SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

Class List

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

Sabertooth	Controls a Sabertooth or SyRen motor driver running in Packet Serial mode

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

Sabertooth Class Reference

Public Member Functions |
Static Public Member Functions |
List of all members

Controls a Sabertooth or SyRen motor driver running in Packet Serial mode. More...

Public Member Functions

	Sabertooth (byte **address**)
	Sabertooth (byte **address**, SabertoothStream &**port**)
byte	**address** () const
SabertoothStream &	**port** () const
void	**autobaud** (boolean dontWait=false) const
void	**command** (byte command, byte value) const
void	**motor** (int power) const
void	**motor** (byte motor, int power) const
void	**drive** (int power) const
void	**turn** (int power) const
void	**stop** () const
void	**setMinVoltage** (byte value) const
void	**setMaxVoltage** (byte value) const

void	**setBaudRate** (long baudRate) const
void	**setDeadband** (byte value) const
void	**setRamping** (byte value) const
void	**setTimeout** (int milliseconds) const

Static Public Member Functions

static void	**autobaud** (SabertoothStream &**port**, boolean dontWait=false)

Detailed Description

Controls a Sabertooth or SyRen motor driver running in Packet Serial mode.

Examples:

1.Basics/Jolty/Jolty.ino, **1.Basics/Sweep/Sweep.ino**, **1.Basics/TankStyleSweep/TankStyleSweep.ino**, **2.Settings/MinVoltage/MinVoltage.ino**, **2.Settings/Persistent/BaudRate/BaudRate.ino**, **2.Settings/Persistent/Deadband/Deadband.ino**, **2.Settings/Persistent/MaxVoltage/MaxVoltage.ino**, **2.Settings/Persistent/Ramping/Ramping.ino**, **2.Settings/SerialTimeout/SerialTimeout.ino**, **3.Advanced/SharedLine/SharedLine.ino**, and **3.Advanced/SoftwareSerial/SoftwareSerial.ino**.

Constructor & Destructor Documentation

Sabertooth::Sabertooth (byte address)

Initializes a new instance of the **Sabertooth** class. The driver address is set to the value given, and the Arduino TX serial port is used.

Parameters

address	The driver address.

Sabertooth::Sabertooth (byte address, SabertoothStream & port)

Initializes a new instance of the **Sabertooth** class. The driver address is set to the value given, and the specified serial port is used.

Parameters

address	The driver address.
port	The port to use.

Member Function Documentation

byte Sabertooth::address ( ) const

inline

Gets the driver address.

Returns

The driver address.

void Sabertooth::autobaud (boolean dontWait = `false`) const

Sends the autobaud character.

Parameters

dontWait If false, a delay is added to give the driver time to start up.

Examples:

3.Advanced/SharedLine/SharedLine.ino.

void Sabertooth::autobaud (SabertoothStream & port, boolean dontWait = `fa`)

Sends the autobaud character.

Parameters

port	The port to use.
dontWait	If false, a delay is added to give the driver time t up.

void Sabertooth::command (byte command, byte value) const

Sends a packet serial command to the motor driver.

Parameters

command	The number of the command.
value	The command's value.

void Sabertooth::drive (int power) const

Sets the driving power.

Parameters

power	The power, between -127 and 127.

void Sabertooth::motor (int power) const

Sets the power of motor 1.

Parameters

power	The power, between -127 and 127.

Examples:

3.Advanced/SharedLine/SharedLine.ino.

void Sabertooth::motor (byte motor, int power) const

Sets the power of the specified motor.

Parameters

motor The motor number, 1 or 2.

power The power, between -127 and 127.

SabertoothStream& Sabertooth::port ( ) const

inline

Gets the serial port.

Returns

The serial port.

void Sabertooth::setBaudRate (long baudRate) const

Sets the baud rate. Baud rate is stored in EEPROM, so changes persist between power cycles.

Parameters

baudRate The baud rate. This can be 2400, 9600, 19200, 38400, or on some drivers 115200.

void Sabertooth::setDeadband (byte value) const

Sets the deadband. Deadband is stored in EEPROM, so changes persist between power cycles.

Parameters

value The deadband value. Motor powers in the range [-deadband, deadband] will be considered in the deadband, and will not prevent the driver from entering nor cause the driver to leave an idle brake state. 0 resets to the default, which is 3.

void Sabertooth::setMaxVoltage (byte value) const

Sets the maximum voltage. Maximum voltage is stored in EEPROM, so changes persist between power cycles.

