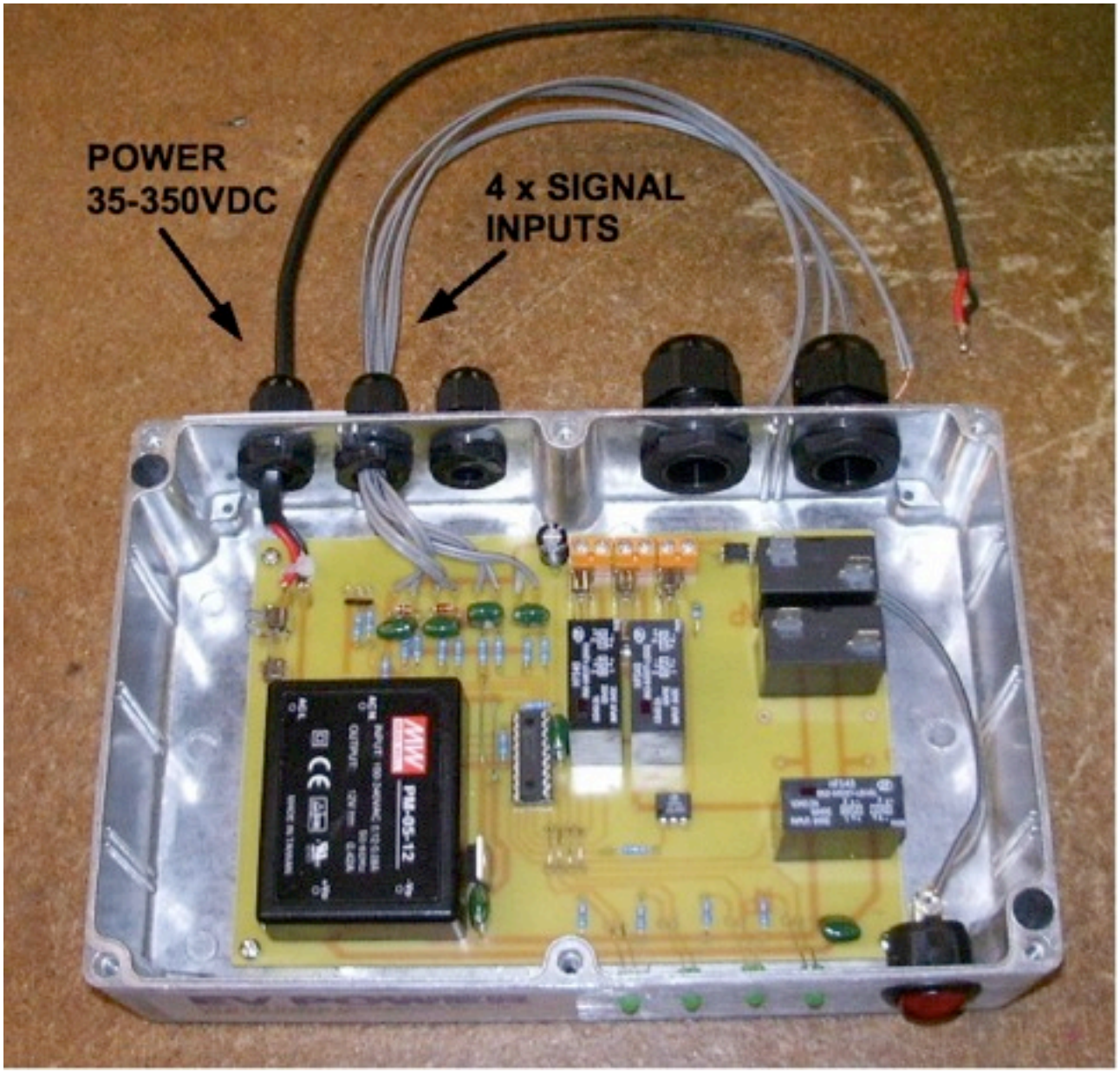
If the batteries are distributed in two or more locations then one channel can be used for each. Connect the two wires from Channel 1 input to the two ends of the signal daisy chain. Cannel 2... etc. Polarity does not matter.

Unused channels can be left with the copper ends twisted together. This will give the master unit the signal that these channels are always OK.
Alternatively, unused channels can be used to monitor over temperature via simple PTC thermistors.

Connecting Power

The MCU-V6-4C is powered directly from the battery pack itself. The normal supply voltage is in the range 96VDC - 350VDC. Lower supply voltages can be accommodated on request. A fuse is located within the unit so no external fuse should be necessary. Connect directly to the positive and negative terminals of the battery pack.
Under no circumstances should any part of the battery pack be tapped for lower voltages, this will unbalance the battery and cause problems.

The main power switch is on the front panel. This can be used to manually reset the unit if required.



Once power is connected and the cell module channels are active the unit can be powered up. If everything is OK the four LEDs on the front panel will illuminate. Try manually disconnecting each channel in turn. Its corresponding LED will go off and stay off until the unit is power cycled or the charger input is activated. See later.

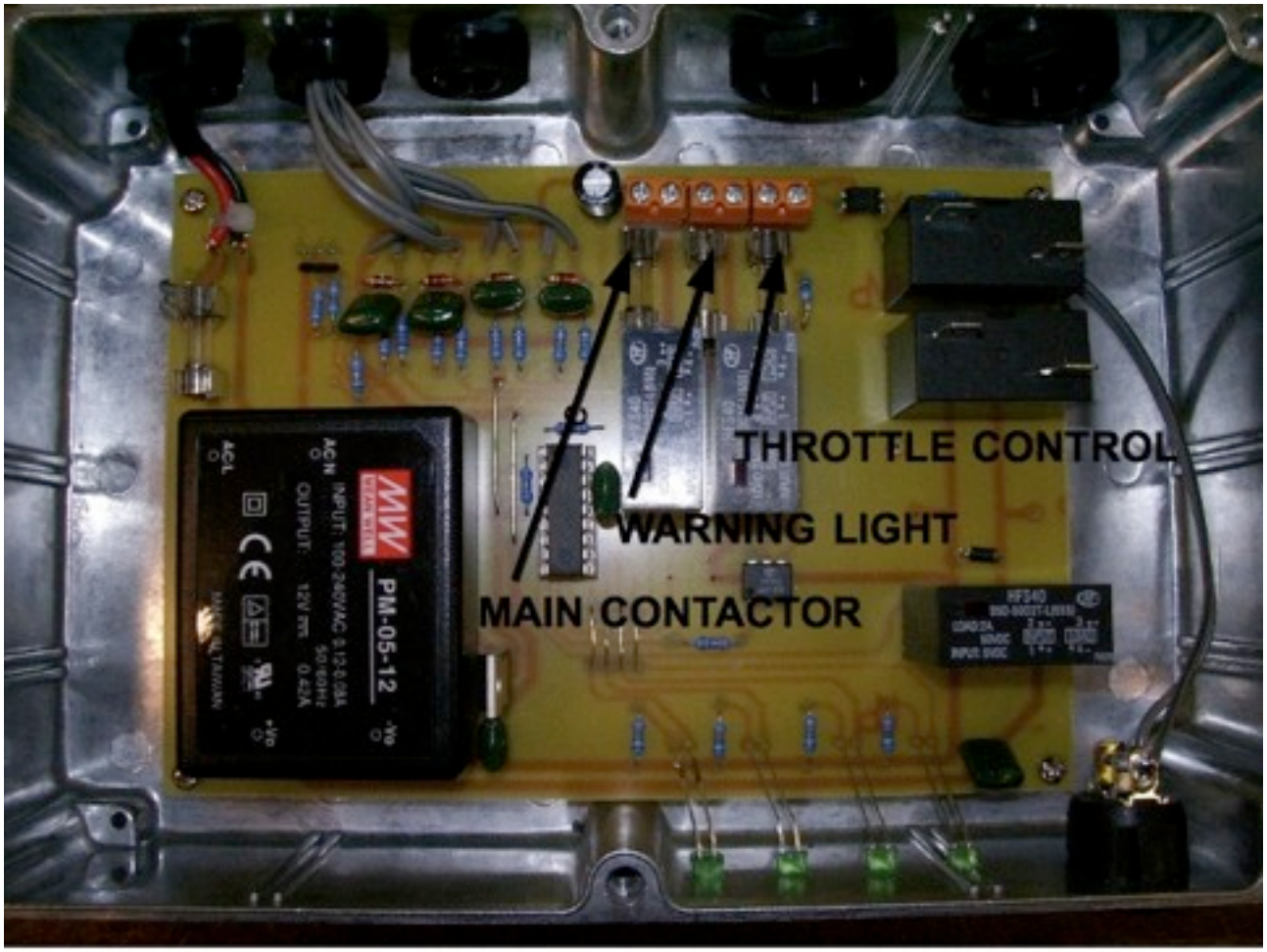
Auxiliary Outputs

The MCU has three outputs to control vehicle functions in the event of one of the channels going open circuit. On an error the chain of events is this:
1) The warning light/buzzer will be immediately activated.
2) The throttle control relay will progressively reduce the throttle pedal input to the controller.
2) If after about 10 seconds seconds there is still an error situation the MCU will drop the main contactor thus completely disabling the vehicle. It will remain disabled even if the error situation ceases. Plugging in the charger or power cycling the MCU will reset the system.

THROTTLE CONTROL should be connected in parallel with the throttle input to the motor controller. If the throttle input has two wires connect directly to these. If the throttle input has three wires the throttle control should connect between the signal and ground lines. If in doubt leave it unconnected. It is non-polarized.

The WARNING LIGHT relay should be used to switch a warning light indicator or buzzer that is clearly visible/audible to the driver. This is a solid state relay output and is polarized. The wire with the black stripe is negative. Maximum 30VDC 2 Amps.

The MAIN CONTACTOR relay can switch the coil on the main contactor. This is a solid state relay output and is polarized. The wire with the black stripe is negative. Maximum 30VDC 2 Amps. Ensure a flyback diode is connected across the coil to prevent voltage spikes from damaging the relay.



Connecting the Charger

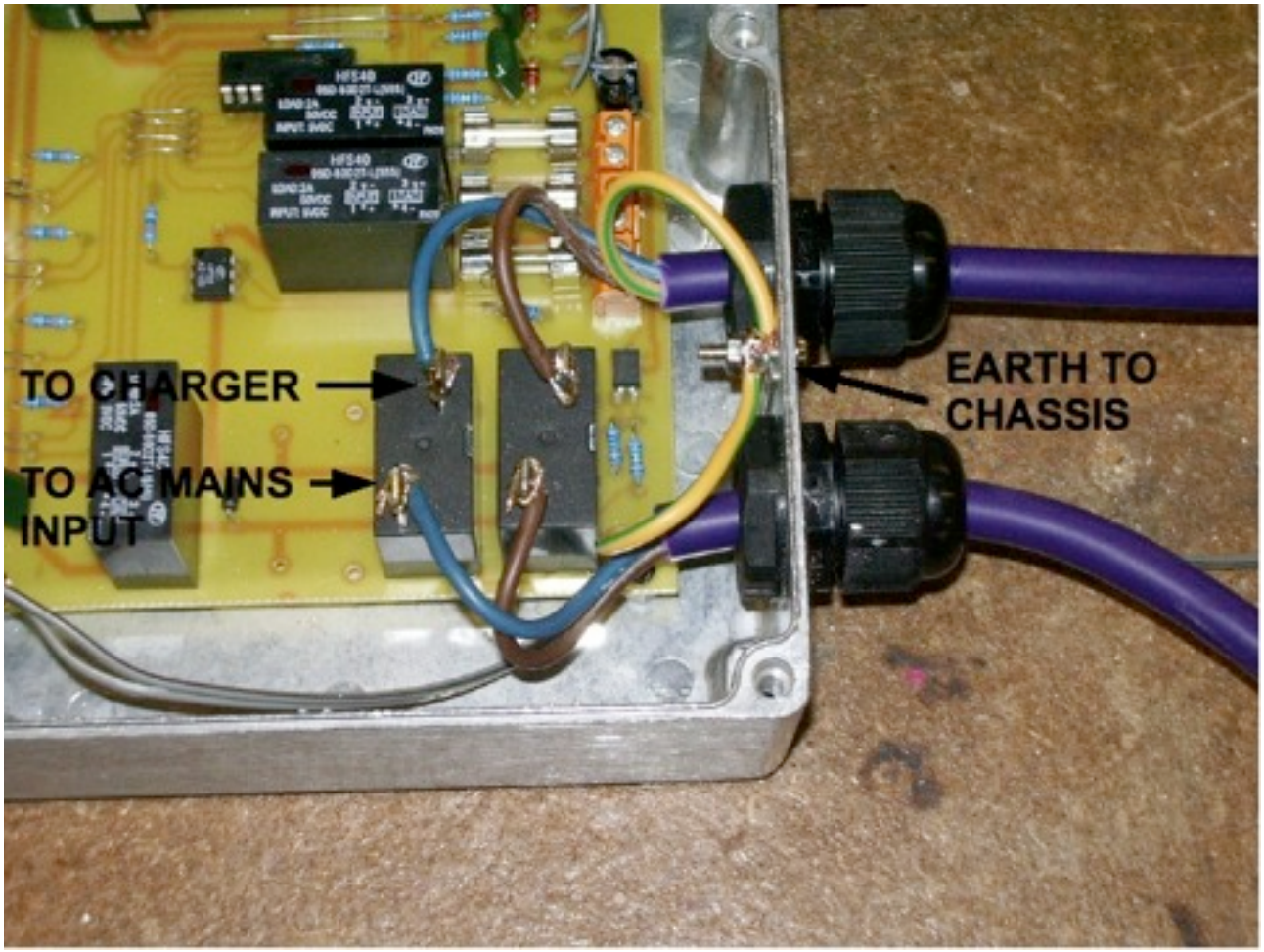
Charging is facilitated generally using an on-board AC mains charger. The AC side of this charger is switched by the MCU. In this way the MCU can switch off the charger in the event of a battery error such as over charge or a failed cell.

The BMS is designed only as a safety and balancing device and should not be used to disconnect the charger at end of charge. The charger should be smart enough to do this for itself.

Any other charging means such as solar charging should be switched by the main contactor and not by the charging relays.

Note the wiring layout below. Pay close attention to the input and output sides. This is important so the MCU can detect when the charger is plugged in and engage the charging relays. The connection below is for a single phase charger. A three phase charger can be switched with the addition of a third relay.

All connections should be carefully soldered.



In the event of one of the channels going open circuit the charger will be immediately disconnected. It will remain disconnected until the AC input is disconnected then reconnected or the unit is power cycled. The MCU will not engage charging if there is an ongoing battery error.

EV POWER LFP BATTERY BALANCING SYSTEM DATASHEET

The cell modules are designed to bolt on top of Thunder Sky LFP series cells or similar. They can act as standalone cell balancers or be daisy chained together using a one wire interface which is NC when all the cells are within safe operating voltage limits and open cicuit otherwise. This can be used to control chargers and loads or to interface with an EV Power master unit.

A cell module shunt regulates the cell to which it is attached when the voltage reaches 3.65V. This allows unbalanced cells to even out during charging.

For temperature protection PTC thermistors can be placed in the signal line and placed against selected cells. This will additionally open the signal circuit in over temperature conditions.

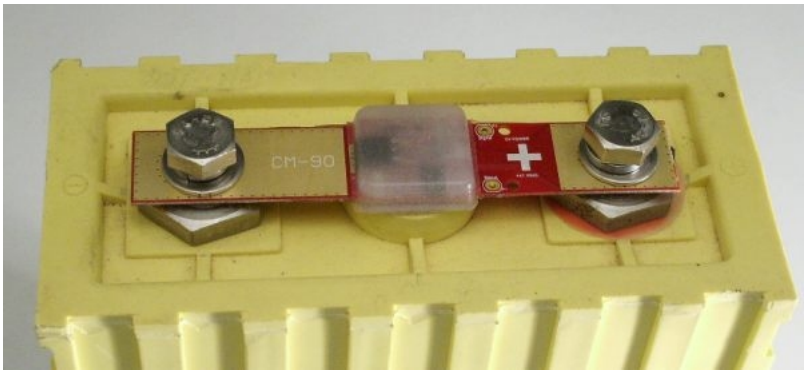
Three modes of operation:
1) Standalone – cell balancers will shunt charge current at 3.65V. Green LED is normally ON if cell is OK. Red LED operates when module is shunting charge current.

