

```

/**
 * @file motors.c
 * @brief Implements the Motor type and the 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/>
 */

#include "../include/motors.h"

Motor motorCreate(unsigned char port, bool isInverted) {
    Motor m = {
        .port      = clipNum(port, 10, 1),
        .isInverted = isInverted,
        .deadband   = 10,
        .recalc     = NULL,
        ._lastTime  = millis(),
        ._mutex     = mutexCreate(),
    };

    return m;
} /* motorCreate */

void motorUpdate(Motor *m) {
    if (!m) {
        return;
    }

    if (!mutexTake(m->_mutex, 5)) {
        return;
    }

    int power = deadBand(m->power, m->deadband);

    if (m->recalc)

```

```

        power = m->recalc(power);

    if (m->_lastPower != power) {
        motorSet(m->port,
            m->isInverted ? power : -power);
    }

    m->_lastPower = m->power;
    mutexGive(m->_mutex);

    if (m->child) {
        m->child->power = m->power;
        motorUpdate(m->child);
    }
} /* motorUpdate */

void motorUpdateSlew(Motor *m, float rate) {
    if (!m) {
        return;
    }

    if (!mutexTake(m->_mutex, 0)) {
        return;
    }

    m->power = deadBand(m->power, 10);
    int change = (m->power == m->_power) ? 0 :
        (int)(rate * (m->_lastTime - micros()) + 0.5);

    if (m->power < m->_power) {
        m->_power += change;

        if (m->_power > m->power) {
            m->_power = m->power;
        }
    } else if (m->power > m->_power) {
        m->_power -= change;

        if (m->_power < m->power) {
            m->_power = m->power;
        }
    }

    if (m->_lastPower != m->_power) {
        motorSet(m->port,
            m->isInverted ? m->_power : -m->_power);
    }
}

```

```
    }

    m->_lastPower = m->_power;
    mutexGive(m->_mutex);

    if (m->child) {
        m->child->power = m->power;
        motorUpdateSlew(m->child, rate);
    }
} /* motorUpdateSlew */
```