Logic Board

User and Maintenance Manual

ECE 492 - Spring 2021

Introduction

The Logic board houses 2 computers of the car, a Raspberry Pi 3B for SCADA and a pic32 for the drive states of the car. It handles I2C and CAN communication, houses the IMU and RTC, and electrically isolates these computers from GLV to prevent them from blowing out. It fits on top of the pi as a shield, and is mounted on the top shelf of the enclosure. A ribbon cable connects the board to the TSI board. Questions can be directed to Zachary Martin, at martinz@lafayette.edu.

Major Logic Components

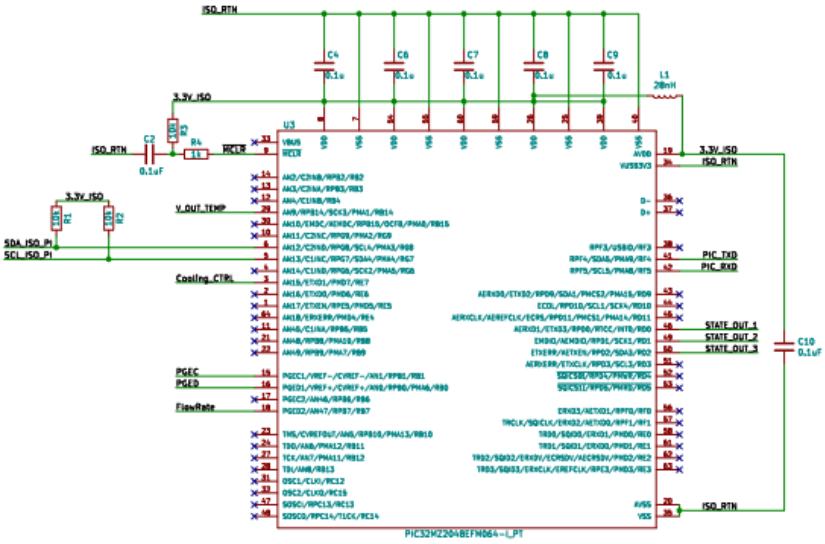
1. Pic32 and Drive States
2. Pi 3B and SCADA
3. Isolation
4. IMU and RTC

Pic32 and Drive States

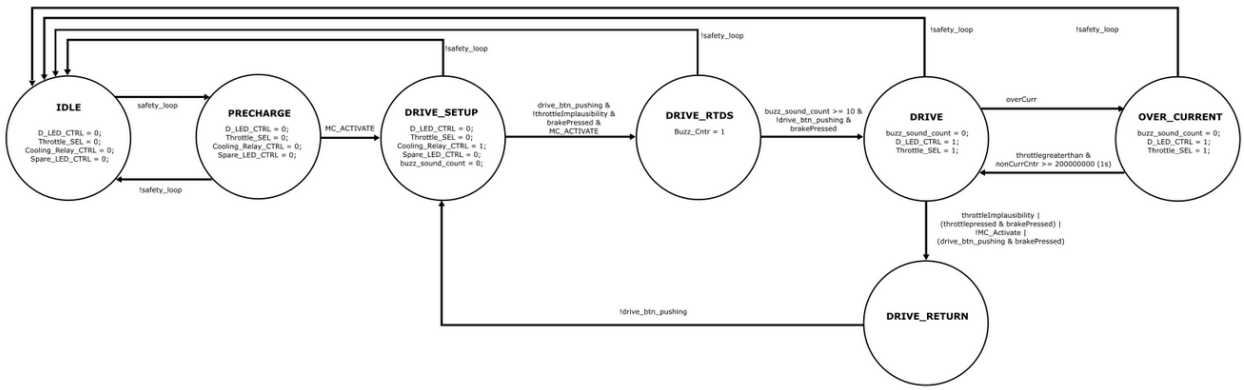
The Pic32 on the car controls the drive states, or when it is acceptable for the user to drive the car. Communicating to a DAC on TSI via I2C, the Pic32 controls a throttle select signal that allows for the motor controller's throttle voltage input to be sent a stepped-down APPS2_b signal when the throttle is plausible (please see TSI's Maintenance Manual for more on throttle plausibility). It is only possible to drive the motor (asserting throttle_sel) when the car is in drive mode. Once the car has its AIRs closed and the motor has been fully charged, pressing the drive button and pressing the brake at the same time puts the car into drive_RTDS mode. You will know when you are here when the RTDS (ready-to-drive sound) buzzer activates for 2 seconds. Letting go of the drive button (but keeping your foot on the brake) puts you into drive mode. Pressing the brake and drive button again at the same time will allow you to exit drive mode.

In addition to drive states, the Pic32 also controls the cooling system. It accepts temperature and flow rate measurements, and if it deems that the car is too hot, it will activate Cooling's 24V-12V converter on TSI to activate the fan and pump. This cooling control feature has not been implemented into the current firmware, and neither has the I2C control signal for the throttle select. My advice to the team reading this next semester: do not let 1 student lie and promise that

they will work on TSI's firmware, and then never work on it for almost 2 semesters. I am most likely still bitter about this fact. Don't make your teammates bitter, just be honest about your work.