Parameters

value The voltage. The units of this value are driver-specific and are specified in the Packet Serial chapter of the driver's user manual.

void Sabertooth::setMinVoltage (byte value) const

Sets the minimum voltage.

Parameters

value The voltage. The units of this value are driver-specific and are specified in the Packet Serial chapter of the driver's user manual.

void Sabertooth::setRamping (byte value) const

Sets the ramping. Ramping is stored in EEPROM, so changes persist between power cycles.

Parameters

value The ramping value. Consult the user manual for possible values.

void Sabertooth::setTimeout (int milliseconds) const

Sets the serial timeout.

Parameters

milliseconds The maximum time in milliseconds between packets. If this time is exceeded, the driver will stop the motors. This value is rounded up to the nearest 100 milliseconds. This library assumes the command value is in units of 100 milliseconds. This is true for most drivers, but not all. Check the packet serial chapter of the driver's user manual to make sure.

void Sabertooth::stop ( ) const

Stops.

void Sabertooth::turn (int power) const

Sets the turning power.

Parameters

power The power, between -127 and 127.

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

Class Index

S

S

Sabertooth

S

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

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

- address() : **Sabertooth**
- autobaud() : **Sabertooth**
- command() : **Sabertooth**
- drive() : **Sabertooth**
- motor() : **Sabertooth**
- port() : **Sabertooth**
- Sabertooth() : **Sabertooth**
- setBaudRate() : **Sabertooth**
- setDeadband() : **Sabertooth**
- setMaxVoltage() : **Sabertooth**
- setMinVoltage() : **Sabertooth**
- setRamping() : **Sabertooth**
- setTimeout() : **Sabertooth**
- stop() : **Sabertooth**
- turn() : **Sabertooth**

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

- address() : **Sabertooth**
- autobaud() : **Sabertooth**
- command() : **Sabertooth**
- drive() : **Sabertooth**
- motor() : **Sabertooth**
- port() : **Sabertooth**
- Sabertooth() : **Sabertooth**
- setBaudRate() : **Sabertooth**
- setDeadband() : **Sabertooth**
- setMaxVoltage() : **Sabertooth**
- setMinVoltage() : **Sabertooth**
- setRamping() : **Sabertooth**
- setTimeout() : **Sabertooth**
- stop() : **Sabertooth**
- turn() : **Sabertooth**

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

File List

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

[detail level 1 2]

Sabertooth	
Sabertooth.h	

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

Sabertooth Directory Reference

Files

file	**Sabertooth.cpp**
file	**Sabertooth.h** [code]

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

Sabertooth.h

```
/*
Arduino Library for SyRen/Sabertooth Packet Serial
Copyright (c) 2012-2013 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,
```

```
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 Sabertooth_h
20 #define Sabertooth_h
21
22 #if defined(ARDUINO) && ARDUINO >= 100
23 #include <Arduino.h>
24 typedef Stream SabertoothStream;
25 #else
26 #include <WProgram.h>
27 typedef Print SabertoothStream;
28 #endif
29
30 #if defined(USBCON)
31 #define SabertoothTXPinSerial Serial1 // Arduino Leonardo has TX->1 on Serial1, not Serial.
32 #else
33 #define SabertoothTXPinSerial Serial
34 #endif
35 #define SyRenTXPinSerial SabertoothTXPinSerial
36
41 class Sabertooth
42 {
43 public:
49   Sabertooth(byte address);
50
57   Sabertooth(byte address, SabertoothStream& port);
58
59 public:
64   inline byte address() const { return _address; }
```

```
  65
  70    inline SabertoothStream& port() const {
    return _port; }
  71
  76    void autobaud(boolean dontWait = false)
    const;
  77
  83    static void autobaud(SabertoothStream&
    port, boolean dontWait = false);
  84
  90    void command(byte command, byte value)
    const;
  91
  92  public:
  97    void motor(int power) const;
  98
 104    void motor(byte motor, int power) const;
 105
 110    void drive(int power) const;
 111
 116    void turn(int power) const;
 117
 121    void stop() const;
 122
 123  public:
 128    void setMinVoltage(byte value) const;
 129
 135    void setMaxVoltage(byte value) const;
 136
 142    void setBaudRate(long baudRate) const;
 143
 152    void setDeadband(byte value) const;
 153
 159    void setRamping(byte value) const;
 160
 169    void setTimeout(int milliseconds) const;
 170
```

```
private:
  void throttleCommand(byte command, int power) const;

private:
  const byte        _address;
  SabertoothStream& _port;
};

#endif
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

Examples

Here is a list of all examples:

- 1.Basics/Jolty/Jolty.ino
- 1.Basics/Sweep/Sweep.ino
- 1.Basics/TankStyleSweep/TankStyleSweep.ino
- 2.Settings/MinVoltage/MinVoltage.ino
- 2.Settings/Persistent/BaudRate/BaudRate.ino
- 2.Settings/Persistent/Deadband/Deadband.ino
- 2.Settings/Persistent/MaxVoltage/MaxVoltage.ino
- 2.Settings/Persistent/Ramping/Ramping.ino
- 2.Settings/SerialTimeout/SerialTimeout.ino
- 3.Advanced/SharedLine/SharedLine.ino
- 3.Advanced/SoftwareSerial/SoftwareSerial.ino

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

1.Basics/Jolty/Jolty.ino

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