2) Daisychained – cell modules are daisychained together via a one wire interface. Operation is similar to standalone mode with the addition that the two signal ends are normally closed circuit if all cells are in the range 2.5 – 4.1V. Open circuit otherwise.

3) Master Control Unit – Various microcontroller based master units are in development to control charging and discharging devices based on daisychained signal outputs.

The system is designed to be failsafe. In order to operate the cell modules require a cell voltage within the recommended limits. An internal fuse protects against overvoltage and cell module failure.

V6 Cell Module Specifications

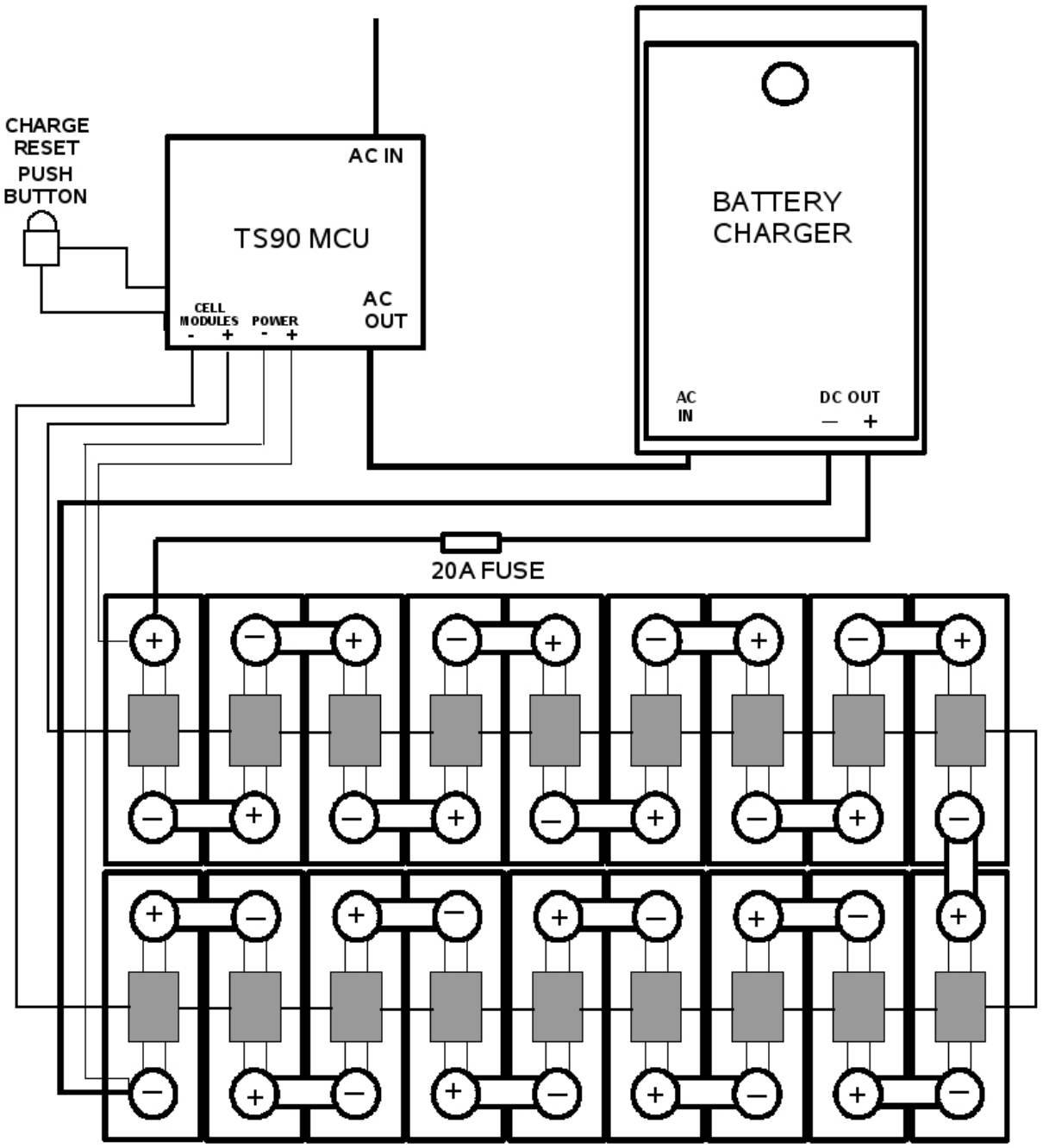


Nominal Cell Voltage:	3.2V
Bypass Voltage:	3.65V (Bypass shunt will switch on)
Max. Bypass Current:	600mA
Weight:	15-20g
Power Consumption:	~3mA @ 3.2V

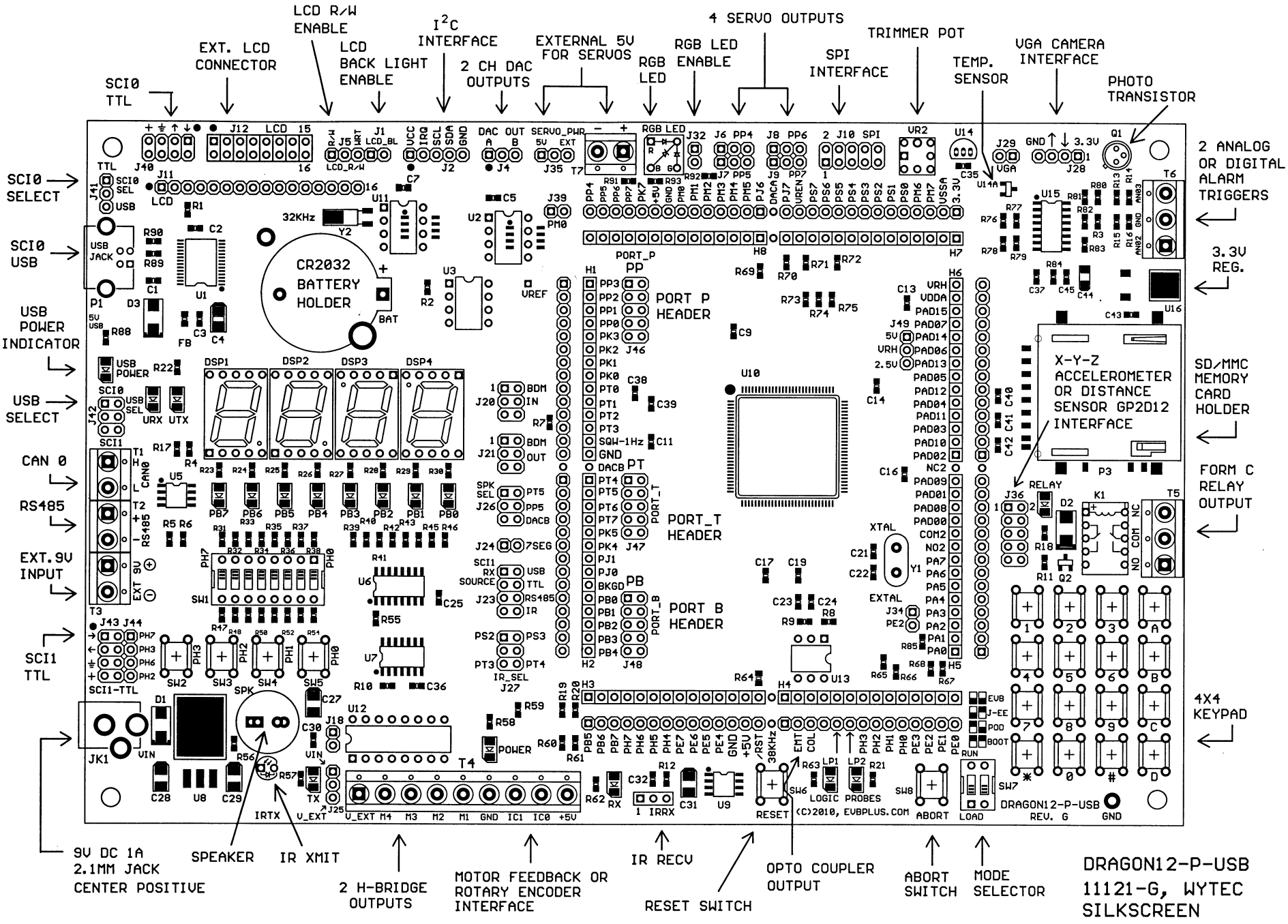
LED Indicators: Green (ON=OK), Red (ON=Bypass active)
Safety Limits: 2.5V < OK < 4.1V
Relay Output: NC when cell voltage is OK. Open circuit with error condition.
Max Signal current: 100mA (non-polarized)
Max height above terminal bolts: 2mm
Epoxy encapsulated against dust and moisture ingress.
Standard sizes available for TS LFP40/60AHA, LFP90AHA, LFP160AHA

For more information on your specific requirement please contact EV Power Australia Pty Ltd.
Http://www.ev-power.com.au Ph: +61 8 9757 2998 WST

TS90 BATTERY MANAGEMENT SYSTEM CONNECTIONS



TRACTION BATTERY WITH CELL MODULES


SCI0 TTL
EXT. LCD CONNECTOR
LCD R/W ENABLE
LCD BACK LIGHT ENABLE
I2C INTERFACE
2 CH DAC OUTPUTS
EXTERNAL 5V FOR SERVOS
RGB LED
RGB LED ENABLE
4 SERVO OUTPUTS
SPI INTERFACE
TRIMMER POT
TEMP. SENSOR
VGA CAMERA INTERFACE
PHOTO TRANSISTOR
2 ANALOG OR DIGITAL ALARM TRIGGERS
3.3V REG.
SD/MMC MEMORY CARD HOLDER
FORM C RELAY OUTPUT
4X4 KEYPAD
SCI0 SELECT
SCI0 USB
USB POWER INDICATOR
USB SELECT
CAN 0
RS485
EXT.9V INPUT
SCI1 TTL
CR2032 BATTERY HOLDER
X-Y-Z ACCELEROMETER OR DISTANCE SENSOR GP2D12 INTERFACE
PORT P HEADER
PORT_T HEADER
PORT B HEADER
9V DC 1A 2.1MM JACK CENTER POSITIVE
SPEAKER
IR XMIT
2 H-BRIDGE OUTPUTS
MOTOR FEEDBACK OR ROTARY ENCODER INTERFACE
IR RECV
RESET SWITCH
OPTO COUPLER OUTPUT
ABORT SWITCH
MODE SELECTOR
DRAGON12-P-USB
11121-G, WYTEC
SILKSCREEN
DRAGON12-P-USB
REV. G
(C)2010, EVBPLUS.COM

Dragon12-Plus-USB Trainer

For Freescale HCS12 microcontroller family

User's Manual for Rev. G board

Revision 1.10

Table OF Contents

Chapter 1. Introduction

1.1 Welcome

Thank you very much for purchasing our Dragon12-Plus-USB trainer. The Dragon12-Plus-USB trainer is a low-cost, feature-packed training board for the new Freescale HCS12 microcontroller family. It is compatible with the Freescale 9S12DP256EVB board and other similar development boards on the market today, but it also incorporates many on-board peripherals that make this board a popular trainer in universities around the world.

For engineers, it is a convenient prototype system suitable for designers who want to rapidly develop and prototype new HCS12 applications. For students, it can not only to be used as a general trainer for freshman and sophomore students, but also as a versatile platform for senior projects as well. The new features of the Dragon12-Plus-USB board create a new potential for students at every level.

The Dragon12-Plus-USB trainer kit comes with the following items:

1. Dragon12-Plus-USB board
2. Software downloadable from our web site:
   a. AsmIDE with HCS12 assembler
   b. Sample programs with source code
   c. Freescale application notes for the HCS12
   d. Data sheets for on-board hardware
   e. User's manual
   f. Reference documents
3. 6 foot USB type B cable
4. 9V, 1A switching power supply AC adapter for North America customers only.

If you miss any part of the kit, please contact sales@EVBplus.com or call 630 894-1440 for help.

The new Dragon12-Plus-USB board is fully backward compatible to the Dragon12-Plus board. All software written for the Dragon12-Plus board will run on the new Dragon12-Plus-USB board without any modifications.

Please carefully examine the default jumper settings before turning on the board:

1. The J1 should have a jumper for LCD backlight.
2. The J24 should have a jumper installed, but J18 should not have a jumper if there is no motor connected to the terminal block T4. The jumper on J18 will turn on the H-Bridge U12. If you see a jumper on J18, move it to J24 to reduce power consumption.
3. The J26 should have a jumper installed in the "TOP" position, so the speaker will be driven by PT5. The speaker can be driven by timer (PT5) or PWM (PP5) or DAC. It defaults for PT5. Without a jumper installed on J26 the speaker won't sound.
4. The J41 should have a jumper installed in the "LOW" position, so the SCI0 receives signal from USB port.
5. The J42 should have two jumpers installed vertically in the "UP" positions, so the USB interface is connected to SCI0. If these two jumpers are installed in the "LOW" positions **and** the jumper on J23 in the "TOP" position labeled with "USB" then the USB interface is connected to SCI1.
6. The J32 should have a jumper installed, so the RGB color LED is enabled. The RGB LED is driven by PP4, PP5 and PP6.

The specification of the switching power supply AC adapter is:

DC input: 110V-240V
DC output: 9V
Current rating: 1A
Type of plug: 2.1mm female barrier plug, center positive

1.2 MC9S12DG256 features and memory map:

The Dragon12-Plus-USB board comes with the MC9S12DP256CCPV or the MC9S12DG256CVPE installed. The MC9S12DG256 is a replacement for the MC9S12DP256 since the latter has been discontinued by Freescale. The only difference between DG256 and DP256 is the number of CAN ports. The DG256 has 2 CAN ports, but the DP256 has 5 CAN ports. Other than the different number of CAN port these two microcontrollers have the same features. If you don't use more than 2 CAN ports these two chips are identical and **all datasheets and manuals** for the DP256 can be used for the DG256.

If your application that needs more than two CAN ports please contact us at sales@evbplus.com and we may be able to ship the board installed with the DP256.

The MC9S12DG256 microcontroller consists of a powerful 16-bit CPU (central processing unit), 256K bytes of flash memory, 12K bytes of RAM, 4K bytes of EEPROM and many on-chip peripherals.

