# MegaMoto Documentation

Generated by Doxygen 1.8.12

# Contents

# Chapter 1

# MegaMoto Arduino Library

An Arduino library for interfacing with the MegaMoto family of motor controllers by Robot Power. The "MegaMogo" name is trademark of Robot Power. You can find and purchase their hardware by visiting their website: www.robotpower.com. I have been granted written permission by Robot Power to use the "MegaMoto" name in the library and associated code as long as I point people to their company name and website. *I encourage you to go check out their stuff!*

## What is this?

This is a library to (hopefully) simplify the interface with Robot Power's "MegaMoto" platform(s).

**NOTE: I have currently only have access to an Arduino UNO and the MegaMoto PLUS** so testing has only been performed with that setup. I cannot garuntee anything working for another configuration, although, by what I read in their documentation, it should be similar enough to be just fine. You have been warned. :-)

## Features

- **Easy pin configuration.** If you decided to change which pins you use for the PWM inputs or Enable control pin, simply change a few numbers when you instantiate a MegaMoto object and you're done! The library will take care of the rest for you.
- **No pin hogging.** Won't touch any pins it doesn't have to. (Example: You chose to set the MegaMoto Enable to a constant 5V, if you use the correct class constructor, the code won't attempt to use the Enable pin.
- **Motor friendly.** Can "ramp" the motor speed instead of abruptly changing from, say, forward to reverse. This is much nicer to the motor you are controlling and (tries) to avoide unneeded wear and tear on your hardware.
- **Industrial motor ready.** I've been testing this with an industrial-grade motor roller (sometimes called a "moller").
- **Understands your hardware config.** While it may not be completely developed just yet, I have tried to lay down the foundations for the library to handle your specific hardware wiring. Whether it be the I/O pin configuration on the Arduino side, or the H-bridge vs. half-bridge motor/device wiring hookup on the MegaMoto output pins, the library allows you to tell it what you've got and it (should) be able to act accordingly. (Take this "feature" with a grain of salt. Like I said, things are still under development.)

## Status

[X] = **Working feature.** I've tested it with my hardware setup and it seems to work great.

[ ] = **Planned feature.** I might have a few lines in the code relating to this concept, but it's either not working yet or I don't have a way of testing it.

[X] Configurable PWMA / PWMB pin assignments

[X] Configurable Enable pin assignment (If no Enable pin is provided, it won't attempt to use Enable at all!)

[X] Handles the H-bridge hardware configuration

[ ] Handles the half-bridge hardware configuration

[X] Ramps motor up/down from current speed to the new speed

[X] Immediate stop/off. If there is some safety-crtical setting, you can shut down your device immediatly without ramping down first.

[ ] Current sense reporting. Reads the voltage from the current sense output on the MegaMoto and converts it to milliamps (mA) and returns the milliamp value as an integer.

## Getting Started

It is assumed that you have already read or understand the options provided by the MegaMoto hardware. You should know how you want to physically configure the controller and wire up your device. I don't want to repeat any of their documentation. It was very well written. So, **if you have not yet read the MegaMoto User Guide, please do so.** Links are provided in the "Useful Links" section at the bottom of this page.

1. Download this library and unzip in your Arduino/libraries folder.
   - `Download the latest .ZIP file`
   - See `these instructions` to add the library to your Arduino IDE installation
2. Choose the appropriate class to use
   - H-bridge wiring on MegaMoto output = `#include <MegaMotoHB.h>`
   - Half-bridge wiring on output = `#include <MegaMotoHalfB.h>`
3. Delcare a motor object and specify the PWM A and PWM B pins you chose to use.
4. Call one of the motion functions, like `::Fwd()`, `::Rev()` to start up your device

**ATTN:** The `/examples` folder has been created for a reason! It might be a lot easier to see how to get started by checking out the example projects.

For more information about how to use this library, plese visit the `/doc` folder of this repository (or your downloaded .ZIP file.

