USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

### Class List

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

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USBSabertooth</td>
<td>Controls a USB Sabertooth motor driver running in Packet Serial mode</td>
</tr>
<tr>
<td>USBSabertoothSerial</td>
<td>Create a USBSabertoothSerial for the serial port you are using, and then attach a USBSabertooth for each motor driver you want to communicate with</td>
</tr>
</tbody>
</table>
USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

**USBSabertooth Class Reference**

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>USBSabertooth</td>
<td>USBSabertoothSerial (USBSabertoothSerial, byte address)</td>
</tr>
<tr>
<td>address () const</td>
<td>byte address () const</td>
</tr>
<tr>
<td>command</td>
<td>void command (byte command, byte value)</td>
</tr>
<tr>
<td>command</td>
<td>void command (byte command, const byte *value, size_t bytes)</td>
</tr>
<tr>
<td>motor</td>
<td>void motor (int value)</td>
</tr>
<tr>
<td>motor</td>
<td>void motor (byte motorOutputNumber, int value)</td>
</tr>
<tr>
<td>power</td>
<td>void power (int value)</td>
</tr>
<tr>
<td>power</td>
<td>void power (byte powerOutputNumber, int value)</td>
</tr>
<tr>
<td>drive</td>
<td>void drive (int value)</td>
</tr>
<tr>
<td>turn</td>
<td>void turn (int value)</td>
</tr>
<tr>
<td>freewheel</td>
<td>void freewheel (int value=2048)</td>
</tr>
<tr>
<td>freewheel</td>
<td>void freewheel (byte motorOutputNumber, int value=2048)</td>
</tr>
<tr>
<td>shutDown</td>
<td>void shutDown (byte type, byte number, boolean value=true)</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>void useChecksum ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>void useCRC ()</td>
</tr>
</tbody>
</table>
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

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.

```cpp
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.

```cpp
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.

```cpp
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.

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

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

Gets the battery voltage.

**Parameters**

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

- **unscaled** If true, gets in unscaled units. If false, 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 or 2.
  You can also use a character, such as '3', to select by Plain Text Serial address.

- **unscaled** If true, gets in unscaled units. If false, gets in scaled units.
Returns
The value, or SABERTOOTH_GET_TIMED_OUT.

```cpp
int32_t USBSabertooth::getGetRetryInterval ( ) const inline
```

Gets the get retry interval.

Returns
The get retry interval, in milliseconds.

```cpp
int32_t USBSabertooth::getGetTimeout ( ) const inline
```

Gets the get timeout.

Returns
The get timeout, in milliseconds.

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

Gets the motor output temperature.

Parameters

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

- **unscaled** If true, gets in unscaled units. If false
Returns
The value, or SABERTOOTH_GET_TIMED_OUT.

```cpp
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.

```cpp
void USBSabertooth::motor ( int value )
```

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

**Parameters**

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

**Examples:**


```cpp
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. The value, between -2047 and 2047.

```cpp
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.

```cpp
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.

```cpp
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).

```cpp
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
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.

```cpp
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.

```cpp
void USBSabertooth::useChecksum ( )
```

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

```cpp
void USBSabertooth::useCRC ( )
```

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

```cpp
boolean USBSabertooth::usingCRC ( ) const
```

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.

## USBSabertoothSerial Class Reference

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USBSabertoothSerial</td>
<td>Stream &amp; port = SabertoothTXPinSerial</td>
</tr>
<tr>
<td>Stream &amp; port ()</td>
<td></td>
</tr>
</tbody>
</table>
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`,
2. `Basics/PowerOutputs/PowerOutputs.ino`,
3. `Basics/Simplest/Simplest.ino`,
4. `Basics/Sweep/Sweep.ino`,
5. `Basics/TankStyleSweep/TankStyleSweep.ino`,
6. `Settings/Ramping/Ramping.ino`,
7. `Settings/SerialTimeout/SerialTimeout.ino`,
8. `Advanced/Checksum/Checksum.ino`,
9. `Advanced/SharedLine/SharedLine.ino`, and
10. `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 ( )

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.