The main features of the MC9S12DG256 are listed below:

- Powerful 16-bit CPU
- 256K bytes of flash memory
- 12K bytes of RAM
- 4K bytes of EEPROM
- SCI ports
- SPI ports
- CAN 2.0 ports
- I^2C interface
- 8-ch 16-bit timers
- 8-ch 8-bit or 4-ch 16 bit PWM
- 16-channel 10-bit A/D converter
- Fast 25 MHz bus speed via on-chip Phase Lock Loop
- BDM for in-circuit programming and debugging
- 112-pin LQFP package offers up to 91 I/O in a small footprint

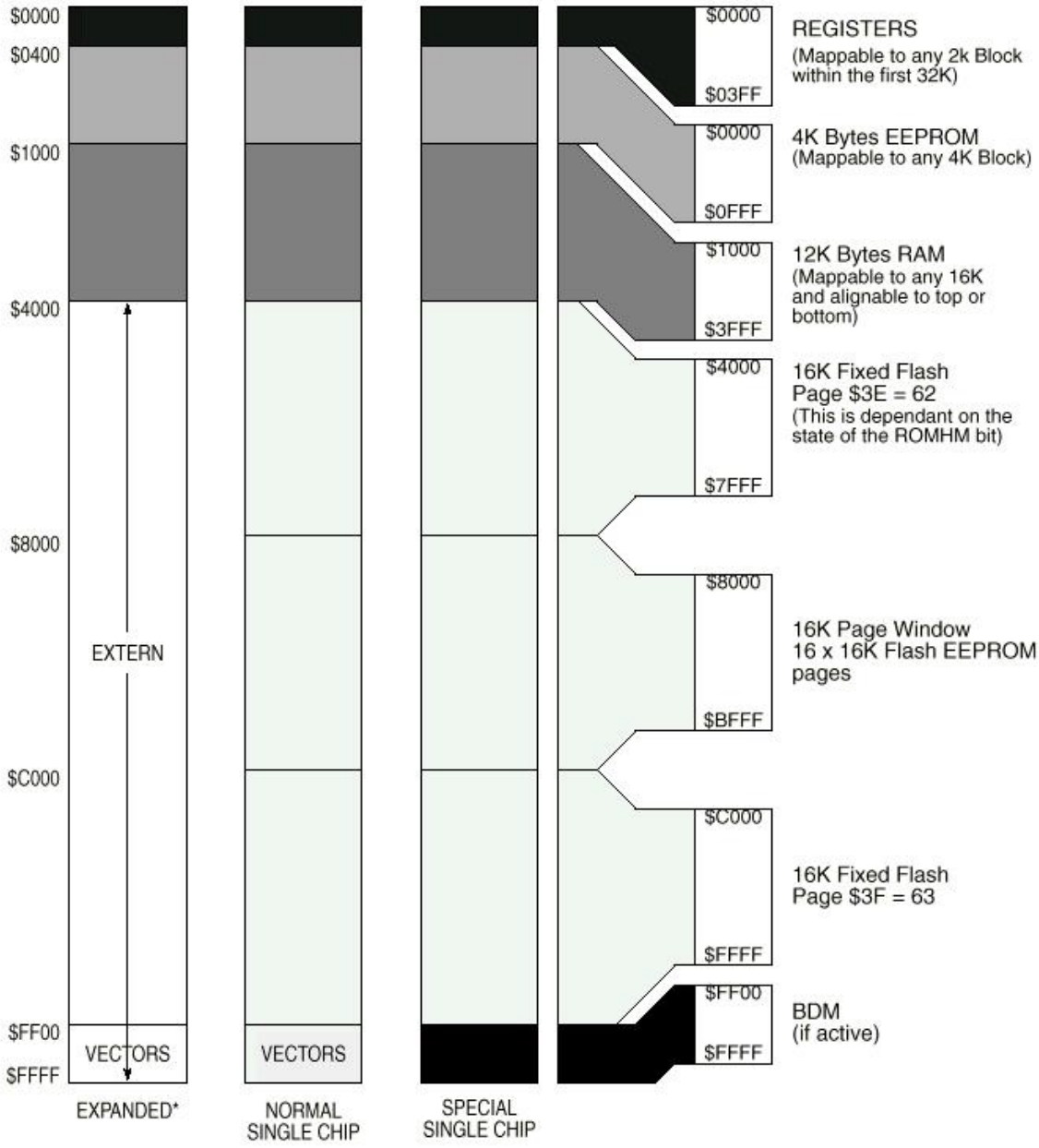


Fig 1-1: MC9S12DG256 Memory map

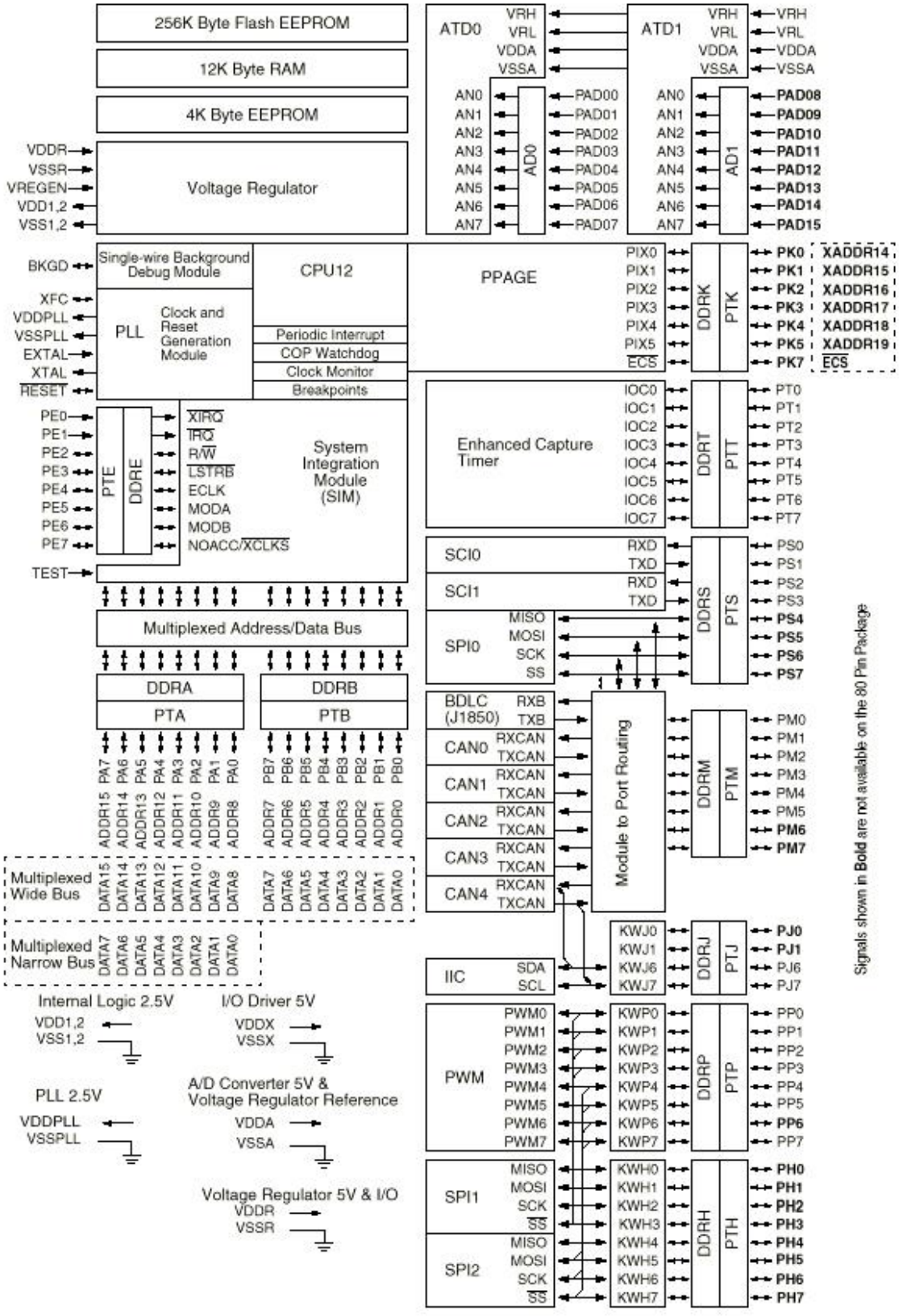


Fig 1-2: MC9S12DG256 MCU block diagram

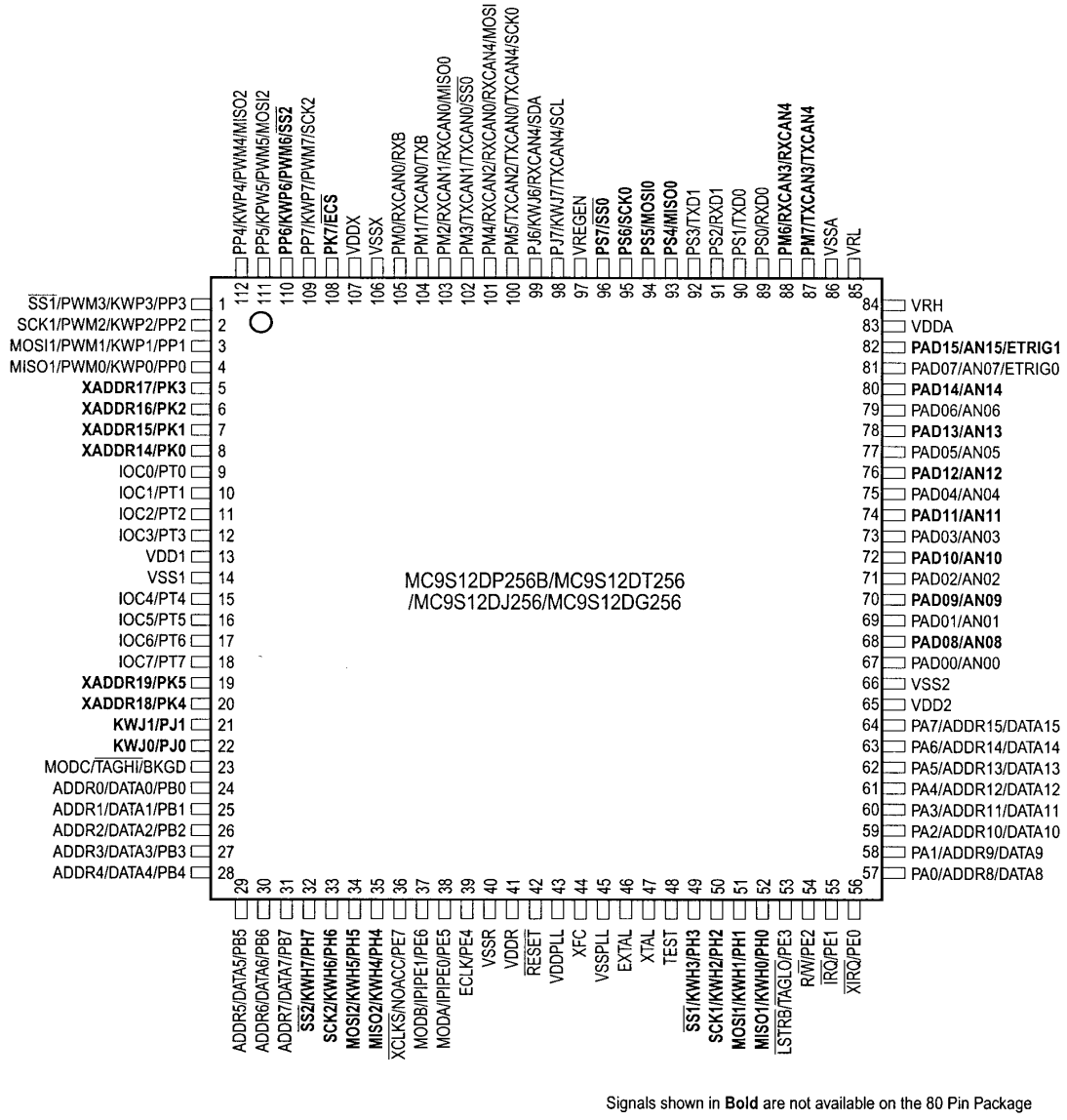


Fig 1-3: MC9S12DG256 MCU pin assignments

1.3 On-board hardware features:

The Dragon12-Plus-USB board includes the following features:

1. On-board USB interface selectable for SCI0 or SCI1
2. RGB color LED
3. RS485 communication port
4. DS1307 RTC with backup battery included for testing I^2C interface
5. I^2C expansion port for interfacing external I^2C devices
6. CAN port
7. SPI expansion port for interfacing external SPI devices
8. Dual 10-bit DAC for testing SPI interface and generating analog waveforms
9. Four robot servo controllers with terminal block for external 5V
10. Four digit 7-segment LED display for learning multiplexing technique

11. Eight LEDs
12. Eight-position DIP switch
13. Four push button switches
14. 5V regulator with DC jack and terminal block for external 9V battery input
15. Speaker to be driven by timer, or DAC or PWM signal for alarm or music applications.
16. Dual H-Bridge motor driver with motor feedback or rotary encoder interface for controlling two DC motors or one Stepper motor
17. Power-On LED indicator
18. IR transceiver with on-board 38KHz oscillator
19. BDM-in connector to be connected with a BDM from multiple vendors for debugging
20. BDM POD mode for programming other HCS12 boards. No extra hardware needed
21. Opto-coupler output
22. Logic probes with LED indicators
23. Abort switch for stopping program when program is hung in a dead loop
24. Mode switch for selecting 4 operating modes: EVB, Jump-to-EEPROM, BDM POD and Bootloader
25. 4 X 4 keypad
26. Form C relay output rated at 3A/30V or 1A/125V
27. Relay-On LED indicator
28. X-Y-Z accelerometer interface or GP2-D12 distance measuring sensor interface for distance measurement
29. Potentiometer trimmer pot for analog input
30. Temperature sensor
31. Communication port for VGA camera with built-in JPEG compression. (Camera is optional)
32. Light sensor
33. Female and male headers provide shortest distance (great for high speed applications!) from bread board to every I/O pin of the MC9S12DG256
34. PC board size is 8.4" X 5.35"

The Dragon12-Plus-USB board has the following features as options:

35. RF transmitter
36. RF receiver
37. SD memory and VGA camera interfaces
38. VGA camera with JPEG compression

The Dragon12-Plus-USB board includes the following features, but not shown in the picture on the front page of this manual:

1. A 16X2 LCD display module with LED backlight is included for learning LCD interface software, but not shown in the picture on the front page. It can be replaced by any size of LCD display module via a 16-pin (8X2) cable assembly.

2. A solderless breadboard is included for fast prototyping, but not shown in the picture on the front page.

1.4 I/O Pin Usage

Many I/O pins of the MC9S12DG256 on the Dragon12-Plus-USB board are used by on-board peripherals and it seems that there are only a few of unused pins left for your circuits on the breadboard. Fortunately, it's unlikely that all on-board peripherals will be used by one application program. So the I/O pins on unused peripheral devices can still be used by your circuits on the breadboard. For instance, if you don't touch the 4x4 on-board keypad, the entire port A will be available to your circuits. If you don't use the LCD or just unplug the LCD, the port K will be available as well. Port B drives LEDs, but if you ignore the status of the LED, the port B can drive any other I/O devices on the breadboard. Each pin in port H reads a switch, but it still can be used as an input for reading a TTL or CMOS output from your circuits.

Pin Name	Pin #	I/O Usage
PA0 (output)	Pin 57	Col_0 of keypad
PA1 (output)	Pin 58	Col_1 of keypad
PA2 (output)	Pin 59	Col_2 of keypad
PA3 (output)	Pin 60	Col_3 of keypad
PA4 (input)	Pin 61	Row_0 of keypad
PA5 (input)	Pin 62	Row_1 of keypad
PA6 (input)	Pin 63	Row_2 of keypad
PA7 (input)	Pin 64	Row_3 of keypad
PB0 (output)	Pin 24	LED0 or H-bridge
PB1 (output)	Pin 25	LED1 or H-bridge
PB2 (output)	Pin 26	LED2 or H-bridge
PB3 (output)	Pin 27	LED3 or H-bridge
PB4 (output)	Pin 28	LED4
PB5 (output)	Pin 29	LED5
PB6 (output)	Pin 30	LED6
PB7 (output)	Pin 31	LED7
PE0 (input)	Pin 56	Abort switch SW8
PE1	Pin 55	not used
PE2 (output)	Pin 54	Relay
PE3 (output)	Pin 53	Opto-coupler
PE4	Pin 39	not used
PE5	Pin 38	not used
PE6	Pin 37	not used
PE7	Pin 36	not used
PH0 (input)	Pin 52	DIP switch 1 or pushbutton switch SW5
PH1 (input)	Pin 51	DIP switch 2 or pushbutton switch SW4 (input)
PH2 (input)	Pin 50	DIP switch 3 or pushbutton switch SW3 (input)
PH3 (input)	Pin 49	DIP switch 4 or pushbutton switch SW2 (input)
PH4 (input)	Pin 35	DIP switch 5 (input)
PH5 (input)	Pin 34	DIP switch 6 (input)
PH6 (input)	Pin 33	DIP switch 7 (input)
PH7 (input)	Pin 32	DIP switch 8 (input)
PJ0 (output)	Pin 22	DIR of RS485
PJ1 (output)	Pin 21	LED enable
PJ6	Pin 99	SDA for DS1307(U11) or external I2C (J2)
PJ7	Pin 98	SCL for DS1307(U11) or external I2C (J2)
PK0 (output)	Pin 8	RS of LCD module
PK1 (output)	Pin 7	EN of LCD module
PK2	Pin 6	DB4 of LCD module (bi-directional)
PK3	Pin 5	DB5 of LCD module (bi-directional)
PK4	Pin 20	DB6 of LCD module (bi-directional)
PK5	Pin 19	DB7 of LCD module (bi-directional)
PK7 (output)	Pin 108	R/W of LCD module

