Motors

Functions

int **get_motor_position_counter** (int **motor**)
Gets the current motor position. More...

int **gmpc** (int **motor**)
Gets the current motor position. More...

void **clear_motor_position_counter** (int **motor**)
Clears the motor position counter. More...

void **cmpc** (int **motor**)
Clears the motor position counter. More...

int **move_at_velocity** (int **motor**, int velocity)
Set a goal velocity in ticks per second. The range is -1500 to 1500, though motor position accuracy may be decreased outside of -1000 to 1000. More...

int **mav** (int **motor**, int velocity)
Set a goal velocity in ticks per second. More...

int **move_to_position** (int **motor**, int speed, int goal_pos)
Set a goal position (in ticks) for the motor to move to. There are approximately 1500 ticks per motor revolution. This function is more accurate if speeds between -1000 and 1000 are used. More...

int **mtp** (int **motor**, int speed, int goal_pos)
Set a goal position (in ticks) for the motor to move to. More...

int **move_relative_position** (int **motor**, int speed, int delta_pos)
Set a goal position (in ticks) for the motor to move to, relative to the current position. More...

int **mrp** (int **motor**, int speed, int delta_pos)
Set a goal position (in ticks) for the motor to move to, relative to the current position. More...

void **set_pid_gains** (int **motor**, short p, short i, short d, short pd, short id, short dd)
Set the motor PID gains, represented as fractions. More...

void **get_pid_gains** (int **motor**, short *p, short *i, short *d, short *pd, short *id, short *dd)
Set the motor PID gains, represented as fractions. More...

int **freeze** (int **motor**)
Active braking to stop a motor. More...

int **get_motor_done** (int **motor**)
Check if the motor has reached it's goal. More...

void **block_motor_done** (int **motor**)
Wait until the motor is at it's goal. More...

void **bmd** (int **motor**)
Wait until the motor is at it's goal. More...

int	**setpwm** (int **motor**, int pwm) Set the motor pwm (percent power) command. More...
int	**getpwm** (int **motor**) Get the current motor pwm command. More...
void	**fd** (int **motor**) Moves the given motor forward at full power. More...
void	**bk** (int **motor**) Moves the given motor backward at full power. More...
void	**motor** (int motor, int percent) Moves a motor at a percent velocity. More...
void	**motor_power** (int **motor**, int percent) Moves a motor at a percent power. More...
void	**off** (int **motor**) Turns the specified motor off. More...
void	**alloff** () Turns all motors off. More...
void	**ao** () Turns all motors off. More...

Detailed Description

Function Documentation

void alloff ( )

Turns all motors off.

See also

ao

void ao ( )

Turns all motors off.

See also

alloff

void bk (int motor)

Moves the given motor backward at full power.

Parameters

motor the motor's port.

void block_motor_done (int motor)

Wait until the motor is at it's goal.

Parameters

`[in]` **motor** The motor port.

See also

bmd

void bmd (int motor)

Wait until the motor is at it's goal.

Parameters

`[in]` **motor** The motor port.

See also

block_motor_done

void clear_motor_position_counter (int motor)

Clears the motor position counter.

Parameters

`[in]` **motor** The motor port.

See also

cmpc

void cmpc (int motor)

Clears the motor position counter.

Parameters

`[in]` **motor** The motor port.

See also

clear_motor_position_counter

void fd (int motor)

Moves the given motor forward at full power.

Parameters

motor the motor's port.

int freeze (int motor)

Active braking to stop a motor.

Parameters

`[in]` **motor** The motor port.

int get_motor_done (int motor)

Check if the motor has reached it's goal.

Parameters

`[in]` **motor** The motor port.

Returns

1: at goal 0: not at goal

int get_motor_position_counter (int motor)

Gets the current motor position.

Parameters

`[in]`	**motor**	The motor port.

See also

gmpc

void get_pid_gains (int motor, short * p, short * i, short * d, short * pd, short * id, short * dd)

```
void get_pid_gains ( int     motor,
                     short * p,
                     short * i,
                     short * d,
                     short * pd,
                     short * id,
                     short * dd
                   )
```

Set the motor PID gains, represented as fractions.

