

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples
Class List	Class Index	Class Members	
<h2>Class List</h2>			
Here are the classes, structs, unions and interfaces with brief descriptions:			

 USBSabertooth	Controls a USB Sabertooth motor driver running in Packet Serial mode
 USBSabertoothSerial	Create a USBSabertoothSerial for the serial port you are using, and then attach a USBSabertooth for each motor driver you want to communicate with

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples
Class List	Class Index	Class Members	
Public Member Functions List of all members			
USBSabertooth Class Reference			

Controls a USB Sabertooth motor driver running in Packet Serial mode. [More...](#)

Public Member Functions

**USBSabertooth (USBSabertoothSerial &serial,
byte address)**

byte **address () const**

void **command (byte command, byte value)**

void **command (byte command, const byte *value,
size_t bytes)**

void **motor (int value)**

void **motor (byte motorOutputNumber, int value)**

void **power (int value)**

void **power (byte powerOutputNumber, int value)**

void **drive (int value)**

void **turn (int value)**

void **freewheel (int value=2048)**

void **freewheel (byte motorOutputNumber, int
value=2048)**

void **shutDown (byte type, byte number, boolean
value=true)**

```
void set (byte type, byte number, int value)
```

```
void setRamping (int value)
```

```
void setRamping (byte motorOutputNumber, int value)
```

```
void setTimeout (int milliseconds)
```

```
void keepAlive ()
```

```
int get (byte type, byte number)
```

```
int getBattery (byte motorOutputNumber, boolean  
unscaled=false)
```

```
int getCurrent (byte motorOutputNumber, boolean  
unscaled=false)
```

```
int getTemperature (byte motorOutputNumber,  
boolean unscaled=false)
```

```
int32_t getGetRetryInterval () const
```

```
void setGetRetryInterval (int32_t intervalMS)
```

```
int32_t getGetTimeout () const
```

```
void setGetTimeout (int32_t timeoutMS)
```

```
boolean usingCRC () const
```

```
void useChecksum()
```

```
void useCRC()
```

Detailed Description

Controls a USB Sabertooth motor driver running in Packet Serial mode.

Examples:

- [1.Basics/Freewheeling/Freewheeling.ino](#),
- [1.Basics/PowerOutputs/PowerOutputs.ino](#),
- [1.Basics/Simplest/Simplest.ino](#),
- [1.Basics/Sweep/Sweep.ino](#),
- [1.Basics/TankStyleSweep/TankStyleSweep.ino](#),
- [2.Settings/Ramping/Ramping.ino](#),
- [2.Settings/SerialTimeout/SerialTimeout.ino](#),
- [3.Advanced/Checksum/Checksum.ino](#),
- [3.Advanced/SharedLine/SharedLine.ino](#), and
- [3.Advanced/SoftwareSerial/SoftwareSerial.ino](#).

Constructor & Destructor Documentation

**USBSabertooth::USBSabertooth ([USBSabertoothSerial](#) &
byte
)**

Initializes a new instance of the [USBSabertooth](#) class. The driver address is set to the value given, and the specified serial port

Parameters

serial The [USBSabertoothSerial](#) whose serial port the driver is on.

address The driver address.

Member Function Documentation

byte USBSabertooth::address() const

inline

Gets the driver address.

Returns

The driver address.

**void USBSabertooth::command(byte command,
 byte value
)**

Sends a packet serial command to the motor driver.

Parameters

command The number of the command.

value The command's value.

**void
USBSabertooth::command(byte command,
 const byte * value,
 size_t bytes
)**

Sends a multibyte packet serial command to the motor driver.

Parameters

- command** The number of the command.
- value** The command's value.
- bytes** The number of bytes in the value.

void USBSabertooth::drive (int value)

Controls the mixed-mode drive channel. In User Mode, this sets MD.

Parameters

- value** The value, between -2047 and 2047.

void USBSabertooth::freewheel (int value = 2048)

Causes motor output 1 to freewheel. In User Mode, this sets Q1.

Parameters

- value** true or a positive value lets the motor outputs freewheel. false or a negative or zero value stops the freewheeling.

void USBSabertooth::freewheel (byte motorOutputNumber, int value = 2048)

Causes the specified motor output to freewheel. In User

Mode, this sets Q1 or Q2.

Parameters

motorOutputNumber The motor output number, 1 or 2. You can also use a character, such as '3', to select the motor output by its Plain Text Serial address.

value true or a positive value lets the motor output freewheel. false or a negative or zero value stops the freewheeling.

```
int USBSabertooth::get ( byte type,  
                           byte number  
 )
```

inline

Gets a value from the motor driver.

Parameters

type The type of channel to get from. This can be 'S' (signal), 'A' (aux), 'M' (motor output), or 'P' (power output).

number The number of the channel, 1 or 2. You can also use a character, such as '3', to select by Plain Text Serial address.

Returns

The value, or SABERTOOTH_GET_TIMED_OUT.

```
int ( byte motorOutputNumber
USBSabertooth::getBattery boolean unscaled = false
)
```

Gets the battery voltage.

Parameters

motorOutputNumber The number of the motor output, 1
You can also use a character, such
'3', to select by Plain Text Serial ad
unscaled If true, gets in unscaled units. If fal
gets in scaled units.

Returns

The value, or SABERTOOTH_GET_TIMED_OUT.

```
int
USBSabertooth::getCurrent ( byte motorOutputNumber
                            boolean unscaled = false
)
```

Gets the motor output current.

Parameters

motorOutputNumber The number of the motor output, 1
You can also use a character, such
to select by Plain Text Serial addre
unscaled If true, gets in unscaled units. If fal
gets in scaled units.

Returns

The value, or SABERTOOTH_GET_TIMED_OUT.

int32_t

USBSabertooth::getGetRetryInterval

() const

inline

Gets the get retry interval.