Table 1-1: I/O pin usage list 1

Pin Name	Pin #	I/O Usage
PM0	Pin 105	CAN0
PM1	Pin 104	CAN0
PM2	Pin 103	Write Enable for SD memory
PM3	Pin 102	Card detect for SD memory
PM4	Pin 101	CS of SD memory
PM5	Pin 100	not used
PM6	Pin 88	CS of LTC1661 (DAC)
PM7	Pin 87	I/O for external SPI (J10)
PP0 (output)	Pin 4	Digit 3 of 7-segment display or EN12 of H-bridge
PP1 (output)	Pin 3	Digit 2 of 7-segment display or EN34 of H-bridge
PP2 (output)	Pin 2	Digit 1 of 7-segment display
PP3 (output)	Pin 1	Digit 0 of 7-segment display
PP4 (output)	Pin 112	Servo motor 1 or RGB LED
PP5 (output)	Pin 111	Servo motor 2 or RGB LED
PP6 (output)	Pin 110	Servo motor 3 or RGB LED
PP7 (output)	Pin 109	Servo motor 4
PS0	Pin 89	SCI0 for PC communication, RECV (DB9 connector P1)
PS1	Pin 90	SCI0 for PC communication, XMIT (DB9 connector P1)
PS2	Pin 91	SCI1 for user applications, RECV, selected by J23
PS3	Pin 92	SCI1 for user applications, XMIT
PS4	Pin 93	MISO for LTC1661, SD memory interface and external SPI (J10)
PS5	Pin 94	MOSI for LTC1661, SD memory interface and external SPI (J10)
PS6	Pin 95	SCLK for LTC1661, SD memory interface and external SPI (J10)
PS7	Pin 96	I/O for external SPI (J10)
PT0 (input)	Pin 9	Rotary encoder
PT1 (input)	Pin 10	Rotary encoder
PT2	Pin 11	not used
PT3 (input)	Pin 12	IR RECV when jumpers on J27 set for PT3 and PT4
PT4 (output)	Pin 15	IR XMIT when jumpers on J27 are set for PT3 and PT4
PT5 (output)	Pin 16	Speaker (output)
PT6 (output)	Pin 17	BDMout reset (used in POD mode only)
PT7	Pin 18	BDMout data line (bi-directional, used in POD mode only)
PAD0	Pin 67	D-bug12 mode select, SW7
PAD1	Pin 69	D-bug12 mode select, SW7
PAD2	Pin 71	Alarm trigger1, analog or digital input
PAD3	Pin 73	Alarm trigger2, analog or digital input
PAD4	Pin 75	Light sensor (phototransistor Q1)
PAD5	Pin 77	Temperature sensor (U14, MCP9701A)
PAD6	Pin 79	Not Used
PAD7	Pin 81	Trimmer pot VR2
PAD8	Pin 68	X axis input for Wytec accelerometer or ADC input for GP12D2
PAD9	Pin 70	Y axis input for Wytec accelerometer or ADC input for GP12D2
PAD10	Pin 72	Z axis input for Wytec accelerometer or ADC input for GP12D2
PAD11	Pin 74	not used
PAD12	Pin 76	not used
PAD13	Pin 78	not used
PAD14	Pin 80	not used
PAD15	Pin 82	not used

Table 1-2: I/O pin usage list 2

Chapter 2. Quick Start

By default the Dragon12-Plus-USB board is pre-installed with the bootloader (Freescale AN2153.pdf) and the D-Bug12 monitor (Freescale DB12RG4.pdf). In chapters 2 and 3 the AsmIDE is used as the main software tool to develop and debug assembly programs. If you prefer to use Code Warrior IDE for program development and your board is pre-installed, per your request, with the serial monitor (Freescale AN2548.pdf), **skip the chapters 2 and 3 after installing software from CD.**

People often use different terminologies. In our product manuals, **Download** means to transfer a file from PC to a development board, while **Upload** means to transfer a file from a development board to PC. Through out the manual, **left click** means that you click the left button of the mouse and **right click** means that you click the right button of the mouse.

2.1 Install software:

After downloading software from our web site, the installation is automated by double clicking on the **SETUP.BAT**. It will create a folder c:\Dragon12P\examples and copy all example program files from the CD to c:\Dragon12P\examples

If the filename is only shown as **SETUP**, not **SETUP.BA**T, you should change a folder option of the Explorer to show file extension. When a file's extension is hiding, it is hard to know what it is. To have your files to be shown with extensions, click on the TOOL tab in Explorer menu, then click on folder options, then click on view tab, finally un-check the item named 'Hide extensions for knowing file types'.

After the software is successfully installed, you can make a shortcut to AsmIDE.exe on the desktop. It's important to make a shortcut so that its target location is C:\Dragon12P, not c:\Windows\desktop or other locations. First, right click the Start button, then left click "Explorer", left click on C:\Dragon12P, right click on AsmIDE.exe (an application program), left click "Send to" and finally left click "Desktop" (do not click "COPY"). It will create an icon named "shortcut to AsmIDE" on the desktop and you can rename it to Dragon12-Plus-USB. You can double check the target location by right clicking on the icon, then left click on "properties". You should see that the target location is C:\Dragon12P. If you want to make a shortcut for AsmIDE on the Desktop, this is the correct way to do it. If you don't follow this method, your may have a problem running your program. Never drag the AsmIDE.exe to the desktop folder.

The default setting of AsmIDE for the Dragon12-Plus-USB board is created in a text file named c:\Dragon12P\AsmIDE.ini. In the future if you get lost with all the changes, you always can copy this file into the folder named c:\Dragon12P.

2.2 Getting Started (for D-Bug12 monitor only)

To operate the Dragon12-Plus-USB board, follow steps1 through 5 below:

1. Make sure that the both DIP switches of SW7 must be set in the "low" positions for EVB mode, then plug the AC adapter into a wall outlet and plug the DC female plug of the AC adapter into the DC jack on the lower left side of the Dragon12-Plus-USB board. After power up, the PB7-PB0 LEDs should light up from left to right one at a time, the speaker should chirp once (If the chirp is too soft you can remove the sticker on the speaker to increase the volume) and the LCD should display the following message:

   "DRAGON12plus EVB" ; you can display your name on LCD and see details
   "D-Bug12 EVB MODE" ; at CDROM\examples\name_display\readme.txt
   If it does not occur, make sure that the Power-On LED indicator is on. The PWR LED is on when VCC (5V) is present. If the PWR LED is off check the output of the AC adapter. It should be about 9V DC.

2. Plug the USB cable to the USB jack P1 on the **upper left** corner of the Dragon12-Plus-USB board. Plug the other end of the USB cable into a USB port of your PC. Make sure that the jumpers on the J41 and J42 are set correctly for USB interface for the SCI0. The header J43 is the MC9S12DG256's SCI1 port in TTL interface that can be used by a user's application program.

3. To invoke the AsmIDE, you can right click the Start button, then left click "Explorer", left click on C:\Dragon12P and finally, double left click on AsmIDE.exe. If you have created a shortcut icon on the desktop, just double click the AsmIDE icon on the desktop.

   Warning note: Always plug the USB cable into the Dragon12-Plus-USB before invoking the AsmIDE and close the AsmIDE before unplugging the USB cable, Otherwise the AsmIDE may hang up and you need to re-establish the USB link again.

   In case the AsmIDE hangs up, you need to close the AsmIDE first, then pull the USB cable out the USB jack P1, wait for a few seconds before re-plug the USB cable into the P1 and then wait for a few more seconds. After cycling USB connection, then invoke the AsmIDE and the PC may re-establish the USB communication. If this does not work, you need to reset your PC. In order to avoid it, always close the AsmIDE before unplugging the USB cable.

   The AsmIDE is simple and very easy to use. You only need to use three commands from the AsmIDE for your HCS12 development work. Use the File command to edit your source code, the Build->Assemble command to assemble your source code, and the Build->Download command to download an s19 file to the Dragon12-Plus-USB board.

   In the View->Option->Terminal Window Options menu, set the COM port as 1 or 2 to match the COM port number that is assigned to the USB port by Device Manager in control panel. Also, set the COM port options at 9600, N,8,1, and check the "enable the terminal window".

4. After reset, the D-Bug12 monitor defaults baud rate at 9600 and and Hyperbaud function is disabled. If Hyperbaud function is enabled, the Hyperbaud toolbar button sends the BAUD 57600 command to the D-Bug12 monitor, and then it also changes the serial port to the 57600 baud rate. **IMPORTANT:** When you reset your board it will go back to 9600 baud and you will see characters 'aaaaaaaaaa' on the screen. You will need to press the Hyperbaud button once to return AsmIDE to 9600 baud, and press it again to get 57600 baud. To stay at the 57600 baud all the time, you need to press the Hyperbaud button twice after every reset. The Hyperbaud function is disabled by default and it should only be used by an experienced user, not a beginner.

5. You can program text values for function keys to be sent from the terminal window. Some function keys are pre-programmed, but you can change it any time in configuration options (View->Options->Terminal Func Keys).

   In the View->Option->Assembler menu, make sure that the chip family is **68HC12**, not 68HC11. If you would like to use your own assembler, you can replace the as12.exe with the name of your own assembler.

6. The screen is divided into two windows. The top window is for editing your source code and the bottom window is shared by the **message window** and the **terminal window**.

   If the terminal options are set correctly, you should see the following prompt every time the reset button on the Dragon12-Plus-USB board is pressed. If you do not see this, the bottom window may be set for message window. Sometime it's a little confusing when terminal window is disabled and the message window does not display what you have typed. In order to enable terminal window you have to click the terminal button in the bottom window to enable the terminal window display, then move the cursor to any location in the terminal

window and click the left button on the mouse. After seeing a solid block cursor flashes, press the <Enter> key and it will enable the terminal window.

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"
>
```

Warning note: If you see the above message, but you cannot type in any character on keyboard then the jumper on J41 is probably installed in the "UP" position. In order to use USB interface, the jumper on J41 must be installed in the "LOW" position.

2.3 Test Hardware:

To help users get up and running, the Dragon12-Plus-USB board comes with many fully debugged and ready-to-run sample programs including source code. The hardware test program, test.asm, simultaneously scans the keypad, plays a song, multiplexes the 4 LED seven segment display, changes display brightness by adjusting the trimmer pot and detects an object by using the IR transceiver as a proximity sensor.

All sample programs must be run from RAM in EVB mode. In order to run the test program in EVB mode, the both DIP switches of SW7 must be set in the "low" positions to match the picture above the SW7.

The steps to run your first sample program are as follows:

1. Click the File button to open the test.asm from c:\Dragon12P\examples. After the test.asm is loaded into the AsmIDE window, you can view instructions of how to test all hardware on the Dragon12-Plus-USB board.
2. Click the Build button to assemble code and generate the test.s19 file. This is how you normally generate an s19 file. You can omit this step, because the test.s19 is already on your hard disk.
3. Press the reset button on the board, you will see:

   ```
   D-Bug12 v4.0.0b32
   Copyright 1996 - 2005 Freescale Semiconductor
   For Commands type "Help"
   >
   ```

4. Type "LOAD", then hit <Enter> key.
5. Click the Build button. Select Download option and locate the file 'test.s19' for downloading. If it prompts you with the "save changes?" message, you can ignore that message and click the "No" answer.
6. After download is done, type "G 2000" and hit <Enter> key to run the test program.

All sample programs on the CD are developed in RAM. You can try to run a different example program later after you have finished reading this manual. You should always press the reset button before downloading a new program, because the new program may not work if an interrupt was enabled by a previous program.

All example programs are fully debugged, so the assembler won't generate an error. If you have an error, even a warning error, in your program, you must correct it before it can generate an s19 file.

Chapter 3. Software descriptions

3.1 Bootloader and D-Bug12 Monitor

The MC9S12DG256 on the Dragon12-Plus-USB board is pre-loaded with bootloader and D-Bug12 monitor firmware and it will operate in 4 different modes depending on the setting of the 2-position DIPswitch, SW7. After power up or reset, the MC9S12DG256 will read the PAD0 and PAD1 to decide which mode to boot up.

The bootloader (**AN2153.PDF)**, the D-Bug12 reference guide (**DB12RG4.PDF**) and the MC9S12DG256 data book **(MC9SDG256.PDF**) are the most important documentations. They can be found on the folder named C:\Dragon12P\document after software installation. The HCS12 instruction set, register map and memory map can be found on page 26, 65 and 120 of the data book, respectively.

The new D-Bug12 V4.x is much different and much larger (about 60K) than old D-Bug12 V2.x. The $C000-$EFFF are just a part of the monitor, In 16-bit S1 record they are $C000-$EFFF. In 24-bit S2 record, they are $FC000-FEFFF (ppage=$3F). Since the ppage register deals with the 16K window $8000-$BFFF the addresses $C000-$FFFF are not affected by the ppage. The other part of the monitor is at C0000-C87FF (16K window $8000-$BFFF when ppage=$30,$31 and $32). See details on page 20 of the app note AN2153 or page 71 of the D-Bug12 v4 reference guide on the CD.

3.1.1 EVB mode: PAD1=0, PAD0=0.

This is the standard debug environment running on the MC9S12DG256 for on-chip RAM or EEPROM based code development. Using an IDE program to view and modify registers and memory locations, you may set breakpoints, single step through programs, and assemble and disassemble code as you would in a BUFFALO monitor based Freescale 68HC11 EVB. It gives you 12K RAM and 3K EEPROM to develop and debug your code. You must place your interrupt vectors at $3E00-$3E7F, because real interrupt vector addresses are taken by bootloader, bootloader and D-Bug12 monitor will redirect interrupts to the RAM interrupt vector table at $3E00-$3E7F.

After booting up in this mode, the LCD should display the following message:

“DRAGON12plus EVB”
“D-Bug12 EVB MODE”

and you should see the following message on PC screen:

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"
>
```

Typing “help” then <Enter> will display a list of available commands.

In this mode, you **cannot** erase or program on-chip flash memory.

If the D-Bug12 monitor is erased, the LCD will display the following message after reset:

“DRAGON12plus EVB”
“ D-Bug12 ERASED ”

You can use bootloader to re-program D-Bug12 monitor into flash memory.

Note: Some user may accidentally erase D-Bug12 monitor in bootloader mode, so it is important to know how to re-program D-Bug12 monitor in bootloader mode.

3.1.2 **Jump-to-EEPROM mode:** PAD1=0, PAD0=1

This mode enables the MC9S12DG256 to jump directly to the internal EEPROM at location $0400 upon reset.

This mode makes the MC9S12DG256 a replacement for the old 68HC811E2 microcontroller, but it also gives you 3K EEPROM instead of 2K EEPROM with the 68HC811E2. The bus speed is 4MHz, one half of the crystal frequency by default, the PLL function must be initialized by user's code for a higher bus speed, because the D-Bug12 monitor firmware that boosts bus speed to 24 MHz is bypassed. If you need to auto start your code upon reset, the procedure is available in the folder named eeprom_programming.

After booting up in this mode, the LCD should display the following message:

"DRAGON12plus EVB"
" JUMP TO EEPROM "

3.1.3 **BDM POD** mode: PAD1=1, PAD0=0

In this BDM POD mode, the D-Bug12 firmware acts as a master to access all target MCU resources on the target board (another Dragon12-Plus-USB board) via the BDM port in a non-intrusive manner. It becomes a BDM that will have all the features that a standard BDM has in debugging the target MCU. Also, it gains all the features a programmer has for programming the flash memory of the MCU on the target board (another Dragon12-Plus-USB board).

To use the master board as a programmer, you need a 6-pin ribbon cable to connect from the BDM OUT of the master board to the BDM IN of the target board (make sure that the orientation of the cable is correct). You don't have to provide the power to both boards, but only to one board. The master board communicates to a PC COM port while the target board does not need to be connected to a PC COM port.

After booting up in this mode, the LCD should display one of the following two messages:

If the D-Bug12 monitor is erased, the LCD will display the following message after reset:

"DRAGON12plus EVB"
"POD-Bug12 ERASED"

Otherwise it will display:

"DRAGON12plus EVB"
" BDM POD MODE "

and you should see the following message on PC screen:

```
Can't Communicate With Target CPU

1.) Set Target Speed (48000 KHz)
2.) Reset Target
3.) Reattempt Communication
4.) Erase & Unsecure
?
```

You first must set the target speed with the choice 1). After entering the first choice, you will be prompted to enter the target speed. It's the crystal frequency, not the bus speed that is boosted up by the on-chip PLL. After a reset, before the PLL is enabled, the target

MC9S12DG256 is running from the crystal frequency, not the PLL frequency. Enter 8000 for the target speed. After the correct speed is entered, the master will try to communicate with the target board. If it's not successful, enter choice 2) to reset the target board.