```
// Jolty Sample for Packet Serial
// Copyright (c) 2012 Dimension Engineering LLC
// See license.txt for license details.

#include <Sabertooth.h>

Sabertooth ST(128); // The Sabertooth is on address
    128. 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 address, etc.
    see the DIP Switch Wizard for
 //   Sabertooth -
    http://www.dimensionengineering.com/datasheets/
    SabertoothDIPWizard/start.htm
 //   SyRen      -
    http://www.dimensionengineering.com/datasheets/
    SyrenDIPWizard/start.htm
 // Be sure to select Packetized Serial Mode for
    use with this library.
 //
 // On that note, you can use this library for
    SyRen just as easily.
 // The diff-drive commands (drive, turn) do not
```

```
    work on a SyRen, of course, but it will respond
    correctly
 // if you command motor 1 to do something
    (ST.motor(1, ...)), just like a Sabertooth.
 //
 // In this sample, hardware serial TX connects to
    S1.
 // See the SoftwareSerial example in 3.Advanced
    for how to use other pins.

void setup()
{
  SabertoothTXPinSerial.begin(9600); // 9600 is the
    default baud rate for Sabertooth packet serial.
  ST.autobaud(); // Send the autobaud command to
    the Sabertooth controller(s).
 // NOTE: *Not all* Sabertooth controllers need
    this command.
 //       It doesn't hurt anything, but V2
    controllers use an
 //       EEPROM setting (changeable with the
    function setBaudRate) to set
 //       the baud rate instead of detecting with
    autobaud.
 //
 //       If you have a 2x12, 2x25 V2, 2x60 or
    SyRen 50, you can remove
 //       the autobaud line and save yourself two
    seconds of startup delay.
}

void loop()
{
  ST.motor(1, 127);  // Go forward at full power.
  delay(2000);       // Wait 2 seconds.
  ST.motor(1, 0);    // Stop.
  delay(2000);       // Wait 2 seconds.
```

```
  ST.motor(1, -127); // Reverse at full power.
  delay(2000);       // Wait 2 seconds.
  ST.motor(1, 0);    // Stop.
  delay(2000);
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or 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.

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

#include <Sabertooth.h>

Sabertooth ST(128); // The Sabertooth is on address
    128. 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 address, etc.
    see the DIP Switch Wizard for
 //   Sabertooth -
    http://www.dimensionengineering.com/datasheets/
    SabertoothDIPWizard/start.htm
 //   SyRen      -
    http://www.dimensionengineering.com/datasheets/
    SyrenDIPWizard/start.htm
 // Be sure to select Packetized Serial Mode for
    use with this library.
 //
 // On that note, you can use this library for
    SyRen just as easily.
 // The diff-drive commands (drive, turn) do not
```

```
    work on a SyRen, of course, but it will respond
    correctly
 // if you command motor 1 to do something
    (ST.motor(1, ...)), just like a Sabertooth.
 //
 // In this sample, hardware serial TX connects to
    S1.
 // See the SoftwareSerial example in 3.Advanced
    for how to use other pins.