Returns

The get retry interval, in milliseconds.

int32_t USBSabertooth::getGetTimeout () const

inline

Gets the get timeout.

Returns

The get timeout, in milliseconds.

int

USBSabertooth::getTemperature (byte motorOutputN

boolean unscaled = false)

Gets the motor output temperature.

Parameters

motorOutputNumber The number of the motor output, 1 can also use a character, such as 'b' by Plain Text Serial address.

unscaled If true, gets in unscaled units. If false,

scaled units.

Returns

The value, or SABERTOOTH_GET_TIMED_OUT.

void USBSabertooth::keepAlive ()

Resets the serial timeout. This is done automatically any time a motor output is set. You can, however, call this if you don't want to set any motor outputs.

void USBSabertooth::motor (int value)

Controls motor output 1. In User Mode, this sets M1.

Parameters

value The value, between -2047 and 2047.

Examples:

[3.Advanced/SharedLine/SharedLine.ino](#).

```
void USBSabertooth::motor ( byte motorOutputNumber,
                           int      value
                           )
```

Controls the specified motor output. In User Mode, this sets M1 or M2.

Parameters

motorOutputNumber The motor output number, 1 or

2. You can also use a character, such as '3', to select the motor output by its Plain Text Serial address.

value

The value, between -2047 and 2047.

void USBSabertooth::power (int **value)**

Controls power output 1, if power output 1 is configured as a controllable output. In User Mode, this sets P1.

Parameters

value The value, between -2047 and 2047.

```
void
USBSabertooth::power      ( byte powerOutputNumber,
                                int   value
                           )
```

Controls the specified power output, if the power output is configured as a controllable output. In User Mode, this sets P1 or P2.

Parameters

powerOutputNumber The power output number, 1 or 2. You can also use a character, such as '3', to select the power output by its Plain Text Serial address.

value The value, between -2047 and 2047.

```
void USBSabertooth::set ( byte type,
                         byte number,
                         int value
)
```

Sets a value on the motor driver.

Parameters

type The type of channel to set. This can be 'M' (motor output), 'P' (power output), 'Q' (freewheel), or 'R' (ramping).

number The number of the channel, 1 or 2. You can also use a character, such as '3', to select by Plain Text Serial address.

value The value, between -16383 and 16383 (though in many cases, only -2047 to 2047 are meaningful).

```
void
USBSabertooth::setGetRetryInterval ( int32_t intervalMS )
```

Sets the get retry interval.

Parameters

intervalMS The command retry interval, in milliseconds.

```
void USBSabertooth::setGetTimeout(int32_t timeoutMS) inline
```

Sets the get timeout.

Parameters

timeoutMS The get timeout, in milliseconds.

```
void USBSabertooth::setRamping ( int value )
```

Sets the ramping for all motor outputs. In User Mode, this sets R1 and R2.

Parameters

value The ramping value, between -16383 (fast) and 2047 (slow).

```
void USBSabertooth::setRamping ( byte motorOutputNumber,  
                         int value  
                         )
```

Sets the ramping for the specified motor output. In User Mode, this sets R1 or R2.

Parameters

motorOutputNumber The motor output number, 1 or 2.
You can also use a character,
such as '3', to select the motor
output by its Plain Text Serial

address.

value

The ramping value, between -16383 (fast) and 2047 (slow).

void USBSabertooth::setTimeout (int milliseconds)

Sets the serial timeout.

Parameters

milliseconds The maximum time in milliseconds between packets. If this time is exceeded, the driver will stop the motor and power outputs. A value of zero uses the DEScribe setting.
SABERTOOTH_INFINITE_TIMEOUT disables the timeout.

void USBSabertooth::shutDown (byte type, byte number, boolean value = true)

Shuts down an output.

Parameters

type The type of output to shut down. This can be 'M' (motor output) or 'P' (power output).

number The number of the output, 1 or 2. You can also use a character, such as '3', to select by Plain Text Serial address.

value true sets the shutdown. false clears the shutdown.

void USBSabertooth::turn (int value)

Controls the mixed-mode turn channel. In User Mode, this sets MT.

Parameters

value The value, between -2047 and 2047.

void USBSabertooth::useChecksum ()

inline

Causes future commands to be sent CRC-protected (larger packets, excellent error detection).

void USBSabertooth::useCRC ()

inline

Causes future commands to be sent checksum-protected (smaller packets, reasonable error detection).

boolean USBSabertooth::usingCRC () const

inline

Gets whether CRC-protected commands are used. They are, by default.

Returns

True if CRC-protected commands are used.

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples
Class List	Class Index	Class Members	
Public Member Functions List of all members			
<h2>USBSabertoothSerial Class Reference</h2>			

Create a **USBSabertoothSerial** for the serial port you are using, and then attach a **USBSabertooth** for each motor driver you want to communicate with. [More...](#)

Public Member Functions

USBSabertoothSerial (Stream
&**port**=SabertoothTXPinSerial)

Stream & **port** ()

Detailed Description

Create a **USBSabertoothSerial** for the serial port you are using, and then attach a **USBSabertooth** for each motor driver you want to communicate with.

Examples:

- [1.Basics/Freewheeling/Freewheeling.ino](#),
- [1.Basics/PowerOutputs/PowerOutputs.ino](#),
- [1.Basics/Simplest/Simplest.ino](#),
- [1.Basics/Sweep/Sweep.ino](#),
- [1.Basics/TankStyleSweep/TankStyleSweep.ino](#),
- [2.Settings/Ramping/Ramping.ino](#),
- [2.Settings/SerialTimeout/SerialTimeout.ino](#),
- [3.Advanced/Checksum/Checksum.ino](#),
- [3.Advanced/SharedLine/SharedLine.ino](#), and
- [3.Advanced/SoftwareSerial/SoftwareSerial.ino](#).