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Classes</th>
<th>Files</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Members</td>
<td></td>
</tr>
</tbody>
</table>

**Class Index**

<table>
<thead>
<tr>
<th>U</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>U</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>USBSabertoothSerial</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>USBSabertooth</th>
<th>U</th>
</tr>
</thead>
</table>
USB Sabertooth Packet Serial Library for Arduino

Control your USB-enabled Sabertooth with reliable Packet Serial.

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.

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Classes</th>
<th>Files</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class List</td>
<td>Class Index</td>
<td>Class Members</td>
<td>Functions</td>
</tr>
<tr>
<td>All</td>
<td>a c d f g k m p s t u</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **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.

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Classes</th>
<th>Files</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File List</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## 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.

<table>
<thead>
<tr>
<th>Main Page</th>
<th>Classes</th>
<th>Files</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>USBSabertooth</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**USBSabertooth Directory Reference**
### Files

<table>
<thead>
<tr>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>USBSabertooth.cpp</td>
</tr>
<tr>
<td>USBSabertooth.h [code]</td>
</tr>
<tr>
<td>USBSabertoothChecksum.cpp</td>
</tr>
<tr>
<td>USBSabertoothCommandWriter.cpp</td>
</tr>
<tr>
<td>USBSabertoothCRC14.cpp</td>
</tr>
<tr>
<td>USBSabertoothCRC7.cpp</td>
</tr>
<tr>
<td>USBSabertoothReplyReceiver.cpp</td>
</tr>
<tr>
<td>USBSabertoothSerial.cpp</td>
</tr>
<tr>
<td>USBSabertoothTimeout.cpp</td>
</tr>
</tbody>
</table>
USB Sabertooth Packet Serial Library for Arduino
Control your USB-enabled Sabertooth with reliable Packet Serial.

USBSabertooth.h File Reference

Go to the source code of this file.
## Classes

<table>
<thead>
<tr>
<th>class</th>
<th>USBSabertoothSerial</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create a <strong>USBSabertoothSerial</strong> for the serial port you are using, and then attach a <strong>USBSabertooth</strong> for each motor driver you want to communicate with. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class</th>
<th>USBSabertooth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Controls a USB Sabertooth motor driver running in Packet Serial mode. <a href="#">More...</a></td>
</tr>
</tbody>
</table>
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.

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.

1. Basics/Freewheeling/Freewheeling.ino

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

```cpp
#include <USBSabertooth.h>

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

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.

1. Basics/PowerOutputs/PowerOutputs.ino

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

```cpp
// 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.

```c
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.

1. Basics/Simplest/Simplest.ino

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

```cpp
// 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 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
}

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.

### 1. Basics/Sweep/Sweep.ino

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

```cpp
// 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.
    // 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).
    // 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
}

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.

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
```
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.

  }
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.

2. Settings/Ramping/Ramping.ino

Modifies the ramp time.

```cpp
// 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.

### 2. Settings/SerialTimeout/SerialTimeout.ino

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

```cpp
// 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.

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
// |-----|-----------|
// | 8 bytes | 10 bytes |
// | 120 cmd/s | 96 cmd/s |
// | 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.

```cpp
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.

3. Advanced/SharedLine/SharedLine.ino

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

```c
#include <USBSabertooth.h>

// 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);
```
```c
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.