Note: The newer D-Bug12 monitor in POD mode may auto-detect the crystal frequency of a target board, so most likely the step 1 may not be needed.

```
Can't Communicate With Target CPU

1.) Set Target Speed (8000 KHz)
2.) Reset Target
3.) Reattempt Communication
4.) Erase & Unsecure
? 1

Enter Target Crystal Frequency (kHz): 8000


Can't Communicate With Target CPU

1.) Set Target Speed (8000 KHz)
2.) Reset Target
3.) Reattempt Communication
4.) Erase & Unsecure
? 2
```

When the communication is established, you will see the following:

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"

S>
```

You will notice that the debug prompt is "S>" in the POD mode, not just a ">" in the EVB mode. The S> tells that this is the POD mode and the MC9S12DG256 on target (slave board) is stopped. Sometimes the prompt could be a "R>" that means the target MCU is running. If you see the "R>", just type "reset" then <Enter> to reset the target and it will come back to the "S>" prompt.

```
R>Reset <Enter>
S>
```

Note: The initial communication in POD mode does not always work smoothly and sometimes the PC screen would only display an incomplete sign-on message. You need to re-start it all over again by pressing reset buttons on both master board and target board, then press the Enter key on PC keyboard. You cannot go to the next step until PC screen shows the prompt 's>'.

In order to program the flash memory, you have to erase it by using the FBULK command.

```
S>fbulk <Enter>
S>
```

When the prompt "s>" returns, the FBULK command has already erased all of the flash memory contents of the target MC9S12DG256 including the bootloader. If it returns with a message "Flash or EEPROM Failed To Erase" the MC9S12DG256 is defective.

Now we are going to program the bootloader and the D-Bug12 into the flash memory of the target MC9S12DG256.
Before we actually program the flash memory, we must understand there are two different types of s-record file that can be generated by compilers and assemblers.

An s1-record uses a 16-bit starting address field while an s2-record uses a 24-bit starting address field.
An s1-record file looks like this:

```
S123FFA0F64CF650F654F658F65CF660F664F668F66CF670F674F678F67CF680F684F6883D
S123FFC0F68CF690F694F698F69CF6A0F6A4F6A8F6ACF6B0F6B4F6B8F6BCF6C0F6C4F6C81D
S123FFE0F6CCF6D0F6D4F6D8F6DCF6E0F6E4F6E8F6ECF6F0F6F4F6F8F6FCF700F704F00009
S9030000FC
```

An s2-record file looks like this:

```
S2240FEFA0DB70DB66DB5CDB52DB48DB3EDB34DB2ADB20DB16DB0CDB02DAF8DAEEDAE4DADA41
S2240FEFC0DAD0DAC6DABCDAB2DAA8DA9EDA94DA8ADA80DA76DA6CDDD0DA62DA58DA4EDA4494
S2240FEFE0DA02DA0ADA12DA1ADA22DA2ADA32DA3AD9FAD9F2D9AFD98AD9D5EF00EF00EF0039
S9030000FC
```

We are not going to explain the s-record format here. If you would like to know more on the subject, you can review the D-Bug12 reference guide on the CDROM (**BD12RG4.PDF**). It explains the subject in great details. Right now, all you need to know is that an s1-record file must be converted to an s2-record file before using the FLOAD command. The "FLOAD" command in the D-Bug12 is for downloading an s2-record file.

Our Dragon12 bootloader is modified from the Motorola's BootDP256.asm. We added our modification to the original source code and the s record file is generated by the AsmIDE. It's an s1-record file and we converted it into an s2-record file by using the following commands:

Sreccvt –m c0000 ffff 32 –of f0000 -o Boot_DR12_8MHz.s29 Boot_DR12_8MHz.s19

Now we type "FLOAD" <Enter> at the prompt. Click the Build button, select the Download option, and select the file named **Boot_DR12P_8MHz.s29** located in the folder named "D-Bug12_Monitor ". You should see the following on the terminal window when programming is done (when the prompt "s>" appears):

```
S>fload <Enter>

**************************************************************************
S>
```

Now we are going to program the D-Bug12 monitor into the flash memory. We need to type "FLOAD" <Enter> at the prompt. Click the Build button, select the Download option, and select the file named **DBug12v32_DR12P_8MHz** located in the folder named "D-Bug12_Monitor". You should see the following on the terminal window when programming is done (when the prompt "s>" appears):

```
S>fload <Enter>

**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**********************************************************
S>
```

With the bootloader and the D-Bug12 programmed in the flash memory, the target board now becomes a true development board. That's how we program the board before we ship it. Your Dragon12-Plus-USB board actually becomes a programmer. You can then repeat above steps as many times as you want. Just unplug the 6-pin BDM cable from the target board, and then plug it into a new target board to program its flash memory with these two files. You even don't have to turn off the power while doing this.

For your convenience, we combined both the bootloader and the D-Bug12 monitor into a single s2 file named **Boot_ DBug12v32_DR12P_8MHz .s29**. In case you need to update both of them, you can download this combined file.

The D-Bug12 monitor is an application program runs from the bootloader. If you program the D-Bug12 portion of flash memory with your application program, your program will run automatically in EVB mode after power up or reset. When running your code instead of the D-Bug12 monitor, the bus speed is 4MHz, one half of the crystal frequency by default. The PLL function must be initialized by your code for a higher bus speed, because the D-Bug12 monitor firmware was not in flash memory anymore. For your convenience, we include a PLL code template in chapter 7.

If you need to auto start your code upon reset, the procedure is available in the folder named flash_programming.

3.1.4 **BOOTLOADER** mode: PAD1=1, PAD0=1

This bootloader allows you to erase/program flash memory and erase EEPROM. It is mainly used to program the D-Bug12 monitor into flash memory or download a user's fully debugged code into the D-Bug12 portion of flash memory. The latter allows the board to be operated in EVB mode and start your code every time the board is turned on or reset.

When you program your code into the D-Bug12 portion of flash memory, it wipes out the D-Bug12 monitor. You can restore it any time, just as if you were downloading another application program since the bootloader is not erased. You can erase and program the D-Bug12 monitor portion of the flash memory of the MC9S12DG256 on its own board in bootloader mode, but you cannot erase and program bootloader by itself. **The bootloader can only be erased by an external BDM via BDM-in port.**

After booting up in this mode, the LCD should display the following message:

"DRAGON12plus EVB"
" BOOT LOADER "

and you should see the bootloader menu on PC screen:

MC9S12DG256 bootloader menu:

a) Erase Flash
b) Program Flash
c) Set Baud Rate
d) Erase EEPROM
?

The option a) will erase the D-Bug12 portion of flash memory, not the bootloader itself.
The option b) will program the D-Bug12 portion of flash memory, not the bootloader itself.

The file to be programmed into flash memory must be an s2-record file. If your assembler and compiler generate s1-record files only, you must convert an s1-record file to an s-2 record file before programming flash memory with the bootloader.

The option c) will set a new baud rate.
The option d) will erase all on-chip EEPROM.

Note: Some users may accidentally erase the D-Bug12 monitor when entering this mode, so it is important to know how to re-program the D-Bug12 monitor.

To program flash memory with the D-Bug12 monitor:

1. Enter the option a) to erase D-Bug12 portion of flash memory. Wait until the bootloader menu re-appears after flash memory is erased.
2. Enter the option b), the bootloader will wait for your file. **Do not type** any thing on keyboard.
3. Click the Build button, select the Download option, and select the file named **DBug12v32_DR12P_8MHz .s29** located in the folder named "D-Bug12_Monitor" for downloading. You should see the following on the screen:

   ```
   ********************************************************************************
   ********************************************************************************
   ********************************************************************************
   ********************************************************************************
   ************************************
   ```

4. It will take 3 minutes to program the D-Bug12 at 9600 baud rate and the bootloader menu will reappear after the D-Bug12 monitor is successfully programmed into flash memory.

3.2 Making a simple assembly program in RAM:

We are using AsmIDE as a terminal program and the following instructions to create your first assembly program. If you are using a different terminal program, the instructions may vary.

The steps to create your first program are as follows:

1. Click the **File** button to open a new file.

   In assembly language, you specify the starting address of your CODE by an ORG statement.

   You can start the data RAM at address $1000 with the statement org $1000 followed by RAM variables, as shown by:

   ```
            org     $1000

   count:   rmb     1           ; reserve one byte of RAM for temp storage
   temp:    rmb     2           ; reserve two bytes of RAM for temp storage
   ```

   If your program is small, say less than 4K, you can start your program at address $2000 with the statement org $2000 followed by your program, as shown by:

   ```
            org     $2000
   ```

   It will assemble your source program and generate hex code within 4K locations from $2000 to $2FFF.

   Here is a very simple program, but it's complete. It will flash the PB0 LED at 2Hz when it's running. The RAM byte named 'counter' is added for demonstrating how a RAM data byte is used in a user program. In this simple program it's not really necessary, because the accumulator A can be used as the RAM byte 'counter'.

For a good programming practice, you should always place the lds instruction in the first line of your code.

```
#include    reg9s12.h
REGBLK:     equ     $0000
STACK:      equ     $2000           ; do not use $4000
;
            org     $1000
counter:    rmb     1

            org     $2000           ; program code
start:      lds     #STACK
            ldx     #REGBLK

            ldaa    #$ff
            staa    ddrj,x          ; make port J an output port
            staa    ddrb,x          ; make port B an output port
            staa    ddrp,x          ; make port P an output port
            staa    ptp,x           ; turn off 7-segment LED display

            clr     ptj,x           ; make PJ1 low to enable LEDs
back:       clr     portb,x         ; turn off PB0
            jsr     d250ms          ; delay 250ms
            inc     portb,x         ; turn on PB0
            jsr     d250ms          ; delay 250ms
            jmp     back
*
d250ms:     ldaa    #250            ; delay 250 ms
            staa    counter

delay1:     ldy     #6000           ; 6000 x 4 = 24,000 cycles = 1ms
delay:      dey                     ; this instruction  takes 1 cycle
            bne     delay           ; this instruction  takes 3 cycles
            dec     counter
            bne     delay1          ; not 250ms yet, delay again
            rts

            end
```

2. Click File button, select Save option to save your assembly source file. Save your file frequently while editing. If you are creating a new file and giving the file a name to save, enter the file name including file extension, such as “Flash_PB0.asm”, not just “Flash_PB0”.

3. Click Build button, select Assemble option, or click the assembler button on the toolbar to assemble your code and generate an s19 file. If the assembler detects an error, the error message will show the line numbers of your source code that caused the error. You have to correct all errors in your program.

4. Go to the line and correct the errors and go back to step 3 until there are no errors.

5. Press the reset button on the board, you will see:

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"
>
```

6. Type "LOAD" and then hit <Enter> key

7. Click Build button, select Download option and locate the file named 'Flash_PB0.s19"' for downloading. After download is done, type "G 2000" and hit <Enter> key to run the program.

For your convenience, we have included this sample program on the CD.

3.3 Software development

3.3.1 Use on-chip 12K RAM for software development in EVB mode.

You can download your s19 file into the RAM and debug it with the D-Bug12 monitor in this mode. You must place your interrupt vectors at $3E00-$3E7F, because real interrupt vector addresses are taken by the bootloader. The bootloader and the D-Bug12 monitor will redirect interrupts to the RAM interrupt vector addresses at $3E00-$3E7F

Because RAM will lose its contents after power off, you have to load your program every time after power-up. In the beginning of your program, you must initialize the interrupt vectors at $3E00-$3E7F.

In all sample programs, the user program code locations are at $2000-$3FFF. The user data RAM locations are at $1000-$1FFF. The 64 RAM interrupt vector addresses are at $3E00-$3E7F.

The 64 RAM interrupt vector addresses (128 bytes of RAM) are assigned by the D-Bug12 monitor to different interrupt sources. The listing of interrupt sources is show on chapter 8.

3.3.2 Use on-chip 3K EEPROM for testing your code in EVB mode.

If your program is small enough to fit into a 3K range, then you can download your code into the EEPROM. In this way, your program can be auto started from $0400 upon reset. You cannot set software breakpoints and single step in the EEPROM in EVB mode, so it makes sense to do development work in the RAM. When your code is completely debugged, then re-assemble or re-compile it at $0400 and download the final s19 file into the EEPROM for the auto start feature.

Like the RAM-based development, your interrupt vectors are at $3E00-$3E7F. In the beginning of your program, you must initialize the interrupt vectors at $3E00-$3E7F.

3.3.3 Use on-chip flash for testing your code in BOOTLOADER mode.

In this mode, you download your program directly into on-chip flash memory. You first erase the D-Bug12 monitor portion of flash memory, and then program that portion of the flash memory by downloading your application program code in an s29 file. Your program will replace the D-Bug12 monitor in the flash memory. The bootloader portion of the flash memory remains intact. To run your code, set the mode switch SW 7 to EVB mode, then press the reset button. It usually runs the D-Bug12 monitor, but now it runs your program. The flash memory is non-volatile like the EEPROM. Your code will run every time the board is turned on or reset.

The bootloader redirects interrupts to $EF80-$EFFF. The D-BUG12 is not present and the interrupt vectors of your program are at $EF80-$EFFF. The addresses $EFFE and $EFFF contains the starting address of your program.

In order to program the MC9S12DG256 flash memory, you must program an even number of bytes and begin on an even address boundary for each s-record. If any one s-record in the file contains an odd number of bytes or begins with an odd address, the flash memory cannot be programmed. If your assembler or compiler cannot generate the even format, you must use the Freescale s-record conversion utility **sreccvt.exe** to convert your odd format to the even format by using the following command line:

Sreccvt –m c0000 fffff 32 –of f0000 –o test.s29 test.s19

It will create a new file named test.s29 that has the even format and can be programmed into flash memory. For your convenience, the sreccvt.exe is included in the folder named CDROM\document\Sreccvt-GUI.

The crystal frequency is 8 MHz and usually it will result in a 4 MHz bus speed, but on this board the MC9S12DG256's internal PLL boosts the bus speed up to 24 MHz.

The circuit is designed in such way that the value of all resistors and capacitors are not critical, with the exception of R10 and C36, which determine the 38KHz for IR transmitter.

4.1 LEDs:

Each port B line is monitored by a LED. In order to turn on port B LEDs, the PJ1 (pin 21 of MC9S12DG256) must be programmed as output and set for logic zero.

4.2 DIP switch and pushbuttons:

Port H is connected to an 8-position DIP switch. The DIP switch is connected to GND via the RN9 (eight 4.7K resistors), so it's not dead short to GND. When port H is programmed as an output port, the DIP switch setting is ignored, but for the best result all 8 DIP switches should be open (at the up positions).

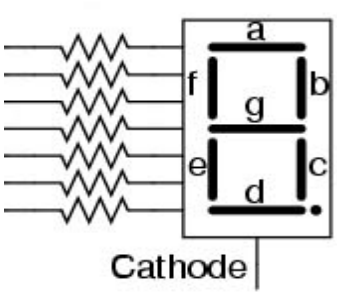
4.3 7-Segment LED multiplexing

There are 4 digits of 7-segment LEDs on the Dragon12-Plus-USB board. The type of the 7-segment LED on board is called common cathode. In an individual digit, all anodes are driven individually by an output port and all cathodes are internally connected together.

Before sending a number to a 7-segment LED, the number must be converted to its corresponding 7-segment code depending how the 7-segment display is connected to an output port.

The Dragon12-Plus-USB board uses port B to drive 7-segment anodes and uses PP0-PP3 to drive common cathodes. We will explain how to multiplex 7-segment by displaying the number 1234 on the display.