Constructor & Destructor Documentation

USBSabertoothSerial::USBSabertoothSerial (Stream & port)

Constructs a **USBSabertoothSerial** object.

Parameters

port The serial port the motor driver is on. By default, this

Member Function Documentation

Stream& USBSabertoothSerial::port ()

inline

Gets the serial port being used.

Returns

The serial port.

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples
Class List	Class Index	Class Members	
<h2>Class Index</h2>			
U		USBSabertoothSerial	

U

[USBSabertooth](#)

U

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples							
Class List	Class Index	Class Members								
All	Functions									
a	c	d	f	g	k	m	p	s	t	u

Here is a list of all documented class members with links to the class documentation for each member:

- a -

- address() : [USBSabertooth](#)

- c -

- command() : [USBSabertooth](#)

- d -

- drive() : [USBSabertooth](#)

- f -

- freewheel() : [USBSabertooth](#)

- g -

- get() : [USBSabertooth](#)
- getBattery() : [USBSabertooth](#)

- `getCurrent()` : **USBSabertooth**
- `getGetRetryInterval()` : **USBSabertooth**
- `getGetTimeout()` : **USBSabertooth**
- `getTemperature()` : **USBSabertooth**

- k -

- `keepAlive()` : **USBSabertooth**

- m -

- `motor()` : **USBSabertooth**

- p -

- `port()` : **USBSabertoothSerial**
- `power()` : **USBSabertooth**

- s -

- `set()` : **USBSabertooth**
- `setGetRetryInterval()` : **USBSabertooth**
- `setGetTimeout()` : **USBSabertooth**
- `setRamping()` : **USBSabertooth**
- `setTimeout()` : **USBSabertooth**
- `shutDown()` : **USBSabertooth**

- t -

- `turn()` : **USBSabertooth**

- u -

- `USBSabertooth()` : **USBSabertooth**
- `USBSabertoothSerial()` : **USBSabertoothSerial**

- useChecksum() : **USBSabertooth**
- useCRC() : **USBSabertooth**
- usingCRC() : **USBSabertooth**

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples							
Class List	Class Index	Class Members								
All	Functions									
a	c	d	f	g	k	m	p	s	t	u

- a -

- address() : **USBSabertooth**

- c -

- command() : **USBSabertooth**

- d -

- drive() : **USBSabertooth**

- f -

- freewheel() : **USBSabertooth**

- g -

- get() : **USBSabertooth**
- getBattery() : **USBSabertooth**
- getCurrent() : **USBSabertooth**

- `getGetRetryInterval()` : **USBSabertooth**
- `getGetTimeout()` : **USBSabertooth**
- `getTemperature()` : **USBSabertooth**

- k -

- `keepAlive()` : **USBSabertooth**

- m -

- `motor()` : **USBSabertooth**

- p -

- `port()` : **USBSabertoothSerial**
- `power()` : **USBSabertooth**

- s -

- `set()` : **USBSabertooth**
- `setGetRetryInterval()` : **USBSabertooth**
- `setGetTimeout()` : **USBSabertooth**
- `setRamping()` : **USBSabertooth**
- `setTimeout()` : **USBSabertooth**
- `shutDown()` : **USBSabertooth**

- t -

- `turn()` : **USBSabertooth**

- u -

- `USBSabertooth()` : **USBSabertooth**
- `USBSabertoothSerial()` : **USBSabertoothSerial**
- `useChecksum()` : **USBSabertooth**

- useCRC() : **USBSabertooth**
- usingCRC() : **USBSabertooth**

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

[Main Page](#)

[Classes](#)

[Files](#)

[Examples](#)

[File List](#)

File List

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

[detail level [1](#) [2](#)]

▼  **USBSabertooth**

 **USBSabertooth.h**

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

[Main Page](#)

[Classes](#)

[Files](#)

[Examples](#)

[USBSabertooth](#)

USBSabertooth Directory Reference

Files

file **USBSabertooth.cpp**

file **USBSabertooth.h** [code]

file **USBSabertoothChecksum.cpp**

file **USBSabertoothCommandWriter.cpp**

file **USBSabertoothCRC14.cpp**

file **USBSabertoothCRC7.cpp**

file **USBSabertoothReplyReceiver.cpp**

file **USBSabertoothSerial.cpp**

file **USBSabertoothTimeout.cpp**

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples
File List			
USBSabertooth			Classes

USBSabertooth.h File Reference

Go to the source code of this file.

Classes

class **USBSabertoothSerial**

Create a **USBSabertoothSerial** for the serial port you are using, and then attach a **USBSabertooth** for each motor driver you want to communicate with.

[More...](#)

class **USBSabertooth**

Controls a USB Sabertooth motor driver running in Packet Serial mode. [More...](#)

Detailed Description

Include this file to use the USB Sabertooth Arduino library.

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

[Main Page](#)

[Classes](#)

[Files](#)

[Examples](#)

Examples

Here is a list of all examples:

- [1.Basics/Freewheeling/Freewheeling.ino](#)
- [1.Basics/PowerOutputs/PowerOutputs.ino](#)
- [1.Basics/Simplest/Simplest.ino](#)
- [1.Basics/Sweep/Sweep.ino](#)
- [1.Basics/TankStyleSweep/TankStyleSweep.ino](#)
- [2.Settings/Ramping/Ramping.ino](#)
- [2.Settings/SerialTimeout/SerialTimeout.ino](#)
- [3.Advanced/Checksum/Checksum.ino](#)
- [3.Advanced/SharedLine/SharedLine.ino](#)
- [3.Advanced/SoftwareSerial/SoftwareSerial.ino](#)

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

[Main Page](#) [Classes](#) [Files](#) [Examples](#)