3. Advanced/SoftwareSerial/SoftwareSerial.ino

Uses a pin other than TX to connect to S1.

```cpp
// 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.

### USBSabertooth Member List

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

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>address()</code> const</td>
</tr>
<tr>
<td><code>command(byte command, byte value)</code></td>
</tr>
<tr>
<td><code>command(byte command, const byte *value, size_t bytes)</code></td>
</tr>
<tr>
<td><code>drive(int value)</code></td>
</tr>
<tr>
<td><code>freewheel(int value=2048)</code></td>
</tr>
<tr>
<td><code>freewheel(byte motorOutputNumber, int value=2048)</code></td>
</tr>
<tr>
<td><code>get(byte type, byte number)</code></td>
</tr>
<tr>
<td><code>getBattery(byte motorOutputNumber, boolean unscaled=false)</code></td>
</tr>
<tr>
<td><code>getCurrent(byte motorOutputNumber, boolean unscaled=false)</code></td>
</tr>
<tr>
<td><code>getGetRetryInterval()</code> const</td>
</tr>
<tr>
<td><code>getGetTimeout()</code> const</td>
</tr>
<tr>
<td><code>getTemperature(byte motorOutputNumber, boolean unscaled=false)</code></td>
</tr>
<tr>
<td><code>keepAlive()</code></td>
</tr>
<tr>
<td><code>motor(int value)</code></td>
</tr>
<tr>
<td><code>motor(byte motorOutputNumber, int value)</code></td>
</tr>
<tr>
<td><code>power(int value)</code></td>
</tr>
<tr>
<td><code>power(byte powerOutputNumber, int value)</code></td>
</tr>
</tbody>
</table>
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.

**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.

Go to the documentation of this file.

```cpp
/*
  Arduino Library for USB Sabertooth Packet Serial
  Copyright (c) 2013 Dimension Engineering LLC
  http://www.dimensionengineering.com/arduino

  Permission to use, copy, modify, and/or distribute this software for any
  purpose with or without fee is hereby granted, provided that the above
  copyright notice and this permission notice appear in all copies.

  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
  WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
  MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
  SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER
  RESULTING FROM LOSS OF USE, DATA OR PROFITS,
```
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

*/

#ifndef USBSabertooth_h
#define USBSabertooth_h

#if defined(ARDUINO) && ARDUINO < 100
#error "This library requires Arduino 1.0 or newer."
#endif

#include <Arduino.h>

#if defined(USBCON)
#define SabertoothTXPinSerial Serial1
Arduino Leonardo has TX->1 on Serial1, not Serial.
#else
#define SabertoothTXPinSerial Serial
#endif
#define SyRenTXPinSerial SabertoothTXPinSerial

#define SABERTOOTH_COMMAND_MAX_BUFFER_LENGTH 10
#define SABERTOOTH_COMMAND_MAX_DATA_LENGTH 5
#define SABERTOOTH_DEFAULT_GET_RETRY_INTERVAL 100
#define SABERTOOTH_DEFAULT_GET_TIMEOUT SABERTOOTH_INFINITE_TIMEOUT
#define SABERTOOTH_GET_TIMED_OUT -32768
#define SABERTOOTH_INFINITE_TIMEOUT
#define SABERTOOTH_MAX_VALUE
16383

enum USBSabertoothCommand
{
    SABERTOOTH_CMD_SET = 40,
    SABERTOOTH_CMD_GET = 41,
};

enum USBSabertoothReplyCode
{
    SABERTOOTH_RC_GET = 73
};

enum USBSabertoothGetType
{
    SABERTOOTH_GET_VALUE = 0x00,
    SABERTOOTH_GET_BATTERY = 0x10,
    SABERTOOTH_GET_CURRENT = 0x20,
    SABERTOOTH_GET_TEMPERATURE = 0x40
};

enum USBSabertoothSetType
{
    SABERTOOTH_SET_VALUE = 0x00,
    SABERTOOTH_SET_KEEPALIVE = 0x10,
    SABERTOOTH_SET_SHUTDOWN = 0x20,
    SABERTOOTH_SET_TIMEOUT = 0x40
};

class USBSabertoothCommandWriter
{
    public:
        static size_t writeToBuffer(byte* buffer, byte address, USBSabertoothCommand command, boolean useCRC, const byte* data, size_t
static void writeToStream(Stream& port, byte address, USBSabertoothCommand command, boolean useCRC, const byte* data, size_t lengthOfData);

class USBSabertoothChecksum
{
    static byte value(const byte* data, size_t lengthOfData);
};

class USBSabertoothCRC7
{
    public:
        void begin();
        void write(byte data);
        void write(const byte* data, size_t lengthOfData);
        void end();

    public:
        inline byte value() const { return _crc; }
        void value(byte crc) { _crc = crc; }

    static byte value(const byte* data, size_t lengthOfData);
};

private:
    byte _crc;
};

class USBSabertoothCRC14
```cpp
public:
    void begin();
    void write(byte data);
    void write(const byte* data, size_t lengthOfData);
    void end();