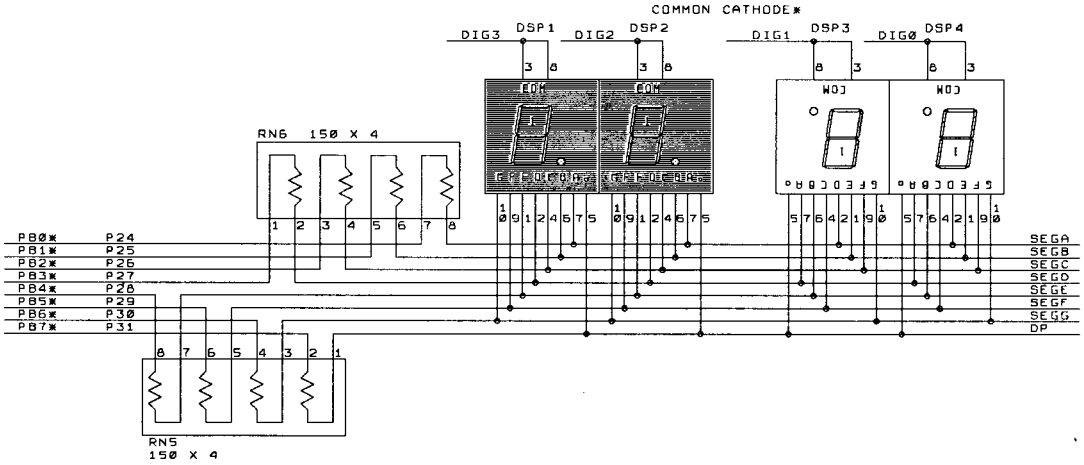
By convention, the 7segments are called segment A, B, C, D, E, F and G. Their locations in the display are shown below:



The segment A, B, C, D, E, F, G and Decimal Point are driven by PB0, PB1, PB2, PB3, PB4, PB5, PB5 and PB7, respectively. The hex value of the segment code is shown in the following table:

Number	DP	G	F	E	D	C	B	A	Hex Value
1	0	0	0	0	0	1	1	0	$06
2	0	1	0	1	1	0	1	1	$5B
3	0	1	0	0	1	1	1	1	$4F
4	0	1	1	0	0	1	1	0	$66

The schematic for multiplexing 4 digits is shown below. The two of the digits at the right are deliberately placed upside down and the hardware connections compensate for this configuration. The reason for the upside down digits is to place two decimal pointers on the middle as a colon for a clock display.



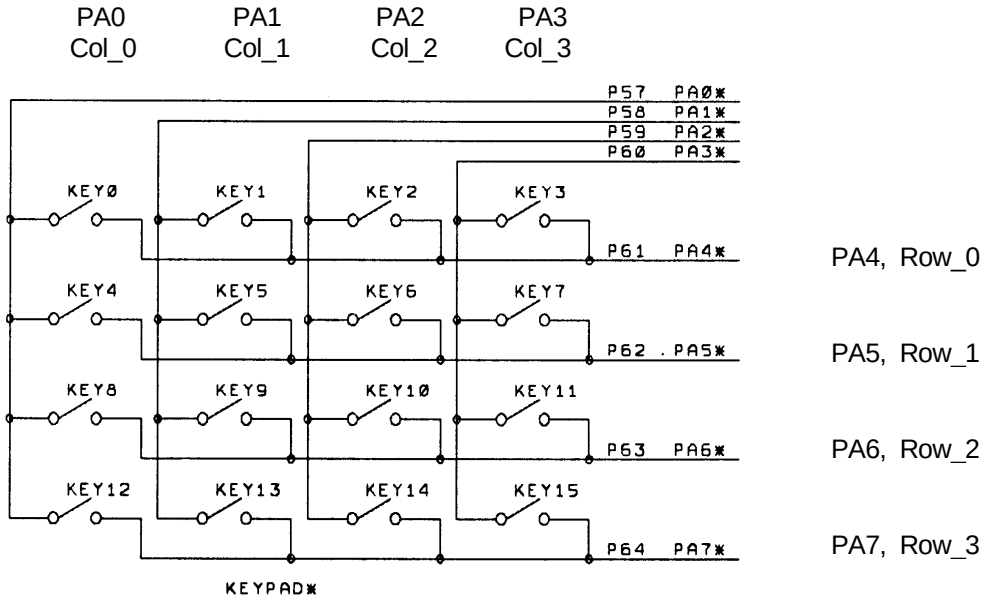
The digit 3, 2, 1, and 0 are driven by PP0, PP1, PP2 and PP3, respectively. The 7-segment LED is turned on one at a time at 250 Hz refresh rate. It's so fast that our eyes will perceive that all 4 digits are turned on at the same time. To display the number 1234 on the 7-segment display, the following steps should be taken:

1. Output $06 to port B, set PP0 low and PP1, PP2, and PP3 high. The number 1 is shown on the digit 3 (the leftmost digit), but other 3 digits are turned off.
2. Delay 1ms.
3. Output $5B to port B, set PP1 low and PP0, PP2, and PP3 high. The number 2 is shown on the digit 2, but other 3 digits are turned off.
4. Delay 1ms.
5. Output $4F to port B, set PP2 low and PP0, PP1, and PP3 high. The number 3 is shown on the digit 1, but other 3 digits are turned off.
6. Delay 1ms.
7. Output $66 to port B, set PP3 low and PP0, PP1, and PP2 high. The number 4 is shown on the digit 0 (the rightmost digit), but other 3 digits are turned off.
8. Delay 1ms.
9. Go back to step 1.

4.4 Keypad:

Port A is an 8-bit bi-directional port. Its primary usage is for a 4X4 keypad. If the port is not used for the keypad, it can be used as a general-purpose I/O.

The schematic for the keypad connections is shown below:



Keypad connections:
- PA0 connects COL0 of the keypad
- PA1 connects COL1 of the keypad
- PA2 connects COL2 of the keypad
- PA3 connects COL3 of the keypad
- PA4 connects ROW0 of the keypad
- PA5 connects ROW1 of the keypad
- PA6 connects ROW2 of the keypad
- PA7 connects ROW3 of the keypad

Keypad scan routine sets PA3 low and PA0, PA1,PA2 high, then tests PA4-PA7.
- If no key is down, PA4-PA7 remain high.
- If PA7 = low, the key 15 is down.
- If PA6 = low, the key 14 is down.
- If PA5 = low, the key 13 is down.
- If PA4 = low, the key 12 is down.

Keypad scan routine sets PA2 low and PA0, PA1, PA3 high, then tests PA4-PA7.
- If no key is down, PA4-PA7 remain high.
- If PA7 = low, the key 11 is down.
- If PA6 = low, the key 10 is down.
- If PA5 = low, the key 9 is down.
- If PA4 = low, the key 8 is down.

Keypad scan routine sets PA1 low and PA0, PA2, PA3 high, then tests PA4-PA7.
- If no key is down, PA4-PA7 remain high.
- If PA7 = low, the key 7 is down.
- If PA6 = low, the key 6 is down.
- If PA5 = low, the key 5 is down.
- If PA4 = low, the key 4 is down.

Keypad scan routine sets PA0 low and PA1, PA2, PA3 high, then tests PA4-PA7.
If no key is down, PA4-PA7 remain high.
If PA7 = low, the key 3 is down.
If PA6 = low, the key 2 is down.
If PA5 = low, the key 1 is down.
If PA4 = low, the key 0 is down.

4.5 LCD display

Port K is an 8-bit bi-directional port. It's used for the LCD display module. If the port is not used for the LCD display, it can be used as a general-purpose I/O port.

The pinouts of J11 and J12 are as follows:

Pin 1	GND	
Pin 2	VCC (5V)	
Pin 3	Via a 220 Ohm resistor to GND	
Pin 4	PK0	RS pin for LCD module
Pin 5	PK7	R/W pin for LCD module
Pin 6	PK1	EN pin for LCD module
Pin 7	Not used	
Pin 8	Not used	
Pin 9	Not used	
Pin 10	Mot used	
Pin 11	PK2	DB4 pin for LCD module
Pin 12	PK3	DB5 pin for LCD module
Pin 13	PK4	DB6 pin for LCD module
Pin 14	PK5	DB7 pin for LCD module
Pin 15	Via a 22 Ohm resistor to VCC	LED backlight for LCD module
Pin 16	GND	

Please notice that PK2-PK5 (not PK4-PK7) are used to drive DB4-DB7 of the LCD module.

The LCD module is hardwired for write-only operation. Experienced user can cut a trace between pin 2 and pin 3 of J5 on solder side, then install a 3-pin male header on J5 to make it for both read and write operations. The jumper on the J5 can be used to select the Read/write function of the LCD module . It's write-only if the jumper is placed in the "right" position. It supports both the read and write functions if the jumper is placed in the "left" position.

4.6 Logic probes

Two on-board logic probe LEDs are connected to pin 47 and pin 48 of header H4 and can be used to monitor high or low states of a circuit as logic probes. Pin 47 and Pin 48 of U10 (MC9S12DG256) are not connected to header H4.

4.7 Trimmer pot

The VR2 is connected to the AN07 input of the ADC port via J30, but the trace at J30 can be cut if AN07 must be used by target circuits.

4.8 Dual Digital-to-Analog Converters (DACs)

The on-board 2-ch, 10-bit DAC is installed for learning SPI communication. It convers a digital binary code to an analog signal so a program can generate different waveforms from the DAC.

The DAC installed on the board is LTC1661. Its analog output, OUTA, is provided on the pin between the headers H7 and H8. The other analog output, OUTB, is provided on the pin between the headers H1 and H2. A good application is to connect a DAC output to an ADC input, so a user can send a binary code to the DAC and read the code back from the ADC.

4.9 Speaker

The speaker is a 5V audio transducer and it can be driven by PT5, Output Comparator 3, or PP5 (PWM 5), or the output B of the DAC LTC1661. The jumper on J26 is preset for the PT5 at factory and all sample programs on the CD will drive the speaker via PT5.

During reset, the bootloader or the serial monitor will generate a chirp via the speaker. If the jumper is not placed for the PT5, the chirp won't happen.

4.10 IR transceiver and 38 KHz oscillator

The U7, CD4093, generates a 38KHz square wave for the IR transmitter. One of the CD4093's gate is used as a 38 KHz oscillator. The value of resistor R10 may vary if the CD4093 is manufactured by a different company.

If the IR transmitter is not used by an application program, the 38 KHz square wave also can be available to the user's circuits on the breadboard. If the pin 4 of J27 is short to ground, the 38 KHz square wave will be present at the pin next to the RESET pin on the header H3. You also can use MM command to force PS3 or PT4 to low to enable the 38 KHz oscillator (U7A).

The IR detector can be used as an object detection sensor. When an object is approach the IR detector, it can reflect the 38 KHz signal from IR transmitter to the detector. Normally the output of the detector is in high state. When a 38 KHz IR signal is detected by the IR detector, the output of the detector goes low.

4.11 Dual SCI communication ports

The USB jack P1 connector is used by SCI0 or SCI1of the DG256 while the J43 is used by SC1 of the DG256 in TTL logic level. The D-Bug12 monitor or serial monitor works with SCI0, so the P1 is connected to a PC's USB port during debugging sessions. The SCI1 can be used by user's application programs. The receiver of the SCI1 can receive signals from many different devices, but only from one device at a time, or it will cause a signal collision. The jumper headers J23 and J29 are used to select which device the SCI1 will receive. The J23 selects a signal among USB interface via P1, Async serial communication in TTL logic alevel, RS485 and IR detector. The J29 selects VGA camera. The J23 and J29 cannot have a jumper at the same time. If VGA camera is used, move the jumper from J23 to J29, otherwise move the jumper from J29 to J23.

4.12 RS485 communication port

U5, SN75176, converts the TTL signal from SCI1 to RS485 differential signals and vice versa.

PJ0 (pin 22) of the MC9S12DG256 is used to control the direction of RS485 communication. If PJ0=0, the RS485 port, U9 DS75176, is set as a receiver port. If PJ0=1, the RS485 port, U9 DS75176, is set as a transmitter port.

4.13 External SPI interface

SPI port (J10) pinouts are as follows:

Pin 1	VCC (5V)	Pin 2	VCC (5V)
Pin 3	PM7 (LOAD)	Pin 4	PS4 (SPI DATA IN)
Pin 5	PS7 (STROBE)	Pin 6	PS5 (SPI DATA OUT)
Pin 7	PE1 (/IRQ)	Pin 8	PS6 (CLOCK)
Pin 9	GND	Pin 10	GND

4.14 External I^{2}C interface

I2C port (J2) pinouts are as follows:

Pin 1	VCC (5V)	Pin 2	/IRQ
Pin 3	PM7 (SCL)	Pin 4	PS4 (SDA)
Pin 5	GND		

4.15 RGB LED (common cathode)

The Common Cathode RGB LED comes in two different pin configurations as shown above. In Type 1 the PP4, PP5 and PP6 control Blue, Red and Green LEDs, respectively.

In Type 2 the PP4, PP5 and PP6 control Red, Green and Blue LEDs, respectively.

4.16 All jumper settings

All on-board jumpers:

J1 Enables the LCD backlight.

J2 I^2C interface

J3 Connection of the terminating resistor for CAN0. It's located on the solder side as a solder bridge pad. If you use CAN0, place a solder bridge on this pad. You also can parallel a 120 ohm resistor on T1 (terminal block for CAN0) instead. If CAN0 is not used, it will save power consumption of the board without the terminal resistor installed.

J4 Two channel 10-bit DAC outputs. The output A is also available between PJ6 of H8 and PJ7 of H7. The output B is also available between GND of H1 and PT4 of H2.

J5 R/W of LCD module. It's hardwired for write-only.

J6 PP4 PWM output for Robot servo

J7 PP5 PWM output for Robot servo

J8 PP6 PWM output for Robot servo

J9 PP7 PWM output for Robot servo

J10 SPI connector

J11 On-board LCD connector for 16x2 LCD

J12 External LCD connector for a LCD module of any size.

J13 RS of CAN0 (U2), is connected to VSS

J15 Connects light sensor to AN04 of ADC. It's hardwired.

J16 Connects SQW of the DS1307 to /IRQ. It's not connected.

J17 Connects temperature sensor U14 to PAD05 of ADC. It's hardwired.

J18 VCC for H-bridge driver, U12, SN754410N. The H-bridge driver and 7-segment LED display should not be enabled at the same time. Move the jumper from J24 to this header only if H-bridge driver is used. When H-bridge driver is not used, move this jumper back to J24

J19 Connects the PS3 (TXD1) of SCI1 to all communication hardware (TTL, RS485, and IR transceiver) on this development board.

J20 BDM input

J21 BDM output, when the board is booted in POD mode

J22 RS485 direction control by PJ0 (pin22) through this jumper

J23 SCI1 receiver source select (numbering from top to bottom)
1= SCI1's PS2 receives signal from USB interface chip FT232RL.
2= SCI1's PS2 receives signal from J43 in TTL logic level.
3= SCI1's PS2 receives signal from the terminal block T2 for RS485 port.
4= SCI1's PS2 receives signal from the on-board IR detector. This is the default setting.
Note: J23 and J29 cannot have a jumper at the same time.
If a VGA camera is used, move this jumper to J29.

J24 Enables 7-segment LED display driver U6, 74HC367. 7-segment LED display and H-bridge driver should not be enabled at the same time. If H-bridge driver is used, move this jumper to J18, otherwise leave the jumper on this header.

J25 DC motor power select. The jumper is placed in the "UP" position if motors are powered by the on-board unregulated 9V (VIN). The jumper is placed in the "LOW" position if motors are powered by external 9V at pin 1 of the terminal block T4.

J26 Selects speaker driving source. The speaker can be driven by PT5 (OC3), PP5 (PWM) and DAC B.

J27 IR transceiver control source select
When the jumpers are placed vertically in the “UP” positions (labeled with PS2 and PS3), The PS3 (TXD1) of SCI1 drives the IR transmitter and the PS2 (RXD1) of SCI1 receives data from the IR detector. The PS3 and PS2 can be programmed as general I/O lines or a SCI UART.
When the jumpers are placed vertically in the “LOW” positions (labeled with ‘PT4 and PT3’), the PT4 drives the IR transmitter and the PT3 receives data from the IR detector.

J28 VGA camera interface.

J29 SCI1’s PS2 receives signal from VGA camera. J29 and J23 cannot have a jumper at the same time. If VGA camera is not used, move this jumper to J23.

J30 Connects the trimmer pot VR2 to PAD07.

J31 Connects the trimmer pot VR2 to VRH.

J32 Enables RGB LED.

J34 Connects PE2 to relay circuit.

J35 Servo motor power select. The jumper is placed in the “LEFT” position if servos are powered by the on-board VCC (5V). The jumper is placed in the “RIGHT” position if servos are powered by an external 5V power supply at the terminal block T7.

J36 X-Y-X Accelerometer module interface or IR distance sensor, GP2D12, interface.

J38 Connects the PE3 to Opto-coupler U13

J39 Connects PM0 to RXD of CAN interface U2

J40 TTL logic level of the SCI0. In order to use this header, the jumper on J41must be installed in the “UP” position.