## Useful Links

- Robot Power website: `http://www.robotpower.com/`
- MegaMoto info page: `http://www.robotpower.com/products/MegaMoto_info.html`
- MegaMoto Plus info page: `http://www.robotpower.com/products/MegaMotoPlus_info.html`
- MegaMoto and MegaMoto Plus User's guide: `http://www.robotpower.com/downloads/MegaMoto-user-manual.pdf`
- MegaMoto and MegaMoto Plus schematic: `http://www.robotpower.com/downloads/MegaMoto-v1.5-schematic.pdf`

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 3

# Class Index

## 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 5

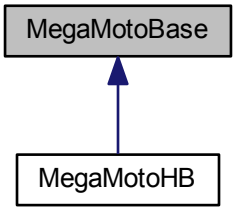
# Class Documentation

## 5.1 MegaMotoBase Class Reference

Provides an easy way to interface with the MegaMoto controller by Robot Power.

`#include <MegaMotoBase.h>`

Inheritance diagram for MegaMotoBase:




### Public Member Functions

- MegaMotoBase (unsigned char pin_pwm_a, unsigned char pin_pwm_b)

  *Constructor to initialize with pin numbers for PWM A and PWM B and specify the motor wiring.*
- MegaMotoBase (unsigned char pin_pwm_a, unsigned char pin_pwm_b, unsigned char pin_enable)

  *Constructor to initialize with pin numbers for PWM A, PWM B, and Enable. Also specify the motor wiring.*
- void SetStepDelay (int new_ms_delay)

  *Setter to change the ms delay when stepping up or down the PWM duty cycle.*
- int GetStepDelay () const

  *Getter which returns the current delay used when changing the PWM duty cycle.*
- void Kill ()

  *Immediatly turn off the output.*
- void Enable ()

  *Enable the MegaMoto controller.*
- virtual void **Disable** ()=0

### Protected Attributes

- const bool use_enable_pin

  *TRUE if Enable pin will be controlled by this object.*
- const unsigned char pin_en

  *Arduino pin number of the Enable line.*
- const unsigned char pin_a

  *Arduino pin number of MegaMoto PWM A.*
- const unsigned char pin_b

  *Arduino pin number of MegaMoto PWM B.*
- unsigned char pwm_step_delay_ms

  *Defines the delay in milliseconds used when "fading" the PWM duty cycle up or down.*

## 5.1.1 Detailed Description

Provides an easy way to interface with the MegaMoto controller by Robot Power.

This is a parent class. You should not be directly using this class in your code. Instead, use MegaMotoHB or MegaMotoHalfB in your code, depending on your device hookup configuration.

Please see the example file(s) for, well... examples! ;-)

## 5.1.2 Constructor & Destructor Documentation

### 5.1.2.1 MegaMotoBase() [1/2]

```
MegaMotoBase::MegaMotoBase (
            unsigned char pin_pwm_a,
            unsigned char pin_pwm_b )
```

Constructor to initialize with pin numbers for PWM A and PWM B and specify the motor wiring.

All data members initialized to their empty or default values. The name of the multiplier is set to the string passed in as the first argument. The multiplier bit width is set to the value of the second argument.

**Parameters**

| | |
|---|---|
| *pin_pwm_a* | is a byte. Specify the pin used for MegaMoto PWM A here. |
| *pin_pwm_b* | is a byte. Specify the pin used for MegaMoto PWM B here. |

### 5.1.2.2 MegaMotoBase() [2/2]

```
MegaMotoBase::MegaMotoBase (
            unsigned char pin_pwm_a,
            unsigned char pin_pwm_b,
            unsigned char pin_enable )
```

Constructor to initialize with pin numbers for PWM A, PWM B, and Enable. Also specify the motor wiring.

All data members initialized to their empty or default values. The name of the multiplier is set to the string passed in as the first argument. The multiplier bit width is set to the value of the second argument.

**Parameters**

| | |
|---|---|
| *pin_pwm_a* | is a byte. Specify the pin used for MegaMoto PWM A here. |
| *pin_pwm_b* | is a byte. Specify the pin used for MegaMoto PWM B here. |
| *pin_enable* | is a byte. Specify the pin used for the MegaMoto Enable pin here. |

### 5.1.3 Member Function Documentation

#### 5.1.3.1 Enable()

```
void MegaMotoBase::Enable ( )
```

Enable the MegaMoto controller.