1.Basics/Freewheeling/Freewheeling.ino

Goes in one direction, lets the motor freewheel, and then goes in the other direction.

```
// Freewheeling Sample for USB Sabertooth Packet
// Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <USBSabertooth.h>

// Instead of braking, this sample lets the motor
// freewheel.

USBSabertoothSerial C; // Use the Arduino TX pin.
// It connects to S1.
// See the SoftwareSerial example in 3.Advanced
// for how to use other pins.

USBSabertooth ST(C, 128); // The USB Sabertooth is
// on address 128 (unless you've changed it with
// DEScribe).

// We'll name its object ST.
//
// If you've set up your Sabertooth on a different
// address, of course change
// that here. For how to configure the Sabertooth,
```

```
see the DIP Switch Wizard at
//
// http://www.dimensionengineering.com/datasheets/
// USBSabertoothDIPWizard/start.htm
// Be sure to select Packet Serial Mode for use
// with this library.

void setup()
{
    SabertoothTXPinSerial.begin(9600); // 9600 is the
        default baud rate for Sabertooth Packet Serial.
    // You can change this with the DEScribe software,
        available at
    // http://www.dimensionengineering.com/describe
}

void loop()
{
    ST.freewheel(1, false); // Turn off freewheeling.
    ST.motor(1, 2047);      // Go forward at full
        power.
    delay(1000);             // Wait 1 second.
    ST.freewheel(1, true);   // Turn on freewheeling.
    delay(2000);             // Wait 2 seconds.
    ST.motor(1, -2047);     // Reverse at full power.
    ST.freewheel(1, false);  // Turn off freewheeling.
    delay(1000);             // Wait 1 seconds.
    ST.freewheel(1, true);   // Turn on freewheeling.
    delay(2000);
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page Classes Files Examples

1.Basics/PowerOutputs/PowerOutputs.ino

Demonstrates the use of power outputs P1 and P2 as additional controllable outputs.

```
// Power Outputs Sample for USB Sabertooth Packet
// Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <USBSabertooth.h>

// This example treats the power outputs P1 and P2
// as controllable outputs,
// useful for fans, lights, single-direction
// motors, etc.

//
// The power outputs are not, by default,
// controllable outputs.

// You will need to use the DEScribe software,
// available at
// http://www.dimensionengineering.com/describe
// To configure them, in DEScribe,
// (1) Connect and Download Settings,
// (2) On the Power Outputs tab, set Mode to
// 'Controllable Output', and then
// (3) Upload Settings to Device
```

```
// This sample will then work.

USBSabertoothSerial C; // Use the Arduino TX pin.
    It connects to S1.
// See the SoftwareSerial example in 3.Advanced
    for how to use other pins.

USBSabertooth ST(C, 128); // The USB Sabertooth is
    on address 128 (unless you've changed it with
        DEScribe).
// We'll name its object ST.
//
// If you've set up your Sabertooth on a different
    address, of course change
// that here. For how to configure the Sabertooth,
    see the DIP Switch Wizard at
//
http://www.dimensionengineering.com/datasheets/
USBSabertoothDIPWizard/start.htm
// Be sure to select Packet Serial Mode for use
    with this library.

void setup()
{
    SabertoothTXPinSerial.begin(9600);
}

void loop()
{
    int value;

    // Ramp power output 1 from -2047 to 2047 (off to
        full power),
    // waiting 20 ms (1/50th of a second) per step.
    for (value = -2047; value <= 2047; value += 16)
    {
        ST.power(1, value);
        delay(20);
```

```
}

// Now go back the way we came.
for (value = 2047; value >= -2047; value -= 16)
{
    ST.power(1, value); // Tip: Typing
    ST.power(value) does the same thing as
    ST.power(1, value).
    delay(20);           //      If you often use
    only one power output, this alternative can
    save you typing.
}
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page Classes Files Examples

1.Basics/Simplest/Simplest.ino

Goes in one direction, stops, and then goes in the other direction.

```
// Simplest Sample for USB Sabertooth Packet Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <USBSabertooth.h>

USBSabertoothSerial c; // Use the Arduino TX pin.
// It connects to S1.
// See the SoftwareSerial example in 3.Advanced
// for how to use other pins.

USBSabertooth ST(c, 128); // The USB Sabertooth is
// on address 128 (unless you've changed it with
// DEScibe).

// We'll name its object ST.
//
// If you've set up your Sabertooth on a different
// address, of course change
// that here. For how to configure the Sabertooth,
// see the DIP Switch Wizard at
//
// http://www.dimensionengineering.com/datasheets/USBSabertoothDIPWizard/start.htm
```

```
// Be sure to select Packet Serial Mode for use
// with this library.
//
// The USBSabertooth library exposes features that
// only exist on USB-enabled Sabertooth motor
// drivers, such as
// 12-bit motor outputs, power outputs, control
// over freewheeling, motor current read-back, and
// User Mode variables.
// If you do not need these features, and want
// your code to be compatible with all
// Sabertooth/SyRen motor drivers,
// including those that are not USB-enabled, use
// the Sabertooth library instead.

void setup()
{
    SabertoothTXPinSerial.begin(9600); // 9600 is the
        default baud rate for Sabertooth Packet Serial.
    // You can change this with the DEScribe software,
        available at
    //      http://www.dimensionengineering.com/describe
}

void loop()
{
    ST.motor(1, 2047); // Go forward at full power.
    delay(2000); // Wait 2 seconds.
    ST.motor(1, 0); // Stop.
    delay(2000); // Wait 2 seconds.
    ST.motor(1, -2047); // Reverse at full power.
    delay(2000); // Wait 2 seconds.
    ST.motor(1, 0); // Stop.
    delay(2000);
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page Classes Files Examples

1.Basics/Sweep/Sweep.ino

Sweeps from full reverse to full forward and then from full forward to full reverse.