J41 Selects SCI0 interface. The jumper is installed in the “LOW” position for USB interface, or in the “UP” position for TTL interface

J42 Both jumpers are installed in the “UP” positions for connecting SCI0 to USB port. Both jumpers are installed in the “LOW” positions for connecting SCI1 to USB port. When connect SCI1 to USB port, the jumper on J23 must be installed on the “TOP” position labeled with “USB”

J43 TTL logic level of the SCI1 for user application. In order to use this header, both jumpers on the J42 are installed in the “UP” positions and the jumper on the J23 must be installed in the second position from the top labeled with ‘TTL’.

J44 PH2, PH3, PH6 and PH7 connections and can be used for handshaking control for the SCI1.

J46 Port P connections

J47 Port T connections

J48 Port B connections

J49 Selects 2.5V or 5V for VRH.

Chapter 5: EmbeddedGNU

Eric Engler has published the EmbeddedGNU IDE that supports GNU C compiler and assembler for any 68HC11/HC12/HCS12 boards including our FOX11, EVBplus2, DRAGON12 and MiniDragon+ boards. It's free software under Open Source, GNU GPL License. It's not freeware nor shareware (be aware that some freeware are not free). To download Eric's free tools including the GNU C compiler and assembler please visit his web site at: http://www.geocities.com/englere_geo/
For your convenience, we downloaded the egnu094.zip for you.

The following page shows the exact terms of the license (Mozilla Public License)
http://www.geocities.com/englere_geo/License.txt

The steps to set up the EmbeddedGNU are as follows:

1. Download the GNU GCC compiler from: http://m68hc11.serveftp.org/m68hc11_pkg_zip.php
   Select the release 3.1 to download. It has the following components in it:
   Gcc 3.3.6
   Gdb 6.4
   Binutils 2.15
   Newlib 1.12.0

2. Run the file that you downloaded to install GNU 68HC11/68HC12 tools into the default directory of C:\usr.

3. Install the EmbeddedGNU on your PC by double clicking on the egnu094.zip. If the egnu094.zip is not on the CD, you can download it from http://www.ericengler.com/EmbeddedGNU.aspx
   Extract all files into a new directory that you need to create on any hard drive. The name of the new directory can be like c:\egnu094 or d:\egnu094. The EmbeddedGNU.exe and example programs will be located at \egnu094, but your application programs can be located in any other directories.

4. Filename Association.
   When you first start EmbeddedGNU.exe it will ask if you want to associate the filename extensions used by EmbeddedGNU with itself. This lets you double-click on a filename and the EmbeddedGNU will be launched to let you edit the file. The default option is to associate ".prj" with EmbeddedGNU. This is the main project file type used by EmbeddedGNU.

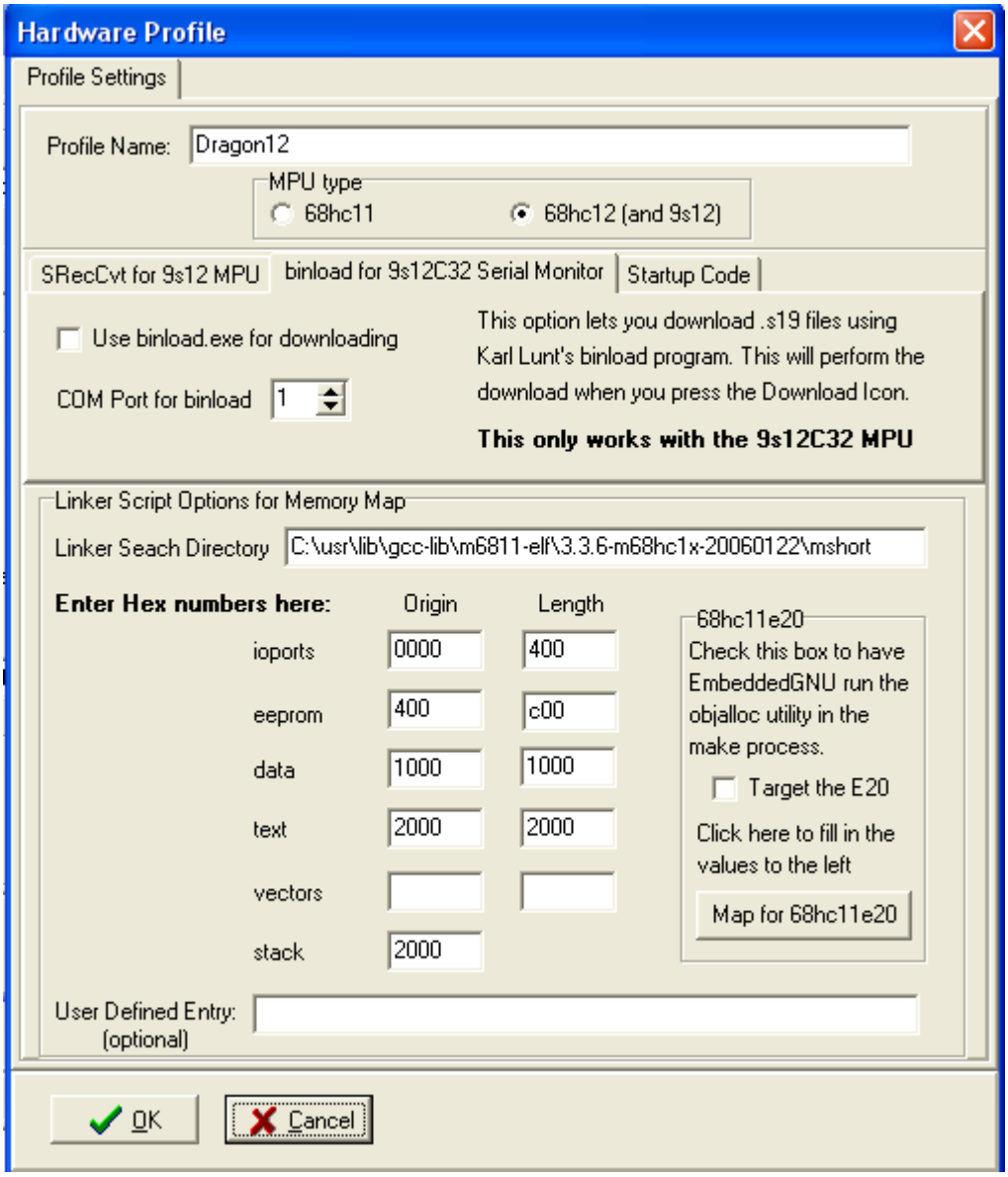
   You also should choose to associate .c, .h, and .s files with EmbeddedGNU.
   WARNING: if you are on WinNT/Win2K/WinXp, then you must be logged in as an administrator to use this option.

   Press OK to continue

5. COM Port Selection.
   It asks if you want to select your COM port. Say Yes. Select your port in the dropdown box. It defaults to 9600 baud, which is normally correct. Now press OK.

6. Select Option-> Environment Options->AutoDownload, then disable ALL automatic commands.

7. The current egnu094.zip is properly set up with the newest release version 3.1 (GCC 3.3.6). In the future when upgrading to a newer version you have to update the linker's search directory. See help file related version upgrade issues.



To change the linker search directory (search path) for GNU C compiler toolset you click on options->project options->edit profile. As it can be seen from above Linker Search Directory, the GCC 3.3.6 is installed on C drive.

Some university web sites offer educational resource for the EmbeddedGNU. The following web site provides A C sample program for the DRAGON12 board using EmbeddedGNU and GCC

York University's CSE4080 computer science project http://www.gcc-hcs12.com/index.php provides easy to use open source and GPL type resources for the HCS12 family.

Chapter 6: Code Warrior and Serial monitor

Code Warrior is a very powerful and professional IDE. The main feature of Code Warrior IDE is the source level debugger in assembler and C. Code Warrior Special Edition is a wonderful gift from Freescale to all of us and it's free for educational use. What's more, by Code Warrior supporting serial monitor, they have made it very affordable to support Code Warrior for the OEM.

Freescale has invested millions of dollar into Code Warrior and the current versions work very well. What's more, Freescale knows they will never sell enough copies of Code Warrior to make back what they have invested. They did it to drive chip sales.

As a software developer, the first thing you look at is available tools and what it will cost. There are many companies making MCU chips these days and for the most part they all have about the same features at a similar price. Special Edition Code Warrior sets Freescale apart from others.

Code Warrior IDE does not work with D-Bug12, but it works with serial monitor. Before Freescale created the serial monitor a BDM is needed as an interface between the PC and HCS12. Freescale created the serial monitor for working with Code Warrior to eliminate the cost of a BDM.

Now a student can use the serial monitor with Code Warrior to debug his program and in fact, many universities have been using the serial monitor with Code Warrior without a BDM in their classrooms.

Without spending money on a BDM, a student will be able to spend his savings on purchasing a more advanced trainer, like the Dragon12-Plus-USB board with many on-board peripherals. Purchasing an EVB board that comes with a BDM at a reasonable price, most likely leaves the student with an EVB of only limited functionality.

Some universities use D-Bug12 monitor first, then replace the D-Bug12 monitor with serial monitor to be used with Code Warrior IDE. In this case, a school laboratory only needs to have one BDM or use one Dragon12 Plus board as a BDM POD, to program all students' boards with serial monitor.

To replace bootloader and D-Bug12 monitor with serial monitor, you need a BDM or a BDM POD to perform the task. The procedure to program the on-chip flash memory is shown on page 16. The file name of the latest serial monitor including a self test program for the Dragon12 board is **SM_and_Test_DR12P_8MHz.s29**, which is available on our web site at:

http://www.evbplus.com/download_hcs12/download_hcs12.html

Some universities use Code Warrior IDE only. In this case, we pre-load the on-chip flash memory with serial monitor.

If your board is pre-loaded with D-Bug12 monitor, the Port B LEDs will light up from **LEFT to RIGHT** one at a time and the speaker will chirp once when the board is turned on. If the chirp is too soft you can remove the sticker on the speaker to increase volume.

If your board is pre-loaded with Serial Monitor the Port B LEDs will light up from **RIGHT to LEFT** one at a time and the speaker will chirp once when the board is turned on. To distinguish RUN mode from LOAD mode, Port B LEDs will light up again from **LEFT to RIGHT** one at a time in RUN mode.

The left DIP switch of SW7 is used to select RUN or downLOAD mode. The left DIP switch is set in the "UP" position for RUN mode and in the "LOW" position for downLOAD mode.

We will add setup procedures for Code Warrior in the future. Our web site provides links to many university web sites and you can visit those web sites for more information.

http://www.evbplus.com/Code_Warrior_hcs12.html

Following is the web site for downloading the free Code Warrior special edition:
http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=01272600610BF1

Following is the web site for downloading the Code Warrior full edition for a 30-day free evaluation:
http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=01272600612247

Chapter 7: PLL code

```
;     The crystal frequency on the Dragon12-Plus-USB board is 8 MHz so the default bus speed
      is
;     4 MHz.  In order to set the bus speed high than 4 MHz the PLL must be initialized.
;
;     You can cut and paste the following code to the beginning of your program.
;
;           The math used to set the PLL frequency is:
;
;           PLLCLK = CrystalFreq * 2 * (initSYNR+1) / (initREFDV+1)
;
;           CrystalFreq =     8 MHz on Dragon12 plus board
;           initSYNR    =     5,  PLL multiplier will be 6
;           initREFDV   =     1,  PLL divisor will be 2
;           PLLCLK = 8*2*6/2 = 48MHz
;           The bus speed = PLLCLK / 2 = 24 MHz
;
;
start:

; PLL code for 24MHz bus speed from a 4/8/16 crystal
            sei
            ldx     #0
            bclr    clksel,x,%10000000      ; clear bit 7, clock derived from oscclk
            bset    pllctl,x,  %01000000    ; Turn PLL on, bit 6 =1 PLL on, bit 6=0 PLL off
            ldaa    #$05                    ; 5+1=6 multiplier
            staa    synr,x
  ;         ldaa    #$03    ; divisor=3+1=4, 16*2*6 /4 =  48MHz PLL freq, for 16 MHz crystal
            ldaa    #$01    ; divisor=1+1=2,   8*2*6 /2 =  48MHz PLL freq, for 8 MHz crystal
  ;         ldaa    #$00    ; divisor=0+1=1,   4*2*6 /1 =  48MHz PLL freq, for 4 MHz crystal

            staa    refdv,x
wait_b3:    brclr   crgflg,x, %00001000 wait_b3     ; Wait until bit 3 = 1
            bset    clksel,x, %10000000
```

8.1 D-Bug12 utility routines

The AN1280 was written for OLD 68HC12 family. If you happen to use printf routine with your old 68HC12 board you should be aware that I/O utility routines are moved to different addresses in D-Bug12 V4.x.

The address for the printf is $EE88 and addresses of other I/O routines are listed below:

Function	Description	Pointer Address
`far main()`	Start of D-Bug12	$EE80
`getchar()`	Get a character from SCI0 or SCI1	$EE84
`putchar()`	Send a character out SCI0 or SCI1	$EE86
`printf()`	Formatted Output - Translates binary values to characters	$EE88
`far GetCmdLine()`	Obtain a line of input from the user	$EE8A
`far sscanhex()`	Convert an ASCII hexadecimal string to a binary integer	$EE8E
`isxdigit()`	Checks for membership in the set [0..9, a..f, A..F]	$EE92
`toupper()`	Converts lower case characters to upper case	$EE94
`isalpha()`	Checks for membership in the set [a..z, A..Z]	$EE96
`strlen()`	Returns the length of a null terminated string	$EE98
`strcpy()`	Copies a null terminated string	$EE9A
`far out2hex()`	Displays 8-bit number as 2 ASCII hex characters	$EE9C
`far out4hex()`	Displays 16-bit number as 4 ASCII hex characters	$EEA0
`SetUserVector()`	Setup user interrupt service routine	$EEA4
`far WriteEEByte()`	Write a data byte to on-chip EEPROM	$EEA6
`far EraseEE()`	Bulk erase on-chip EEPROM	$EEAA
`far ReadMem()`	Read data from the M68HC12 memory map	$EEAE
`far WriteMem()`	Write data to the M68HC12 memory map	$EEB2

Fig 8-1: D-Bug12 utility routines

8.2 Interrupt vector table

Table 5-1 Interrupt Vector Locations

Vector Address	Interrupt Source	CCR Mask	Local Enable	HPRIO Value to Elevate
$FFFE, $FFFF	Reset	None	None	–
$FFFC, $FFFD	Clock Monitor fail reset	None	PLLCTL (CME, SCME)	–
$FFFA, $FFFB	COP failure reset	None	COP rate select	–
$FFF8, $FFF9	Unimplemented instruction trap	None	None	–
$FFF6, $FFF7	SWI	None	None	–
$FFF4, $FFF5	XIRQ	X-Bit	None	–
$FFF2, $FFF3	IRQ	I-Bit	IRQCR (IRQEN)	$F2
$FFF0, $FFF1	Real Time Interrupt	I-Bit	CRGINT (RTIE)	$F0
$FFEE, $FFEF	Enhanced Capture Timer channel 0	I-Bit	TIE (C0I)	$EE
$FFEC, $FFED	Enhanced Capture Timer channel 1	I-Bit	TIE (C1I)	$EC
$FFEA, $FFEB	Enhanced Capture Timer channel 2	I-Bit	TIE (C2I)	$EA
$FFE8, $FFE9	Enhanced Capture Timer channel 3	I-Bit	TIE (C3I)	$E8
$FFE6, $FFE7	Enhanced Capture Timer channel 4	I-Bit	TIE (C4I)	$E6
$FFE4, $FFE5	Enhanced Capture Timer channel 5	I-Bit	TIE (C5I)	$E4
$FFE2, $FFE3	Enhanced Capture Timer channel 6	I-Bit	TIE (C6I)	$E2
$FFE0, $FFE1	Enhanced Capture Timer channel 7	I-Bit	TIE (C7I)	$E0
$FFDE, $FFDF	Enhanced Capture Timer overflow	I-Bit	TSRC2 (TOF)	$DE
$FFDC, $FFDD	Pulse accumulator A overflow	I-Bit	PACTL (PAOVI)	$DC
$FFDA, $FFDB	Pulse accumulator input edge	I-Bit	PACTL (PAI)	$DA
$FFD8, $FFD9	SPI0	I-Bit	SP0CR1 (SPIE, SPTIE)	$D8
$FFD6, $FFD7	SCI0	I-Bit	SC0CR2 (TIE, TCIE, RIE, ILIE)	$D6
$FFD4, $FFD5	SCI1	I-Bit	SC1CR2 (TIE, TCIE, RIE, ILIE)	$D4
$FFD2, $FFD3	ATD0	I-Bit	ATD0CTL2 (ASCIE)	$D2
$FFD0, $FFD1	ATD1	I-Bit	ATD1CTL2 (ASCIE)	$D0
$FFCE, $FFCF	Port J	I-Bit	PTJIF (PTJIE)	$CE
$FFCC, $FFCD	Port H	I-Bit	PTHIF(PTHIE)	$CC
$FFCA, $FFCB	Modulus Down Counter underflow	I-Bit	MCCTL(MCZI)	$CA