Used when the Enable pin is being controlled to enable the MegaMoto controller. If you have called Disable(), you will probably need to call this function again.

#### 5.1.3.2 GetStepDelay()

```
int MegaMotoBase::GetStepDelay ( ) const
```

Getter which returns the current delay used when changing the PWM duty cycle.

Returns the delay time in milliseconds used when "fading/feathering/sliding/ramping" the PWM duty cycle up or down.

**Returns**

integer value of pwm_step_delay_ms

#### 5.1.3.3 Kill()

```
void MegaMotoBase::Kill ( )
```

Immediatly turn off the output.

Might be good to use for some safety-critical situation where slowing down the motor/device before turning it off is a *bad* idea.

#### 5.1.3.4 SetStepDelay()

```
void MegaMotoBase::SetStepDelay (
            int new_ms_delay )
```

Setter to change the ms delay when stepping up or down the PWM duty cycle.

Provide the new delay time in milliseconds to this method if you want to change the delay time when "fading/feathering/sliding/ramping" the PWM duty cycle up or down.

**Parameters**

| | |
|---|---|
| *new_ms_delay* | is an int |

### 5.1.4 Member Data Documentation

#### 5.1.4.1 pwm_step_delay_ms

`MegaMotoBase::pwm_step_delay_ms [protected]`

Defines the delay in milliseconds used when "fading" the PWM duty cycle up or down.

This defines the value used in a delay() call in the StepPwmDuty() method. Therefore, this value is in milliseconds (ms). To slow the changes in speed (make the change longer), increase this value.

The documentation for this class was generated from the following files:

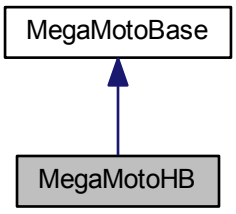
- MegaMotoBase.h
- MegaMotoBase.cpp

## 5.2 MegaMotoHB Class Reference

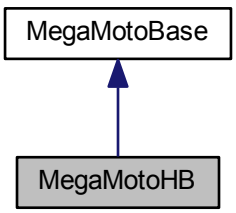
MegaMoto class for H-bridge driving configurations.

`#include <MegaMotoHB.h>`

Inheritance diagram for MegaMotoHB:




Collaboration diagram for MegaMotoHB:



### Public Types

- enum TMegaMotoHBDir { FWD, REV, STOP }

  *Defines the three possible output directions.*

### Public Member Functions

- MegaMotoHB (unsigned char pin_pwm_a, unsigned char pin_pwm_b)

  *Constructor to initialize with pin numbers for PWM A and PWM B and specify the motor wiring.*
- MegaMotoHB (char pin_pwm_a, char pin_pwm_b, char pin_enable)

  *Constructor to initialize with pin numbers for PWM A, PWM B, and Enable.*
- void **Fwd** (const unsigned char new_pwm_duty)
- void **Rev** (const unsigned char new_pwm_duty)
- void **Stop** ()
- void Power (const TMegaMotoHBDir dir, const unsigned char pwm_duty)

  *Change the power output, either speed (power strength) or direction (polarity)*
- void StepPwmDuty (unsigned char pwm_duty_in)

  *Gradually change the PWM output from what it was to the new desired PWM duty cycle.*
- void Disable ()

  *"Gracefully" power down the motor/device and then (if applicable) disable the MegaMoto.*

### Additional Inherited Members

## 5.2.1 Detailed Description

MegaMoto class for H-bridge driving configurations.

This is the class used when you are using a single motor/device with both (+) and (-) terminals hooked up directly to the A and B outputs of the MegaMoto controller.

See section "3. Application Configurations" in the MegaMoto user manual for more wiring information.