```
// Sweep Sample for USB Sabertooth Packet Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <USBSabertooth.h>

USBSabertoothSerial C; // Use the Arduino TX pin.
// It connects to S1.
// See the SoftwareSerial example in 3.Advanced
// for how to use other pins.

USBSabertooth ST(C, 128); // The USB Sabertooth is
// on address 128 (unless you've changed it with
// DEScibe).

// We'll name its object ST.
//
// If you've set up your Sabertooth on a different
// address, of course change
// that here. For how to configure the Sabertooth,
// see the DIP Switch Wizard at
//
// http://www.dimensionengineering.com/datasheets/USBSabertoothDIPWizard/start.htm
```

```
// Be sure to select Packet Serial Mode for use
// with this library.
//
// The USBSabertooth library exposes features that
// only exist on USB-enabled Sabertooth motor
// drivers, such as
// 12-bit motor outputs, power outputs, control
// over freewheeling, motor current read-back, and
// User Mode variables.
// If you do not need these features, and want
// your code to be compatible with all
// Sabertooth/SyRen motor drivers,
// including those that are not USB-enabled, use
// the Sabertooth library instead.

void setup()
{
    SabertoothTXPinSerial.begin(9600); // 9600 is the
    // default baud rate for Sabertooth Packet Serial.
    // You can change this with the DEScribe software,
    // available at
    // http://www.dimensionengineering.com/describe
}

void loop()
{
    int power;

    // Ramp motor 1 from -2047 to 2047 (full reverse
    // to full forward),
    // waiting 20 ms (1/50th of a second) per step.
    for (power = -2047; power <= 2047; power += 16)
    {
        ST.motor(1, power);
        delay(20);
    }
}
```

```
// Now go back the way we came.  
for (power = 2047; power >= -2047; power -= 16)  
{  
    ST.motor(1, power); // Tip: Typing  
    ST.motor(power) does the same thing as  
    ST.motor(1, power).  
    delay(20);           // If you often use  
    only one motor, this alternative can save you  
    typing.  
}  
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page Classes Files Examples

1.Basics/TankStyleSweep/TankStyleSweep.ino

Sweeps various ranges in mixed (rover) mode.

```
// Tank-Style (Diff-Drive) Sweep Sample for USB
// Sabertooth Packet Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <USBSabertooth.h>

USBSabertoothSerial C; // Use the Arduino TX pin.
// It connects to S1.
// See the SoftwareSerial example in 3.Advanced
// for how to use other pins.

USBSabertooth ST(C, 128); // The USB Sabertooth is
// on address 128 (unless you've changed it with
// DEScribe).

// We'll name its object ST.
//
// If you've set up your Sabertooth on a different
// address, of course change
// that here. For how to configure the Sabertooth,
// see the DIP Switch Wizard at
//
// http://www.dimensionengineering.com/datasheets/USBSabertoothDIPWizard/start.htm
```

```
// Be sure to select Packet Serial Mode for use
// with this library.
//
// The USBSabertooth library exposes features that
// only exist on USB-enabled Sabertooth motor
// drivers, such as
// 12-bit motor outputs, power outputs, control
// over freewheeling, motor current read-back, and
// User Mode variables.
// If you do not need these features, and want
// your code to be compatible with all
// Sabertooth/SyRen motor drivers,
// including those that are not USB-enabled, use
// the Sabertooth library instead.

void setup()
{
    SabertoothTXPinSerial.begin(9600); // 9600 is the
        default baud rate for Sabertooth Packet Serial.
    // You can change this with the DEScribe software,
        available at
    //      http://www.dimensionengineering.com/describe

    ST.drive(0); // The Sabertooth won't act on mixed
        mode packet serial commands until
    ST.turn(0); // it has received power levels for
        BOTH throttle and turning, since it
    // mixes the two together to get diff-drive power
        levels for both motors.
}

// The SLOW ramp here is turning, and the FAST ramp
// is throttle.
// If that's the opposite of what you're seeing,
// swap M2A and M2B.
void loop()
{
```

```
int power;

// Don't turn. Ramp from going backwards to going
// forwards, waiting 20 ms (1/50th of a second)
// per step of 16.
for (power = -2047; power <= 2047; power += 16)
{
    ST.drive(power);
    delay(20);
}

// Now, let's use a power level of 400 (out of
// 2047) forward.
// This way, our turning will have a radius.
ST.drive(400);

// Ramp turning from full left to full right
// SLOWLY by waiting 20 ms (1/50th of a second)
// per step of 4.
for (power = -2047; power <= 2047; power += 4)
{
    ST.turn(power);
    delay(20);
}

// Now stop turning, and stop driving.
ST.turn(0);
ST.drive(0);

// Wait a bit. This is so you can catch your robot
// if you want to. :-)
delay(5000);
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

[Main Page](#) [Classes](#) [Files](#) [Examples](#)

2.Settings/Ramping/Ramping.ino

Modifies the ramp time.

```
// Set Ramping Sample for USB Sabertooth Packet
// Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <USBSabertooth.h>

USBSabertoothSerial C;
USBSabertooth      ST(C, 128);

void setup()
{
    SabertoothTXPinSerial.begin(9600);

    // Ramping values run from -16383 (fast) to 2047
    // (slow).
    // -16383 is equivalent to turning off ramping.
    ST.setRamping(1980); // (approximately 2 seconds)
}

void loop()
{
    // Full forward, both motors.
    ST.motor(1, 2047);
```

```
ST.motor(2, 2047);
delay(5000);

// Full reverse
ST.motor(1, -2047);
ST.motor(2, -2047);
delay(5000);
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page Classes Files Examples