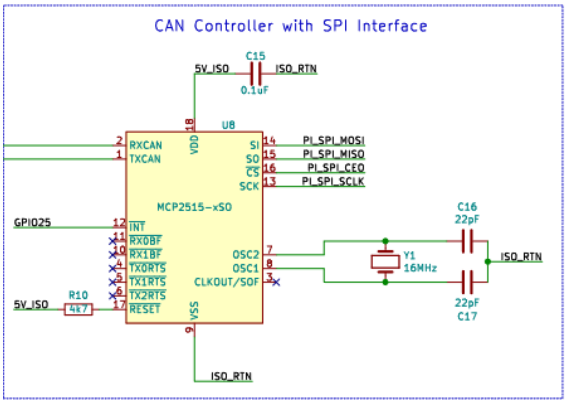
Note: The capacitors attached to those IO pins of the Pic32 act like a cool glass of current for those pins to slurp up. Shown below is the FSM diagram of the drive states.



Pi 3B and SCADA

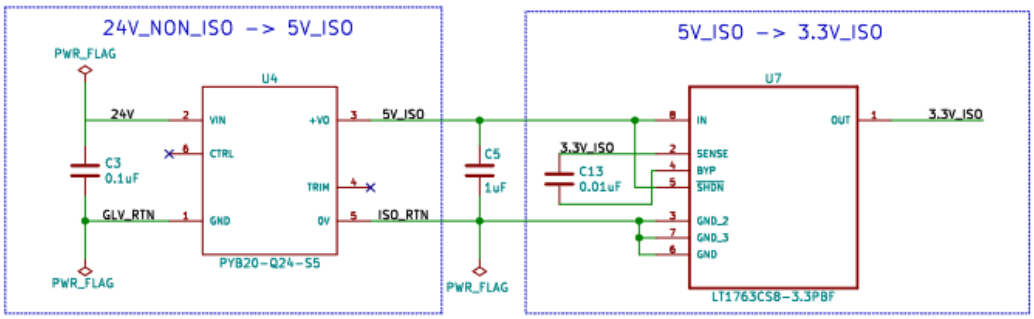
The Raspberry Pi on the board gets directly powered by Logic's isolating converter (more on that below). Like the Pic32, it gets I2C and CAN lines. Unlike the Pic32, the Pi cannot directly receive CAN communication and so it first goes into a CAN controller that converts it to SPI for the Pi. Header J6 has a 120 Ohm terminating resistor for the CAN line if you need it. SCADA

uses these communication lines to read and display sensor data. Please see SCADA documentation for more information, and our ATP (acceptance test plan for the dyno room) for a list of sensors on the car. The Pi also controls the SR_CTRL signal, which is routed to TSI to open the SCADA relay.

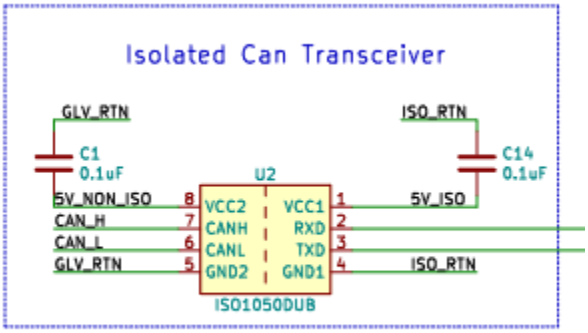


Isolation

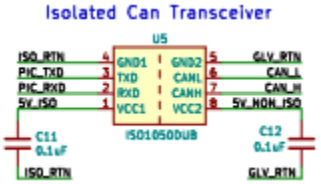
Power, I2C, and CAN all get isolated on logic. The decision to electrically isolate the Pic32 and Pi was made to stop the Pi from repeatedly boot-looping (an issue last year's team had) and to protect these devices from an inrush of current. Isolated 24V and 5V are used to power the devices on board. Since these devices are isolated, all communication between Logic and any other board must be isolated as well.



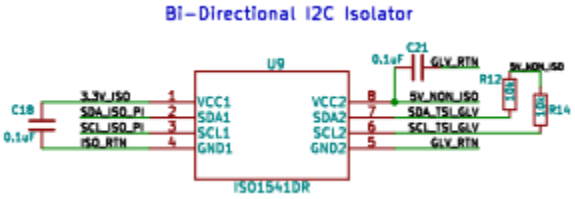
For the Pi's CAN-SPI isolation:



For the Pic32's CAN line:



I2C for both:



To test if you can detect sensors on the I2C lines, run the command "`i2c detect -y 3`" on the Pi's terminal to see the detected addresses. Depending on how certain pins are configured on the sensors, they will have different addresses. Their datasheets list what they are.

IMU and RTC

The correct IMU being used is the STEMMA QT BNO055 IMU. It plugs directly into the board, with an extra pin being left out of the header. The correct RTC being used is the RTC for Raspberry Pi (https://www.adafruit.com/product/3386#description). Don't trust SCADA's BOM for these two devices.