Please see the example file(s) for, well... examples! ;-)

## 5.2.2 Member Enumeration Documentation

### 5.2.2.1 TMegaMotoHBDir

```
enum MegaMotoHB::TMegaMotoHBDir
```

Defines the three possible output directions.

Thinking from a single motor wired in H-bridge mode, there are three possible directions

**Enumerator**

| | |
|---|---|
| FWD | Forward direction: PWM A is on at a duty cycle defined by pwm_duty. PWM B is off. MegaMoto output MA will have a higher potential than MB. |
| REV | Reverse direction: PWMA is off. PWMB is on at a duty cycle defined by pwm_duty. MegaMoto output MA will have a higher potential than MB. |
| STOP | Motor is off: PWMA is off. PWMB is off. MegaMoto output MA will have the same potential as MB (which is 0V). |

### 5.2.3 Constructor & Destructor Documentation

#### 5.2.3.1 MegaMotoHB() [1/2]

```
MegaMotoHB::MegaMotoHB (
            unsigned char pin_pwm_a,
            unsigned char pin_pwm_b )
```

Constructor to initialize with pin numbers for PWM A and PWM B and specify the motor wiring.

Initializes the data members and sets up the Arduino pins appropriately to be able to talk to the MegaMoto properly.

**Parameters**

| | |
|---|---|
| *pin_pwm_a* | is a byte. Specify the pin used for MegaMoto PWM A here. |
| *pin_pwm_b* | is a byte. Specify the pin used for MegaMoto PWM B here. |

#### 5.2.3.2 MegaMotoHB() [2/2]

```
MegaMotoHB::MegaMotoHB (
            char pin_pwm_a,
            char pin_pwm_b,
            char pin_enable )
```

Constructor to initialize with pin numbers for PWM A, PWM B, and Enable.

Initializes the data members and sets up the Arduino pins appropriately to be able to talk to the MegaMoto properly.

**Parameters**

| | |
|---|---|
| *pin_pwm_a* | is a byte. Specify the pin used for MegaMoto PWM A here. |
| *pin_pwm_b* | is a byte. Specify the pin used for MegaMoto PWM B here. |
| *pin_enable* | is a byte. Specify the pin used for the MegaMoto Enable pin here. |

### 5.2.4 Member Function Documentation

#### 5.2.4.1 Disable()

```
void MegaMotoHB::Disable ( )  [virtual]
```

"Gracefully" power down the motor/device and then (if applicable) disable the MegaMoto.

If the Enable pin IS being used, you will need to call Enable() to resume operation after using Disable().

Implements MegaMotoBase.

Here is the call graph for this function:


#### 5.2.4.2 Power()

```
void MegaMotoHB::Power (
            const TMegaMotoHBDir dir,
            const unsigned char new_pwm_duty )
```

Change the power output, either speed (power strength) or direction (polarity)

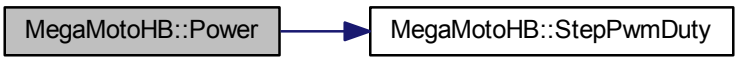
Use this method to change the current output state of the MegaMoto. If the device being controlled is a motor, then this adjusts the direction (polarity) and speed (PWM duty cycle). However, for a light or some other device, you might call it changing the voltage direction and power/brightness.

Calls StepPwmDuty() to gradually adjust the old PWM duty cycle to the new PWM duty cycle (i.e. If going from TMegaMotoDir::FWD to TMegaMotoDir::REV, this method will slow down in the FWD direction, and then speed up in the REV direction.)

**Parameters**

| | |
|---|---|
| *dir* | is the new motor direction, defined by TMegaMotoDir states. |
| *new_pwm_duty* | is the new PWM duty cycle or power output in the aforementioned direction. Valid values are 0 - 255, so it is a byte datatype. |

Here is the call graph for this function:



Here is the caller graph for this function:


#### 5.2.4.3 StepPwmDuty()

```
void MegaMotoHB::StepPwmDuty (
            unsigned char pwm_duty_in )
```

Gradually change the PWM output from what it was to the new desired PWM duty cycle.