2.Settings/SerialTimeout/SerialTimeout.ino

Sets a serial timeout, and then delays to demonstrate its stopping behavior.

```
// Set Serial Timeout Sample for USB Sabertooth
// Packet Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <USBSabertooth.h>

USBSabertoothSerial C;
USBSabertooth      ST(C, 128);

void setup()
{
    SabertoothTXPinSerial.begin(9600);

    // Set a timeout of 1500 ms here.
    // A value of 0 resets the serial timeout to its
    // default (normally, disabled).
    // You can also set the serial timeout in
    // DEScribe, available at
    // http://www.dimensionengineering.com/describe
    ST.setTimeout(1500);
}
```

```
void loop()
{
    // Set motor 1 to reverse 400 (out of 2047), and
    // sleep for 5 seconds.
    // Notice how it cuts out after 1.5 seconds --
    // this is the serial timeout in action.
    // Since we configured it in setup() for 1.5
    // second, 1.5 second without any new
    // commands will cause the motors to stop.
    ST.motor(1, -400);
    delay(5000);

    // Why do this?
    // If your program crashes, or the signal wire is
    // not working properly,
    // the Sabertooth will stop receiving commands
    // from the Arduino.
    // With a timeout, your robot will stop.
    //
    // So, serial timeout is primarily a safety
    // feature. That being the case,
    // it's best to set the serial timeout in DEScribe
    // if you can -- if the
    // signal line is noisy when the command is sent,
    // it may be lost. DEScribe
    // settings are saved on the motor driver,
    // eliminating that possibility.
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

[Main Page](#) [Classes](#) [Files](#) [Examples](#)

3. Advanced/Checksum/Checksum.ino

Changes from CRC to Checksum mode.

```
// Checksum Sample for USB Sabertooth Packet Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <USBSabertooth.h>

// This sample changes the type of error detection
// that is done.
// It uses checksums to achieve a faster update
// rate than the Sweep sample.

//
// The tradeoffs are as follows:
// |-----|-----|
// | CRC | | Checksum |
// |-----|-----|
// | Command Size | | 8 bytes |
// | 10 bytes | |
// | Max Command Rate at 9600 Baud | | 120 cmd/s |
// | 96 cmd/s | |
// | Detectable Bit Flips (Worst-Case) | | 1 (HD=2) |
// | 5 (HD=6) | |
// |-----|-----|
```

```
- | ----- |
//  
// If you want to, you can require the use of CRC-  
// protected commands  
// with DEScribe. Go to DEScribe's Serial tab to  
// find this option.  
//  
// This tab also lets you change the serial baud  
// rate. Increasing the  
// baud rate is, in most situations, a better way  
// to increase the max  
// command rate than weakening error detection.  
  
USBSabertoothSerial C;  
USBSabertooth ST(C, 128);  
  
void setup()  
{  
    SabertoothTXPinSerial.begin(9600);  
  
    ST.useChecksum(); // ST.useCRC() is the default.  
}  
  
void loop()  
{  
    int power;  
  
    // Ramp motor 1 from -2047 to 2047 (full reverse  
    // to full forward),  
    // waiting 9 ms (1/111th of a second) per step.  
    for (power = -2047; power <= 2047; power += 8)  
    {  
        ST.motor(1, power);  
        delay(9);  
    }  
  
    // Now go back the way we came.  
    for (power = 2047; power >= -2047; power -= 8)
```

```
{  
    ST.motor(1, power);  
    delay(9);  
}  
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page Classes Files Examples

3.Advanced/SharedLine/SharedLine.ino

Communicates with three Sabertooth motor drivers using a shared TX/S1 wire.

```
// Shared Line Sample for USB Sabertooth Packet
// Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <USBSabertooth.h>

// Up to 8 Sabertooth/SyRen motor drivers can share
// the same S1 line.
// This sample uses three: address 128 and 129 on
// ST1[0] and ST1[2],
// and address 130 on ST2.
//
// To change the address of a USB Sabertooth motor
// driver, go to the
// Serial tab in DEScribe. DEScribe can be
// downloaded from
// http://www.dimensionengineering.com/describe
USBSabertoothSerial C;
USBSabertooth      ST1[2] = { USBSabertooth(C,
    128), USBSabertooth(C, 129) };
USBSabertooth      ST2(C, 130);
```

```
void setup()
{
    SabertoothTXPinSerial.begin(9600);
}

void loop()
{
    // ST1[0] (address 128) has power 800 (of 2047
        max) on M1,
    // ST1[1] (address 129) has power 1000 (of 2047
        max) on M2, and
    // ST2      (address 130) we'll do tank-style and
        have it drive 300 and turn right 800.
    // Do this for 5 seconds.
    ST1[0].motor(1, 800);
    ST1[1].motor(2, 1000);
    ST2.drive(300);
    ST2.turn(800);
    delay(5000);

    // And now let's stop for 5 seconds, except
        address 130 -- we'll let it stop and turn
        left...
    ST1[0].motor(1, 0);
    ST1[1].motor(2, 0);
    ST2.drive(0);
    ST2.turn(-600);
    delay(5000);
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

[Main Page](#) [Classes](#) [Files](#) [Examples](#)

3. Advanced/SoftwareSerial/SoftwareSerial.ino

Uses a pin other than TX to connect to S1.

```
// Software Serial Sample for USB Sabertooth Packet
// Serial
// Copyright (c) 2012-2013 Dimension Engineering
// LLC
// See license.txt for license details.

