**Goals**

Headnode is one of two different custom PCBs in solar car that are a part of our Battery Protection System (BPS). The first board, Moduleboard, monitors the status of one battery module, which includes 34 18650GA Lithium-Ion batteries in parallel. Our entire battery pack includes 35 of these battery modules in series to reach a nominal voltage of 140 V, and in turn 35 Moduleboards, each going on the top of a battery module. The second board, Headnode, controls the battery pack relays by checking if either the hardware or software faults on either board are asserted. Both the hardware checks made by comparators and the software checks made against a voltage reference need to not be triggered on a Moduleboard. The hardware fault is ANDed together with the logic check from the next board, so when one module faults in the middle of the board, the proceeding Moduleboard will also fault in a ripple carry. The initial 5V "Working" signal for the fault is initially supplied by Headnode, and sent to Moduleboard #1. At the last Moduleboard, #35, this ANDed logic signal will be sent back in to Headnode as an input. The software checks are made over an isolated CAN network that is connected only to the boards inside the battery pack.

In General, the purpose of BPS is to monitor these faults and control the battery pack relays that open/close to control battery consumption. Both BPS and Powerboard, a board used to regulate the battery pack voltage down to 12 V, are powered off of a 25 V supplemental battery used to check the state of the batteries before drawing current from them. This is needed as a safety precaution and because of regulations we have to follow to build the car. With this supplemental battery, BPS will check to see if any hardware and software faults trigger in the car, and if they don't, BPS will close the positive, negative, and charge relays in the battery pack. After this is done, Headnode will send a buffered 12V logic signal to powerboard indicating that it is now okay to draw current from the battery pack through the closed relays. Powerboard will simultaneously switch off of drawing power from the supplemental battery pack to our main battery pack. After this, the fourth and final board in the battery pack, Precharge, is powered and can start to slowly increase the voltage given to the motors through a power resistor. Since we use DC series motors, if we instantly give it 140 V through a closed relay, we would blow them up.

Generally, this lays out how our newly revised battery pack architecture works. Before this revision of Headnode, Precharge was used to drive 12 V through the positive and negative relays that connect to the battery pack terminals. So, the primary goal of this rev was to maintain the same requirements for turning on the car and moving relay logic from Precharge to Headnode.

**Design**

Since in the goals tab I talked about the top level overview of the architecture for our battery pack, I would like to talk about the different changes to the schematic and PCB layout that I helped accomplish for this rev of Headnode here. On this revision, since there were 8 different schematic pages, we used input and output ports to manage the different signals and power lines. So, we had a master page that showed how the interconnections worked between the pages. This was very helpful since Altium would highlight when a specific port was either not connected or labeled as an input or output wrong.

On the connector page, we listed the 12V power circuit, which at this time was not the same as Buttonboard or HAL. We did not update this circuit for this change, but it should be noted that I did update the power circuit to be the same in the following revision. This page also has ports tying to all of the connectors on the board, including logic signals for faults or enable signals for different boards. This page also included the 12 V control lines for the three different battery pack relays, positive, negative, and charge that are driven by Headnode as of this revision. There are also two CAN connectors for our CAN bus and a connector used for logic to turn on the solar cell array and external charger. The charging connector is a redundancy and we noted to remove many of these logic signals in the next rev since they were not used and would lead to confusion during system implementation.

On the next schematic page, the interface logic, we handled multiple AND gates, an inverter, and MOSFET buffers to manage different logic signals going to Powerboard and Precharge, and coming in from Moduleboard. The logic signal START\_EN was used to monitor any fault conditions that could happen externally from Headnode. Three logic signals were AND'ed together, including external kill, a software fault from a Moduleboard, and a hardware fault from a Moduleboard. External kill is a switch that we are required to have by regulation on the outside of the car that will open the relays connected to the battery pack. Each individual relay now also is AND'ed with a relay control signal from the microcontroller on Headnode and the given START\_EN logic. The other AND gates are used to create an XOR gate to enable externally charging or our solar array. As mentioned before, many of these logic signals related to charging went unused, so much of this got removed later on. The powerboard buffer circuit is very similar to what is used now on HAL, as mentioned before. Generally, a PMOS and an NMOS are used to buffer a 5V enable signal from a microcontroller to create a 12V source that is outputted from Headnode to Powerboard that is not being drawn directly from the microcontroller. Precharge used a 5V enable pin and a BJT, which was later changed due to our team not using many BJT's, which affected part availability.