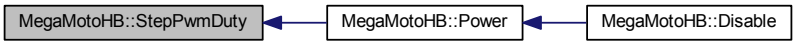
Since motors and many other physical things don't like sudden changes, it is a good idea to fade between different speeds/power levels. See the definition of MEGA_MOTO_FEATHER_STEP for the actual fade speed.

Called by the Power() method.

**Parameters**

| | |
|---|---|
| *pwm_duty_in* | is the new desired duty cycle. Acceptable values range from 0 - 255. |

Here is the caller graph for this function:




The documentation for this class was generated from the following files:

- MegaMotoHB.h
- MegaMotoHB.cpp

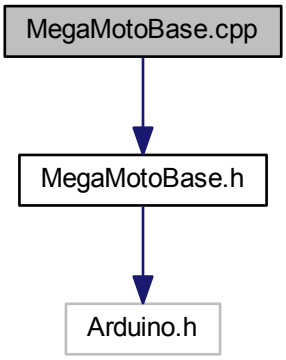
# Chapter 6

# File Documentation

## 6.1 MegaMotoBase.cpp File Reference

Contains the definitions for all the member methods of the MegaMotoBase class.

`#include "MegaMotoBase.h"`
Include dependency graph for MegaMotoBase.cpp:




### 6.1.1 Detailed Description

Contains the definitions for all the member methods of the MegaMotoBase class.

**Author**

Alexander Hogen

**Date**

1/3/2017

**Version**

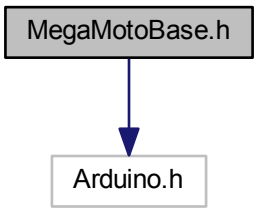
0.1 Released into the public domain.

## 6.2 MegaMotoBase.h File Reference

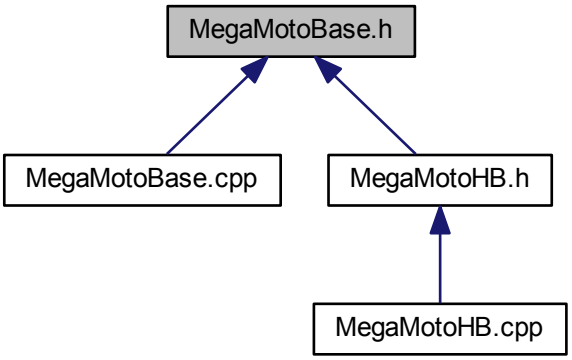
An object oriented Arduino library to control the MegaMoto Plus by Robot Power.

```
#include <Arduino.h>
```

Include dependency graph for MegaMotoBase.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class MegaMotoBase

  *Provides an easy way to interface with the MegaMoto controller by Robot Power.*

### 6.2.1 Detailed Description

An object oriented Arduino library to control the MegaMoto Plus by Robot Power.

**Author**

Alexander Hogen

**Date**

1/3/2017

**Version**

0.1 This is a base class for the MegaMoto library. You should not be including this header in your code. Instead, go look at the MegaMotoHB or MegaMotoHalfB classes. You should use one of those in your project.

NOTE: References to MegaMoto, MegaMoto Plus, and Robot Power are either reserved or trademarked by Robot Power:

2745 Martin Way E, Suite D Olympia, WA 98506 http://www.robotpower.com/

I do not own or claim ownership to these names.

Released into the public domain.

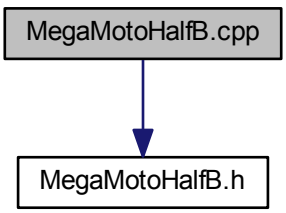
[1] "MegaMoto & MegaMoto Plus User Manual," Robot Power, Version 1.6, May 28, 2016. Avaliable: http://www.robotpower.com/downloads/MegaMoto-user-manual.pdf

## 6.3 MegaMotoHalfB.cpp File Reference

Contains the definitions for all the member methods of the MegaMotoHalfB class.

```
#include "MegaMotoHalfB.h"
```

Include dependency graph for MegaMotoHalfB.cpp:



### 6.3.1 Detailed Description

Contains the definitions for all the member methods of the MegaMotoHalfB class.

