```
/**
 * @file motors.h
 * @brief Motor structure and a motor handler
 * Copyright (C) 2017 Ethan Wells
 *
 * This program is free software: you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by the Free
 * Software Foundation, either version 3 of the License, or(at your option) any
 * later version.
 *
 * This program is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
 * details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program. If not, see <https://www.gnu.org/licenses/>
 */

#ifndef CARL_MOTORS_H_
#define CARL_MOTORS_H_
#include "API.h"
#include "sensors.h"

#define clipNum(input, high, low) \
  ((input > high) ? high : (input < low) ? low : input)
#define sgn(input) ((input > 0) ? 1 : (input < 0) ? -1 : 0)
#define deadBand(input, \
                 dead) ((input - dead > 0 || input + dead < 0) ? input : 0)
#define expand(input, tip, high, low) ((input > tip) ? high : low)
#define new(type) ((type *)malloc(sizeof(type)))
#define delete(pointer) free((void *)pointer)
#define create(type, name, value) new(type); *name = value

/**
 * A convienence to distinguish tasks from regular functions
 */
typedef void Task;

/**
 * A motor structure, containing a motor's port, invertation, and power
 */
typedef struct Motor {
        /** The child in the linked list of Motors */
        struct Motor *child;
```

```
        /** The controlling Sensor to be used by default for PID */
        Sensor *sensor;

        /** Cortex port, 1-10 that the motor is plugged in to */
        unsigned char port;

        /** Whether or not the Motor is inverted */
        bool isInverted;

        /** Motor power */
        int power;

        /** Deadband for the Motor power */
        int deadband;

        /** Recalculation function */
        float (*recalc)(int);

        int           _power;
        int           _lastPower;
        unsigned long _lastTime;
        Mutex         _mutex;
} Motor;

/**
 * Configure a Motor
 *
 * @param port       the port that the motor is in
 * @param isInverted whether or not the motor is isInverted
 */
Motor motorCreate(unsigned char port,
                  bool          isInverted);

/**
 * Update and set a motor
 *
 * @param m the Motor to update
 */
void motorUpdate(Motor *m);

#endif // CARL_MOTORS_H_
```