#include <SoftwareSerial.h>
#include <USBSabertooth.h>

SoftwareSerial      SWSerial(NOT_A_PIN, 11); // RX
on no pin (unused), TX on pin 11 (to S1).
USBSabertoothSerial C(SWSerial);           // Use
SWSerial as the serial port.
USBSabertooth       ST(C, 128);           // Use
address 128.

void setup()
{
  SWSerial.begin(9600);
}

void loop()
{
  int power;
```

```
// Ramp motor 1 from -2047 to 2047 (full reverse  
// to full forward),  
// waiting 20 ms (1/50th of a second) per step.  
for (power = -2047; power <= 2047; power += 16)  
{  
    ST.motor(1, power);  
    delay(20);  
}  
  
// Now go back the way we came.  
for (power = 2047; power >= -2047; power -= 16)  
{  
    ST.motor(1, power);  
    delay(20);  
}  
}
```

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples
Class List	Class Index	Class Members	
<h2>USBSabertooth Member List</h2>			

This is the complete list of members for **USBSabertooth**, including all inherited members.

address() const
command(byte command, byte value)
command(byte command, const byte *value, size_t bytes)
drive(int value)
freewheel(int value=2048)
freewheel(byte motorOutputNumber, int value=2048)
get(byte type, byte number)
getBattery(byte motorOutputNumber, boolean unscaled=false)
getCurrent(byte motorOutputNumber, boolean unscaled=false)
getGetRetryInterval() const
getGetTimeout() const
getTemperature(byte motorOutputNumber, boolean unscaled=false)
keepAlive()
motor(int value)
motor(byte motorOutputNumber, int value)
power(int value)
power(byte powerOutputNumber, int value)

set(byte type, byte number, int value)
setGetRetryInterval(int32_t intervalMS)
setGetTimeout(int32_t timeoutMS)
setRamping(int value)
setRamping(byte motorOutputNumber, int value)
setTimeout(int milliseconds)
shutDown(byte type, byte number, boolean value=true)
turn(int value)
USBSabertooth(USBSabertoothSerial &serial, byte address)
useChecksum()
useCRC()
usingCRC() const

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples
Class List	Class Index	Class Members	

USBSabertoothSerial Member List

This is the complete list of members for **USBSabertoothSerial**, including all inherited members.

port()

USBSabertoothSerial(Stream &port=SabertoothTXPinSerial)

USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

Main Page	Classes	Files	Examples
File List			
USBSabertooth			

USBSabertooth.h

Go to the documentation of this file.

```
1  /*
2  Arduino Library for USB Sabertooth Packet
3  Serial
4  Copyright (c) 2013 Dimension Engineering LLC
5  http://www.dimensionengineering.com/arduino
6  Permission to use, copy, modify, and/or
7  distribute this software for any
8  purpose with or without fee is hereby
9  granted, provided that the above
10 copyright notice and this permission notice
11 appear in all copies.
12
13 THE SOFTWARE IS PROVIDED "AS IS" AND THE
14 AUTHOR DISCLAIMS ALL WARRANTIES
15 WITH REGARD TO THIS SOFTWARE INCLUDING ALL
16 IMPLIED WARRANTIES OF
17 MERCHANTABILITY AND FITNESS. IN NO EVENT
18 SHALL THE AUTHOR BE LIABLE FOR ANY
19 SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
20 DAMAGES OR ANY DAMAGES WHATSOEVER
21 RESULTING FROM LOSS OF USE, DATA OR PROFITS,
```

```
    WHETHER IN AN ACTION OF CONTRACT,  
15| NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING  
    OUT OF OR IN CONNECTION WITH THE  
16| USE OR PERFORMANCE OF THIS SOFTWARE.  
17| */  
18|  
19| #ifndef USBSabertooth_h  
20| #define USBSabertooth_h  
21|  
27| #if defined(ARDUINO) && ARDUINO < 100  
28| #error "This library requires Arduino 1.0 or  
    newer."  
29| #endif  
30|  
31| #include <Arduino.h>  
32|  
33| #if defined(USBCON)  
34| #define SabertoothTXPinSerial Serial1 //  
    Arduino Leonardo has TX->1 on Serial1, not  
    Serial.  
35| #else  
36| #define SabertoothTXPinSerial Serial  
37| #endif  
38| #define SyRenTXPinSerial  
    SabertoothTXPinSerial  
39|  
40| #define SABERTOOTH_COMMAND_MAX_BUFFER_LENGTH  
    10  
41| #define SABERTOOTH_COMMAND_MAX_DATA_LENGTH  
    5  
42| #define SABERTOOTH_DEFAULT_GET_RETRY_INTERVAL  
    100  
43| #define SABERTOOTH_DEFAULT_GET_TIMEOUT  
    SABERTOOTH_INFINITE_TIMEOUT  
44| #define SABERTOOTH_GET_TIMED_OUT  
    -32768  
45| #define SABERTOOTH_INFINITE_TIMEOUT
```

```
-1
46 #define SABERTOOTH_MAX_VALUE
16383
47
48 enum USBSabertoothCommand
49 {
50     SABERTOOTH_CMD_SET = 40,
51     SABERTOOTH_CMD_GET = 41,
52 };
53
54 enum USBSabertoothReplyCode
55 {
56     SABERTOOTH_RC_GET = 73
57 };
58
59 enum USBSabertoothGetType
60 {
61     SABERTOOTH_GET_VALUE      = 0x00,
62     SABERTOOTH_GET_BATTERY    = 0x10,
63     SABERTOOTH_GET_CURRENT    = 0x20,
64     SABERTOOTH_GET_TEMPERATURE = 0x40
65 };
66
67 enum USBSabertoothSetType
68 {
69     SABERTOOTH_SET_VALUE      = 0x00,
70     SABERTOOTH_SET_KEEPALIVE  = 0x10,
71     SABERTOOTH_SET_SHUTDOWN   = 0x20,
72     SABERTOOTH_SET_TIMEOUT    = 0x40
73 };
74
75 class USBSabertoothCommandWriter
76 {
77 public:
78     static size_t writeToBuffer(byte* buffer,
        byte address, USBSabertoothCommand command,
        boolean useCRC, const byte* data, size_t
```