The Moduleboard interface schematic is pretty empty, but it also makes it very easy to follow. On the top portion, a NPN transistor is used for the Moduleboard hardware fault return line, which will buffer a 5V signal to determine if there is a hardware fault on any module. If the fault return line is 5V, or logic "high", then there is no fault, but if it is 0V, or logic level "low", then we have faulted. We decided to do it this way so that it would be easier to AND this signal for START\_EN on the previous page. The external kill line is connected to a latching button on the outside of the car, with the other end being ground. There is a pull up resistor to 5V on the external switch line on Headnode, so that when the button is not latched, we will not fault the car and disconnect the batteries. When the button is pressed, current will flow through ground since it is the path of least resistance, and the car will fault since this logic is used for START\_EN. There is also a 0-ohm jumper resistor that is used connected between 5V and the analog fault output that starts the hardware checks at Moduleboard #1. this jumper was included for testing and is always connected, which was later removed due to redundancies in a later revision.

The compute page is used to show input and output port connections to the microcontroller. Some notable inputs include the external charge switch and hardware Moduleboard fault logic for monitoring. Some essential outputs include the relay enable logic that is AND'ed with the START\_EN logic to close the relays, and the logic that is buffered to send a signal to the other battery pack boards Powerboard and Precharge.

The regulators page is used to generate a 5V power line and a 2.5V reference. The 5V regulator is very important as it powers our microcontroller and is used for multiple logic gates on the board. The 2.5V reference is a redundancy from our previous architecture and will be removed on later revisions.

The core of the board is within the next two schematic pages, which are used to power the relays that will supply a 12V output from Headnode to the battery pack relays. So, in general, we use relays to power more relays. Like I mentioned before, 2 input AND gates are used with START\_EN and a software logic signal to turn on each individual relay. The positive, negative, and charge relays are all separated by 200 milliseconds of turn on in the car, which leads to a very satisfying sound when we turn the car fully on. We included 0-ohm resistors so that we could choose between using the relays to supply 12V or instead a power switch part that we were testing with this revision. Due to the semiconductor shortage, we were unable to test or implement the power switches, but those will make a return in revision 2 of Headnode. The relay page uses a similar NMOS/PMOS buffer with a 5V logic signal to suppler 12V to the relays on Headnode. We also used flyback diodes across the inductors of the relays to ensure no relay would accidentally close when undesired, which would accidentally turn the battery pack on!

Since we added so many parts to this version of Headnode, we decided to completely redo the PCB layout. In general, the left side of the board is used for the MOSFET buffers and relays that are used to supply the 12 V output to drive the battery pack relays. The right side of the board is used to manage logic gates relating to charging, and it also includes the input for the external kill switch. The top of the board includes connectors to power battery pack fans, current sensors for our pack, and the logic signal that is sent to Precharge. On the bottom of the board, we included the buffer circuit that creates a 12 V logic signal for Powerboard. The microcontroller is included in the middle of the board so that it can be easily reached between every circuit on the respective sides of the board.

We decided to make the new version of the board 4 layers, so that it would be easier to manage the 12V, 5V, and ground layers. On the top and bottom layers, we made the left side of the board 12V and the right side of the board ground. We then used stitching vias to connect the top and bottom layers so that we would not have to worry about islands on our power planes. This board is also where I was taught how to manage signals on different layers. In general, we tried to keep vertical traces on the bottom plane in blue, and horizontal signals on the top plane in red.

**What I Learned**

Overall, I would say this project is where I took a deep dive in to solar car. After learning how to do the general PCB design process with the team on Buttonboard, I moved to designing Headnode during the Spring of 2021. During this time, I got to learn more about designing circuits and layout for logic gates and diodes, while also learning more about the upcoming changes to our battery pack architecture. This opened tons of great opportunities for me, since I was then able to actively participate in more debugging sessions before our Summer 2021 FSGP race.