**Author**

Alexander Hogen

**Date**

1/3/2017

**Version**

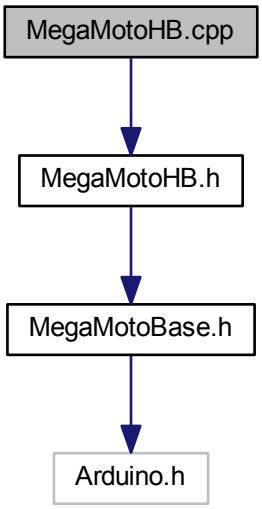
0.1 Released into the public domain.

## 6.4 MegaMotoHB.cpp File Reference

Contains the definitions for all the member methods of the MegaMotoHB class.

```
#include "MegaMotoHB.h"
```

Include dependency graph for MegaMotoHB.cpp:



### 6.4.1 Detailed Description

Contains the definitions for all the member methods of the MegaMotoHB class.

**Author**

Alexander Hogen

**Date**

1/3/2017

**Version**

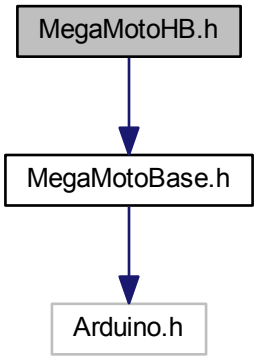
0.1 Released into the public domain.

## 6.5 MegaMotoHB.h File Reference

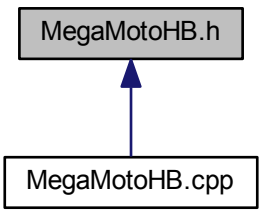
H-Bridge configuration – An Arduino library to control the MegaMoto motor controller family by Robot Power.

```
#include "MegaMotoBase.h"
```

Include dependency graph for MegaMotoHB.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class MegaMotoHB

  *MegaMoto class for H-bridge driving configurations.*

### 6.5.1 Detailed Description

H-Bridge configuration – An Arduino library to control the MegaMoto motor controller family by Robot Power.

**Author**

Alexander Hogen

**Date**

1/3/2017

**Version**

0.1 The MegaMotoHalfB class is a derived class from the MegaMotoBase base class. You should use the MegaMotoHalfB class if your device is wired in a half-bridge configuration (hence the "HalfB" at the end of the class name). If You do not know what this means, then you probably haven't checked out the MegaMoto User Guide written by the people who made it, Robot Power! Go have a look first. Things over here might make more sense afterwards. Trust me.

NOTE: References to MegaMoto, MegaMoto Plus, or Robot Power are either reserved or trademarked by Robot Power:

2745 Martin Way E, Suite D Olympia, WA 98506 http://www.robotpower.com/

I do not own or claim ownership to these names.

Released into the public domain.

[1] "MegaMoto & MegaMoto Plus User Manual," Robot Power, Version 1.6, May 28, 2016. Avaliable: http://www.robotpower.com/downloads/MegaMoto-user-manual.pdf

**Author**

Alexander Hogen

**Date**

1/3/2017

**Version**

0.1 The MegaMotoHB class is a derived class from the MegaMotoBase base class. You should use the MegaMotoHB class if your device you are controlling is wired in an H-bridge configuration (hence the "HB" at the end of the class name). If You do not know what this means, then you probably haven't checked out the MegaMoto User Guide written by the people who made it, Robot Power! Go have a look first. Things over here might make more sense afterwards. Trust me.

NOTE: References to MegaMoto, MegaMoto Plus, or Robot Power are either reserved or trademarked by Robot Power:

2745 Martin Way E, Suite D Olympia, WA 98506 http://www.robotpower.com/

I do not own or claim ownership to these names.

Released into the public domain.

[1] "MegaMoto & MegaMoto Plus User Manual," Robot Power, Version 1.6, May 28, 2016. Avaliable: http://www.robotpower.com/downloads/MegaMoto-user-manual.pdf

# Index