```
    lengthOfData);
79| static void    writeToStream(Stream& port,
   byte address, USBSabertoothCommand command,
   boolean useCRC, const byte* data, size_t
   lengthOfData);
80| };
81|
82| class USBSabertoothChecksum
83| {
84| public:
85|     static byte value(const byte* data, size_t
   lengthOfData);
86| };
87|
88| class USBSabertoothCRC7
89| {
90| public:
91|     void begin();
92|     void write(byte data);
93|     void write(const byte* data, size_t
   lengthOfData);
94|     void end  ();
95|
96| public:
97|     inline byte value() const { return _crc;
   }
98|     void value(byte crc) { _crc = crc;
   }
99|
100|    static byte value(const byte* data, size_t
   lengthOfData);
101|
102| private:
103|     byte _crc;
104| };
105|
106| class USBSabertoothCRC14
```

```
107 {
108 public:
109     void begin();
110     void write(byte data);
111     void write(const byte* data, size_t
lengthOfData);
112     void end();
113
114 public:
115     inline uint16_t value() const
116     {
117         return _crc;
118     }
119
120     void value(uint16_t crc) { _crc
= crc; }
121
122
123
124 class USBSabertoothReplyReceiver
125 {
126 public:
127     USBSabertoothReplyReceiver();
128
129 public:
130     inline byte address
131     () const { return _data[0];
132     }
133     inline USBSabertoothReplyCode command
134     () const { return
(USBSabertoothReplyCode)_data[1]; }
135     inline const byte*
136     () const { return _data;
137     }
138     inline boolean usingCRC()
```

```
    const { return _usingCRC;
}
134 public:
135     inline boolean ready() const { return
136         _ready; }
137     void      read (byte data);
138     void      reset();
139
140 private:
141     byte
142     _data[SABERTOOTH_COMMAND_MAX_BUFFER_LENGTH];
143     size_t _length; boolean _ready, _usingCRC;
144 };
145 class USBSabertoothTimeout
146 {
147 public:
148     USBSabertoothTimeout(int32_t timeoutMS);
149
150 public:
151     boolean canExpire() const;
152
153     boolean expired() const;
154
155     void expire();
156
157     void reset();
158
159 private:
160     uint32_t _start;
161     int32_t  _timeoutMS;
162 };
163
169 class USBSabertoothSerial
170 {
171     friend class USBSabertooth;
```

```
172 |
173 |     public:
179 |         USBSabertoothSerial(Stream& port =
180 |             SabertoothTXPinSerial);
180 |
181 |     public:
186 |         inline Stream& port() { return _port; }
187 |
188 |     private:
189 |         boolean tryReceivePacket();
190 |
191 |     private:
192 |         USBSabertoothSerial(USBSabertoothSerial&
193 |             serial); // no copy
193 |         void operator = (USBSabertoothSerial&
194 |             serial);
194 |
195 |     private:
196 |         USBSabertoothReplyReceiver _receiver;
197 |         Stream&
198 |             _port;
198 |
199 |
204 | class USBSabertooth
205 | {
206 |     public:
213 |         USBSabertooth(USBSabertoothSerial& serial,
214 |             byte address);
214 |
215 |     public:
220 |         inline byte address() const { return
221 |             _address; }
221 |
227 |         void command(byte command, byte value);
228 |
235 |         void command(byte command, const byte*
236 |             value, size_t bytes);
```

```
237 public:  
243     void motor(int value);  
244  
253     void motor(byte motorOutputNumber, int  
      value);  
254  
260     void power(int value);  
261  
270     void power(byte powerOutputNumber, int  
      value);  
271  
277     void drive(int value);  
278  
284     void turn(int value);  
285  
292     void freewheel(int value = 2048);  
293  
303     void freewheel(byte motorOutputNumber, int  
      value = 2048);  
304  
316     void shutDown(byte type, byte number,  
      boolean value = true);  
317  
318 public:  
332     void set(byte type, byte number, int  
      value);  
333  
339     void setRamping(int value);  
340  
349     void setRamping(byte motorOutputNumber, int  
      value);  
350  
359     void setTimeout(int milliseconds);  
360  
366     void keepAlive();  
367  
368 public:
```

```
381     inline int get(byte type, byte number)
382     {
383         return get(type, number,
384             SABERTOOTH_GET_VALUE, false);
385     }
386
387     inline int getBattery(byte
388         motorOutputNumber, boolean unscaled = false)
389     {
390         return get('M', motorOutputNumber,
391             SABERTOOTH_GET_BATTERY, unscaled);
392     }
393
394     inline int getCurrent(byte
395         motorOutputNumber, boolean unscaled = false)
396     {
397         return get('M', motorOutputNumber,
398             SABERTOOTH_GET_CURRENT, unscaled);
399     }
400
401     inline int getTemperature(byte
402         motorOutputNumber, boolean unscaled = false)
403     {
404         return get('M', motorOutputNumber,
405             SABERTOOTH_GET_TEMPERATURE, unscaled);
406     }
407
408     public:
409         inline int32_t getGetRetryInterval() const
410         { return _getRetryInterval; }
411
412         inline void setGetRetryInterval(int32_t
413             intervalMS) { _getRetryInterval = intervalMS; }
414
415         inline int32_t getGetTimeout() const {
416             return _getTimeout; }
417
418
```

```
447     inline void setGetTimeout(int32_t
448         timeoutMS) { _getTimeout = timeoutMS; }
449
450     inline boolean usingCRC() const { return
451         _crc; }
452
453     inline void useChecksum() { _crc = false; }
454
455     inline void useCRC() { _crc = true ; }
456
457 private:
458     int get(byte type, byte number,
459             USBSabertoothGetType getType,
460             boolean raw);
461
462     void set(byte type, byte number, int value,
463             USBSabertoothSetType setType);
464
465 private:
466     void init();
467
468 private:
469     const byte           _address;
470     boolean              _crc;
471     int32_t               _getRetryInterval;
472     int32_t               _getTimeout;
473     USBSabertoothSerial& _serial;
474 };
475
476 #endif
```