**Current Thoughts:**

Goal: Get the two boards to both send and receive real data. Use fuel cell temperature and voltage for one node and fuel cell ambient temperature and brake indicator for the other. It should

* Be transmitting the logging data twice a second for each data
* Periodically (once per 5 seconds) send the brake message.

Once we are seeing sending and receiving for both nodes, try:

* Having one node spamming the bus and the other node periodically sending data
* If possible, have one of the high priority data packets trigger a shutdown/slowdown of the spamming node so that we know it got through.

If all else fails, solve the filtering problem in user code.

**Regardless**

1. Create a new library that will be specific to the device being programmed. For example:
   1. Have Filter0\_ID through Filter5\_ID that can be set to different IDs in cand\_message\_defs
   2. Have different settings routed through there like how data logger won’t filter at all ‘11’.
   3. Consider having 4 lsb’s of id reserved for data type
2. Create a program that can be sent three different messages and
   1. Only accept 1 or 2
   2. Tell what data type is being sent
3. Get two nodes transmitting and receiving.
4. Try to set up a can network with 3 or more nodes

**White Paper (Summary of workings of CAN bus)**

Messages sent over CAN have two broad categories, data logging and real time.

**Data logging:** Data does not need to be processed in real time, so less consideration is needed in how it is processed. By logging the ID(2 bytes), length(1 byte) and data (integer of up to 8 bytes) the data can be later constructed. Data logging is also low priority and should have a lower priority than all real time messages, this is done by setting first two bits to ones for all message IDs.

* Example:
  + ID: 11 10101 1010 (11 bit, space added for clarity).
    - Bit 10 indicates a data logging message (1) or a RT message (0).
    - Bits 9-4 indicate the sending node (64 combinations)
    - Bits 3-0 indicate the data type, in this case 8 bytes consisting of two 4 byte integer readings of temperature.
    - All together it indicates data to log (00) about the fuel cell (10101) that is two temperature values.
    - Consider the case of fuel cell where there could be multiple fuel cell temperature readings, how to differentiate? (use data type bits or node bits? If using node bits, consider reducing first section to only 1 bit indicating data logging with a 1 and real time messages with a 0.

# Assembling CAN Messages:

* Use can\_message\_defs to write values to message structure
* Message structure contains:
  + **ID** (11 bits) – lowest ID takes priority in a collision / in the case of pending messages
  + **RTransR** (1 bit) – 1 if the message is a remote transit request
  + **Length** **of** **Data** (4 bits) – Number from 0 - 8
  + **Data** **Field** – (0 – 8 bytes)

# Message Filtering:

* Summary of MCP2515 filtermasking. See the datasheet for more details
  + MCP2515 has the ability to implement 5 filters. These are simple equivalence filters. Incoming IDs are compared to the 11 bit filters. IDs which are exact matches are loaded into the rx buffers.
  + Two filters RXF0 and RXF1 load into the rx0 buffer and RX2-5 load into the rx1 buffer.
  + Masks are used to set which bits to compare. To ignore certain ID bits, leave those bits in the mask at 0. Set all bits to consider to 1.
  + Note that once the controller is set to use filtering, each buffer should have at least one filter mask. This is because the default value of the mask for each buffer is zero leading to automatic acceptance of each message.
* Filtering will be done by assigning a unique ID for each message.
* The least significant 4 bits of the IDs are reserved for data type. This is to take advantage of the fact that the car does not require 2047 different IDs but would suffer from using valuable data slots for metadata. This leaves 127 unique IDs from the most significant 7 bits.
* In short, the first 7 bits will indicate where the message came from and the least significant 4 bits will tell what type of data.
* Data types:
* Data Logging is done in regularly scheduled intervals and can be put off temporarily
* Indicators are time sensitive and need to be sent in relatively short times
* Message categories:
  + Low Priority Data Logging (512 – 1023)
  + High Priority Data Logging (256 – 511)
  + Low Priority Indicators (192 – 255)
  + Medium Priority Indicators (128 – 191)
  + Critical Indicators (16 – 127)
  + Reserved (1 – 15)

# Data Packaging:

* Summary of MCP2515 filtermasking. See the datasheet for more details

MCP2515