Parameters

`[out]`	**motor**	The motor port.
`[out]`	**p**	The P (proportional) gain numerator
`[out]`	**i**	The I (integral) gain numerator
`[out]`	**d**	The D (derivative) gain numerator
`[out]`	**pd**	The P (proportional) gain denominator
`[out]`	**id**	The I (integral) gain denominator
`[out]`	**dd**	The D (derivative) gain denominator

int getpwm (int motor)

Get the current motor pwm command.

Parameters

`[in]`	**motor**	The motor port.

int gmpc (int motor)

Gets the current motor position.

Parameters

`[in]` **motor** The motor port.

See also

get_motor_position_counter

int mav (int motor,
int velocity
)

Set a goal velocity in ticks per second.

Parameters

`[in]`	**motor**	The motor port.
`[in]`	**velocity**	The goal velocity in -1500 to 1500 ticks / second

See also

move_at_velocity

void motor (int motor,
int percent
)

Moves a motor at a percent velocity.

Parameters

`[in]`	**motor**	The motor port.
`[in]`	**percent**	The percent of the motors velocity, between -100 and 100.

```
void motor_power ( int  motor,
                   int  percent
                 )
```

Moves a motor at a percent power.

Parameters

[in]	**motor**	the motor port.
[in]	**percent**	The power of the motor, between -100 and 100.

```
int move_at_velocity ( int  motor,
                       int  velocity
                     )
```

Set a goal velocity in ticks per second. The range is -1500 to 1500, though motor position accuracy may be decreased outside of -1000 to 1000.

Parameters

[in]	**motor**	The motor port.
[in]	**velocity**	The goal velocity in -1500 to 1500 ticks / second

See also

mav

```
int move_relative_position ( int  motor,
                             int  speed,
                             int  delta_pos
                           )
```

Set a goal position (in ticks) for the motor to move to, relative to the current position.

Parameters

[in]	**motor**	The motor port.
[in]	**speed**	The speed to move at, between -1500 and 1500 ticks / second
[in]	**delta_pos**	The position to move to (in ticks) given the current position

See also

mrp

```
int move_to_position ( int  motor,
                       int  speed,
                       int  goal_pos
                     )
```

Set a goal position (in ticks) for the motor to move to. There are approximately 1500 ticks per motor revolution. This function is more accurate if speeds between -1000 and 1000 are used.

Parameters

[in]	**motor**	The motor port.
[in]	**speed**	The speed to move at, between -1500 and 1500 ticks / second
[in]	**goal_pos**	The position to move to (in ticks)

See also

mtp

```
int mrp ( int  motor,
          int  speed,
          int  delta_pos
        )
```

Set a goal position (in ticks) for the motor to move to, relative to the current position.

Parameters

[in]	**motor**	The motor port.
[in]	**speed**	The speed to move at, between -1500 and 1500 ticks / second
[in]	**delta_pos**	The position to move to (in ticks) given the current position

See also

move_relative_position

```
int mtp ( int  motor,
          int  speed,
          int  goal_pos
        )
```

Set a goal position (in ticks) for the motor to move to.

Parameters

[in]	**motor**	The motor port.
[in]	**speed**	The speed to move at, between -1500 and 1500 ticks / second
[in]	**goal_pos**	The position to move to (in ticks)

See also

move_to_position

```
void off ( int  motor )
```

Turns the specified motor off.

Parameters

motor the motor's port.

```
void set_pid_gains ( int    motor,
                     short  p,
                     short  i,
                     short  d,
                     short  pd,
                     short  id,
                     short  dd
                   )
```

Set the motor PID gains, represented as fractions.

Parameters

[in]	**motor**	The motor port.
[in]	**p**	The P (proportional) gain numerator
[in]	**i**	The I (integral) gain numerator
[in]	**d**	The D (derivative) gain numerator
[in]	**pd**	The P (proportional) gain denominator
[in]	**id**	The I (integral) gain denominator
[in]	**dd**	The D (derivative) gain denominator

```
int setpwm ( int  motor,
             int  pwm
           )
```

Set the motor pwm (percent power) command.

Parameters

[in]	**motor**	The motor port.
[in]	**pwm**	A new motor pwm command between 0 and 100