public:
    inline uint16_t value() const {
        return _crc;
    }
    void value(uint16_t crc) {
        _crc = crc;
    }
    static uint16_t value(const byte* data, size_t lengthOfData);

private:
    uint16_t _crc;
};

class USBSabertoothReplyReceiver
{
pub}
```
const { return _usingCRC;
}

public:
inline boolean ready() const { return _ready;
}

void read (byte data);
void reset();

private:
byte _data[SABERTOOTH_COMMAND_MAX BUFFER LENGTH];
size_t _length; boolean _ready, _usingCRC;

};

class USBSabertoothTimeout
{
public:
USBSabertoothTimeout(int32_t timeoutMS);

public:
boolean canExpire() const;
boolean expired() const;
void expire();
void reset();

private:
uint32_t _start;
int32_t _timeoutMS;
};

class USBSabertoothSerial
{
friend class USBSabertooth;

public:
    USBSabertoothSerial(Stream& port = SabertoothTXPinSerial);

public:
    inline Stream& port() { return _port; }

private:
    boolean tryReceivePacket();

private:
    USBSabertoothSerial(USBSabertoothSerial& serial); // no copy
    void operator = (USBSabertoothSerial& serial);

private:
    USBSabertoothReplyReceiver _receiver;
    Stream& _port;
};

class USBSabertooth
{
public:
    USBSabertooth(USBSabertoothSerial& serial, byte address);

public:
    inline byte address() const { return _address; }

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

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);

public:

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

void setRamping(int value);

void setRamping(byte motorOutputNumber, int value);

void setTimeout(int milliseconds);

void keepAlive();
inline int get(byte type, byte number) {
    return get(type, number, SABERTOOTH_GET_VALUE, false);
}

inline int getBattery(byte motorOutputNumber, boolean unscaled = false) {
    return get('M', motorOutputNumber, SABERTOOTH_GET_BATTERY, unscaled);
}

inline int getCurrent(byte motorOutputNumber, boolean unscaled = false) {
    return get('M', motorOutputNumber, SABERTOOTH_GET_CURRENT, unscaled);
}

inline int getTemperature(byte motorOutputNumber, boolean unscaled = false) {
    return get('M', motorOutputNumber, SABERTOOTH_GET_TEMPERATURE, unscaled);
}

public:
inline int32_t getGetRetryInterval() const {
    return _getRetryInterval;
}

inline void setGetRetryInterval(int32_t intervalMS) {
    _getRetryInterval = intervalMS;
}

inline int32_t getGetTimeout() const {
    return _getTimeout;
}
void setGetTimeout(int32_t timeoutMS) { _getTimeout = timeoutMS; }

inline boolean usingCRC() const { return _crc; }

inline void useChecksum() { _crc = false; }
inline void useCRC() { _crc = true; }

private:
int get(byte type, byte number, USBSabertoothGetType getType, boolean raw);

void set(byte type, byte number, int value, USBSabertoothSetType setType);

private:
void init();

private:
const byte _address;
boolean _crc;
int32_t _getRetryInterval;
int32_t _getTimeout;
USBSabertoothSerial& _serial;

};

#endif