To use the oneCAN library, you need to download and import both the oneCAN and the mcp\_can libraries in this folder. To ensure that both libraries are imported properly in your arduino code, use the following include statements, in this order:

#include <SPI.h>

#include <mcp\_can.h>

#include <oneCAN.h>

Data sending functions:

void sendFloatData(unsigned int dataType, float f1, float f2)

void sendIntData(unsigned int dataType, int i1, int i2)

Heartbeat functions:

void beat(unsigned int systemID)

bool areControlsOn(void)

bool isEmergencyOn(void)

bool isRRPEOn(void)

Currently implemented CAN IDs:

CAN\_motionData

CAN\_controlsHeartbeat

CAN\_emergencyHeartbeat

CAN\_RRPEHeartbeat

The oneCAN library contains the oneCAN class, which has specific functions for sending and receiving data between the OneLoop subsystems. To use these functions, first instantiate the oneCAN class in a global variable, and call init() with the instance. To begin receiving data, call the beginReceiving() function.

The library contains a list of constant unsigned ints that store the CAN IDs for sending different kinds of data, and these must be used when sending data using the beat(), sendFloatData(), and sendIntData() functions. The sentData() functions take three arguments: the ID for the specific kind of data being sent (i.e. motion), and two floats or ints, respectively. The beat() function takes one argument, which is the heartBeat ID of a particular subsystem.

Once it is receiving data, a oneCAN object will automatically keep track of incoming heartbeats. To tell whether a specific subsystem in online, call the corresponding isOn() function, which can be found in the header file oneCAN.h. A oneCAN object will also keep track of position and velocity data sent over the CAN network, and store these in public float variables called “velocity” and “position.”