Fig 8-2: MC9S12DG256 Interrupt vector table 1

$FFC8, $FFC9	Pulse Accumulator B Overflow	I-Bit	PBCTL(PBOVI)	$C8
$FFC6, $FFC7	CRG PLL lock	I-Bit	CRGINT(LOCKIE)	$C6
$FFC4, $FFC5	CRG Self Clock Mode	I-Bit	CRGINT (SCMIE)	$C4
$FFC2, $FFC3	BDLC	I-Bit	DLCBCR1(IE)	$C2
$FFC0, $FFC1	IIC Bus	I-Bit	IBCR (IBIE)	$C0
$FFBE, $FFBF	SPI1	I-Bit	SP1CR1 (SPIE, SPTIE)	$BE
$FFBC, $FFBD	SPI2	I-Bit	SP2CR1 (SPIE, SPTIE)	$BC
$FFBA, $FFBB	EEPROM	I-Bit	EECTL(CCIE, CBEIE)	$BA
$FFB8, $FFB9	FLASH	I-Bit	FCTL(CCIE, CBEIE)	$B8
$FFB6, $FFB7	CAN0 wake-up	I-Bit	CAN0RIER (WUPIE)	$B6
$FFB4, $FFB5	CAN0 errors	I-Bit	CAN0RIER (CSCIE, OVRIE)	$B4
$FFB2, $FFB3	CAN0 receive	I-Bit	CAN0RIER (RXFIE)	$B2
$FFB0, $FFB1	CAN0 transmit	I-Bit	CAN0TIER (TXEIE2-TXEIE0)	$B0
$FFAE, $FFAF	CAN1 wake-up	I-Bit	CAN1RIER (WUPIE)	$AE
$FFAC, $FFAD	CAN1 errors	I-Bit	CAN1RIER (CSCIE, OVRIE)	$AC
$FFAA, $FFAB	CAN1 receive	I-Bit	CAN1RIER (RXFIE)	$AA
$FFA8, $FFA9	CAN1 transmit	I-Bit	CAN1TIER (TXEIE2-TXEIE0)	$A8
$FFA6, $FFA7	CAN2 wake-up	I-Bit	CAN2RIER (WUPIE)	$A6
$FFA4, $FFA5	CAN2 errors	I-Bit	CAN2RIER (CSCIE, OVRIE)	$A4
$FFA2, $FFA3	CAN2 receive	I-Bit	CAN2RIER (RXFIE)	$A2
$FFA0, $FFA1	CAN2 transmit	I-Bit	CAN2TIER (TXEIE2-TXEIE0)	$A0
$FF9E, $FF9F	CAN3 wake-up	I-Bit	CAN3RIER (WUPIE)	$9E
$FF9C, $FF9D	CAN3 errors	I-Bit	CAN3RIER (TXEIE2-TXEIE0)	$9C
$FF9A, $FF9B	CAN3 receive	I-Bit	CAN3RIER (RXFIE)	$9A
$FF98, $FF99	CAN3 transmit	I-Bit	CAN3TIER (TXEIE2-TXEIE0)	$98
$FF96, $FF97	CAN4 wake-up	I-Bit	CAN4RIER (WUPIE)	$96
$FF94, $FF95	CAN4 errors	I-Bit	CAN4RIER (CSCIE, OVRIE)	$94
$FF92, $FF93	CAN4 receive	I-Bit	CAN4RIER (RXFIE)	$92
$FF90, $FF91	CAN4 transmit	I-Bit	CAN4TIER (TXEIE2-TXEIE0)	$90
$FF8E, $FF8F	Port P Interrupt	I-Bit	PTPIF (PTPIE)	$8E
$FF8C, $FF8D	PWM Emergency Shutdown	I-Bit	PWMSDN (PWMIE)	$8C
$FF80 to $FF8B	Reserved			

Fig 8-3: MC9S12DG256 Interrupt vector table 2

Interrupt Source	Secondary Vector Address	Interrupt Source	Secondary Vector Address
Reserved $FF80	$EF80	I^2C bus	$EFC0
Reserved $FF82	$EF82	DLC	$EFC2
Reserved $FF84	$EF84	SCME	$EFC4
Reserved $FF86	$EF86	CRG lock	$EFC6
Reserved $FF88	$EF88	Pulse accumulator B overflow	$EFC8
Reserved $FF8A	$EF8A	Modulus down counter underflow	$EFCA
PWM emergency shutdown	$EF8C	Port H interrupt	$EFCC
Port P interrupt	$EF8E	Port J interrupt	$EFCE
MSCAN 4 transmit	$EF90	ATD1	$EFD0
MSCAN 4 receive	$EF92	ATD0	$EFD2
MSCAN 4 errors	$EF94	SCI1	$EFD4
MSCAN 4 wakeup	$EF96	SCI0	$EFD6
MSCAN 3 transmit	$EF98	SPI0	$EFD8
MSCAN 3 receive	$EF9A	Pulse accumulator A input edge	$EFDA
MSCAN 3 errors	$EF9C	Pulse accumulator A overflow	$EFDC
MSCAN 3 wakeup	$EF9E	Timer overflow	$EFDE
MSCAN 2 transmit	$EFA0	Timer channel 7	$EFE0
MSCAN 2 receive	$EFA2	Timer channel 6	$EFE2
MSCAN 2 errors	$EFA4	Timer channel 5	$EFE4
MSCAN 2 wakeup	$EFA6	Timer channel 4	$EFE6
MSCAN 1 transmit	$EFA8	Timer channel 3	$EFE8
MSCAN 1 receive	$EFAA	Timer channel 2	$EFEA
MSCAN 1 errors	$EFAC	Timer channel 1	$EFEC
MSCAN 1 wakeup	$EFAE	Timer channel 0	$EFEE
MSCAN 0 transmit	$EFB0	Real-time interrupt	$EFF0
MSCAN 0 receive	$EFB2	IRQ	$EFF2
MSCAN 0 errors	$EFB4	XIRQ	$EFF4
MSCAN 0 wakeup	$EFB6	SWI	$EFF6
FLASH	$EFB8	Unimplemented instruction trap	$EFF8
EEPROM	$EFBA	COP failure reset	$EFFA
SPI2	$EFBC	Clock monitor fail reset	$EFFC
SPI1	$EFBE	Reset	$EFFE

Fig 8-4: MC9S12DG256 secondary interrupt vector table

8.3 Useful web links

The web is the best source for getting more information about the HCS12. The Freescale web site has all documents and application notes that you need.
The HC12 user group http://groups.yahoo.com/group/68HC12/ and Freescale's forums http://forums.freescale.com/freescale/ are good places to ask a question and get a prompt answer from many other HC12 users.

You also can visit our web site at:
http://www.evbplus.com/hc11_68hc11_hc12_68hc12_9s12_hcs12_sites.html

to get links to many university web sites that offer course materials and lab assignments for the Dragon12 and Dragon12-Plus-USB boards.

All HCS12 boards that are pre-loaded with Freescale serial monitor, bootloader and D-Bug12 monitor on the market today are basically the same products as far as software development is concerned. If you are going to use a BDM to debug a HCS12 board, all HCS12 boards will respond to all BDM commands in the same manner because the BDM directly communicates with the MC9S12DG256 MCU. The information on our manual can apply to the boards from other manufacturers, and vice versa.

8.4 Troubleshooting notes

The following are some important notes that you should know and they may save you time:

1. Things to do if the board does not work.

Many little mistakes can cause a big problem, especially for beginners. For instance, if you want to run the board in single chip mode, but MODEB, A, and C are set for expanded mode, you know it won't work. If the jumper on J1 is missing, the LCD backlight won't work and if the jumper on J24 is missing, the 7-segment display won't be lit.

Before troubleshooting the board, you must apply power to the board. When the board is powered, the PWR LED indicator must be on. If it's off, the board does not have 5V DC. Sometimes it may be caused by a bad AC adapter or the AC adapter may not even be plugged in.

To determine if the board malfunctions, you can restore the following jumper settings to the original default settings when you receive the board. The default settings are as follows:

J1	Enables the LCD backlight, jumper is installed.
J18	VCC for H-bridge, U12, SN754410N, no jumper is installed.
J23	SCI1 receiver select, jumper is set for IR (in the "BOTTOM" position)
J24	7-segment_EN, jumper is installed
J25	DC motor power select. Jumper is placed in the "UP" position.
J26	Speaker driving source. Jumper is placed in the "TOP" position (driven by PT5)
J27	IR select, both jumpers are placed vertically in the "UP" positions
J29	SCI1 receiver select, no jumper installed
J32	Enables RGB LED, jumper is installed.
J34	Connects PE2 to relay circuit, jumper is installed.
J35	Servo motor power select. The jumper is placed in the "LEFT" position.
J41	Selects SCI0 interface. The jumper is installed in the "LOW" position for USB interface.

J42 Both jumpers are installed vertically in the “UP” positions for connecting SCI0 to USB port.

J49 Selects 2.5V or 5V for VRH, jumper is installed in the “UP” position for 5V.

SW7 MODE select, both DIP switches of SW7 are installed in the “LOW” positions for EVB mode.

If all above settings are correct and you press the reset button, the PB0-PB7 LEDs should light up from left to right one at a time and the LCD should display the following message:

“DRAGON12plus EVB”
“D-Bug12 EVB MODE”

If the LEDs lighted up and the LCD displays the following message:

“DRAGON12plus EVB”
“ D-Bug12 ERASED ” or “RUN USER PROGRAM” or “********”**

then the D-Bug12 monitor is erased. You can re-program the D-Bug12 in bootloader mode according the instructions on page 19. If the board does not communicate with the PC, the COM port number may not be set correctly by AsmIDE. If the screen displays some garbled characters, the baud rate may not be set correctly. The D-Bug12 resets the baud rate to 9600 during power up, if you changed the baud, you must change the AsmIDE’s baud rate to the same baud.

If the PB0-PB7 LEDs don’t light up from left to right one at a time, the bootloader could be erased by a BDM. You can use a BDM with instructions from the manufacturer or use another Dragon12 Plus board as a BDM POD to re-program bootloader and D-Bug12 monitor into flash memory according to the instructions on page 16.

The newest firmware can be downloaded at: www.evbplus.com/download_hcs12.html

2. Always reset the board before downloading a new program.

If the previous application program that you ran was aborted, then you may need to reset the board before downloading a new application program. The reset action will disable the interrupt that was enabled by the previous application. If the interrupt was caused by a timer and is not disabled, the timer interrupt will continue even it’s not called for in your new application program. The result will be unpredictable.

3. In EVB mode, reset clears your pseudo RAM interrupt vectors.

When you develop code with interrupts in RAM, you must initialize pseudo RAM interrupt vectors in the very beginning of your program, because if you press the reset button it will clear all pseudo RAM interrupt vectors. If you don’t initialize pseudo RAM interrupt vectors in your program and your application program uses interrupts, your program may not run correctly since the interrupt vectors do not exist.

4. Operating mode changing is only effective after reset.

There are four operating modes that are selected by SW7. The mode change won’t be effective until you reset the board. So you must always press the reset button after a mode change.

8.5 Revision History

Date	Rev. #	Notes
04/18/07	A	Prototype
06/30/07	B	First batch of production boards. 1. The external resistor network (RN12) for port A on the original Dragon12 board is eliminated on the new Dragon12-Plus-USB board. In order for keypad routine to work with the new Dragon12-Plus-USB board, the internal pull-up resistors for port A must be enabled by adding the following instruction at the beginning of your program source code: bset pucr,1 ; enable pull-up resistors on port A 2. The PB0-PB7 LEDs on the original Dragon12 board prior to rev. E board are driven by port B only, but they are also controlled by PJ1 on the new Dragon12-Plus-USB board. In order to turn on PB0-PB7 LEDs on the new Dragon12-Plus-USB board, the PJ1 (pin 21 of MC9S12DG256) must be programmed as output and set for logic zero.
09/05/07	C	1. Eliminated low battery detection circuit for easy manufacturing. 2. Eliminated VR1 footprint for easy manufacturing. 3. Eliminated U4, U17 footprints for preventing solder bridges. 4. Eliminated J24B (LED_EN) and rename J24A (7SEG_EN) to J24. 5. The headers J20(BDM-IN) and J21(BDM-OUT) are rotated by 90 degrees for easy manufacturing. 6. Added footprint of a LPF (R12 and C26) for RF receiver. 7. C37,C38,C39,C40,C41,C42,C43,C44,C45,J28 and J29 are not installed since they are not used by standard features. 8. RN11 is not installed and it's replaced by internal pull-up resistors.
09/26/07	D	Same as rev C board except: 1. LCD module is hardwired for write-only operation. Experienced user can cut a trace between pin 2 and pin 3 of J5 on solder side, then install a 3-pin male header on J5 to make it for both read and write operations. 2. Locations of J1 and J5 are swapped for easy manufacturing. 3. Corrected the misprinted label PT6 to PT5 for J26.
06/01/08	E	Designed for a special customer and was never sold to public.
10/06/08	F	Same as rev D board except: 1. Most discrete components, such as resistors, capacitors, diodes, transistors and LEDs are changed to SMD. 2. The temperature sensor is changed from MCP9701A (U14) to LM45 (U14A). Both are the same type of sensors with analog voltage outputs, but with different sensitivities. The MCP9701A outputs 19.5mV per degree C while the LM45 outputs 10mV per degree C.

04/18/10	G	Same as rev F board except: 1. Added on-board USB interface 2. Added RGB LED 3. 5x2 male headers for port P, port T and port B 4. 2.5V or 5V jumper selectable for VRH 5. 2 logic probes 6. P91 (PS2) of the U10 (DG256) is directly connected to header H7. 7. Switching power supply AC adapter

DETERMINING PROPER WIRE SIZE

The two (2) most important factors to be considered are how much current the wire has to carry and how long the wire run is. The most important factor of these is current. The more current drawn the larger the wire you will need. Wire size (diameter) is specified in terms of "AWG" which is more commonly known as "Gauge". The confusing thing about this is that as the wire gets larger, its' gauge number gets smaller. For our models, we generally use wire in the 12 to 26 Gauge range.

Before you can select the proper size wire, you will need to know how much current the wire will carry. Ideally, you know or can measure the current use of each item in your model, then it is a simple matter to add these numbers up to get the total amount. In reality, it is not that complicated, most items, (lights, sound effects, smoke units and motors), tell you the current draw on their packaging. For those that give a range like drive motors, select a value that is 50% to 75% of the max depending on the size of the prop. Once all of these individual amounts are known, you can add them up to determine the total. From the chart provided at the end of this article, you can now select the size wire required or needed. To be on the safe side, select a size that will carry your total load plus an additional 25%, this will be the size wire needed to run from your battery to your power distribution point. From the power distribution point to each individual device, use the proper size wire to carry the current draw of that specific device.

I also mentioned that the length of the wire is important. This is because of the resistance of the wire. This resistance causes voltage drops. The longer the wire, the more voltage lost. To counter this, a lager wire (smaller gauge), is used as the larger the wire the less the resistance. For any wire run over five (5) feet, I would consider using the next size larger wire than what the current draw requires. If in doubt, it is better to use too large a wire as opposed to too small a wire. The only disadvantage to the larger wire is that it is stiffer and therefore harder to work with.

This Chart will help you determine what Wire Gauge you will need:

WIRE GAUGE	CURRENT CAPACITY
12 Gauge	41 Amps
14 Gauge	32 Amps
16 Gauge	22 Amps
18 Gauge	16 Amps
20 Gauge	11 Amps
22 Gauge	7 Amps
24 Gauge	3.5 Amps
26 Gauge	2.2 Amps
28 Gauge	1.4 Amps

Excerpt from Loyalhanna Dockyard Newsletter March 2005