void setup()
{
  SabertoothTXPinSerial.begin(9600); // 9600 is the
    default baud rate for Sabertooth packet serial.
  ST.autobaud(); // Send the autobaud command to
    the Sabertooth controller(s).
 // NOTE: *Not all* Sabertooth controllers need
    this command.
 //       It doesn't hurt anything, but V2
    controllers use an
 //       EEPROM setting (changeable with the
    function setBaudRate) to set
 //       the baud rate instead of detecting with
    autobaud.
 //
 //       If you have a 2x12, 2x25 V2, 2x60 or
    SyRen 50, you can remove
 //       the autobaud line and save yourself two
    seconds of startup delay.
}

void loop()
{
 int power;

 // Ramp motor 1 from -127 to 127 (full reverse to
    full forward),
 // waiting 20 ms (1/50th of a second) per value.
```

```
 for (power = -127; power <= 127; power ++)
  {
    ST.motor(1, power);
    delay(20);
  }

 // Now go back the way we came.
 for (power = 127; power >= -127; power --)
  {
    ST.motor(1, power); // Tip for SyRen users:
     Typing ST.motor(power) does the same thing as
     ST.motor(1, power).
    delay(20);          //
     Since SyRen doesn't have a motor 2, this
     alternative can save you typing.
  }
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

1.Basics/TankStyleSweep/TankStyleSweep.ino

Sweeps various ranges in mixed (rover) mode.

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

#include <Sabertooth.h>

// Mixed mode is for tank-style diff-drive robots.

Sabertooth ST(128); // The Sabertooth is on address
    128. 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 address, etc.
    see the DIP Switch Wizard for
 //   Sabertooth -
    http://www.dimensionengineering.com/datasheets/
    SabertoothDIPWizard/start.htm
 //   SyRen      -
    http://www.dimensionengineering.com/datasheets/
    SyrenDIPWizard/start.htm
 // Be sure to select Packetized Serial Mode for
    use with this library.
 //
 // This sample uses the mixed mode (diff-drive)
    commands, which require two motors
 // and hence do not work on a SyRen.
```

```
 //
 // In this sample, hardware serial TX connects to
    S1.
 // See the SoftwareSerial example in 3.Advanced
    for how to use other pins.

void setup()
{
  SabertoothTXPinSerial.begin(9600); // 9600 is the
     default baud rate for Sabertooth packet serial.
  ST.autobaud(); // Send the autobaud command to
     the Sabertooth controller(s).
 // NOTE: *Not all* Sabertooth controllers need
    this command.
 //       It doesn't hurt anything, but V2
    controllers use an
 //       EEPROM setting (changeable with the
    function setBaudRate) to set
 //       the baud rate instead of detecting with
    autobaud.
 //
 //       If you have a 2x12, 2x25 V2, 2x60 or
    SyRen 50, you can remove
 //       the autobaud line and save yourself two
    seconds of startup delay.

  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 value.
 for (power = -127; power <= 127; power ++)
  {
    ST.drive(power);
    delay(20);
  }

 // Now, let's use a power level of 20 (out of 127)
    forward.
 // This way, our turning will have a radius.
  ST.drive(20);

 // Ramp turning from full left to full right
    SLOWLY by waiting 50 ms (1/20th of a second)
    per value.
 for (power = -127; power <= 127; power ++)
  {
    ST.turn(power);
    delay(50);
  }

 // 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);
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

2.Settings/MinVoltage/MinVoltage.ino

Sets the minimum voltage.

```
// Set Minimum Voltage Sample for Packet Serial
// Copyright (c) 2012 Dimension Engineering LLC
// See license.txt for license details.

// The values in this sample are specifically for
    the Sabertooth 2x25, and may
// not have the same effect on other models.
#include <Sabertooth.h>

Sabertooth ST(128);

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

 // This setting does not persist between power
    cycles.
 // See the Packet Serial section of the
    documentation for what values to use
 // for the minimum voltage command. It may vary
    between Sabertooth models
 // (2x25, 2x60, etc.).
 //
 // On a Sabertooth 2x25, the value is (Desired
    Volts - 6) X 5.
```

```
 // So, in this sample, we'll make the low battery
    cutoff 12V: (12 - 6) X 5 = 30.
  ST.setMinVoltage(30);
}

void loop()
{
  ST.motor(1, 50);
  delay(5000);

  ST.motor(1, -50);
  delay(5000);
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

2.Settings/Persistent/BaudRate/BaudRate.ino

Changes the Packet Serial baud rate.

WARNING: This sample makes changes that will persist between restarts.

```
// Set Baud Rate Sample for Packet Serial
// Copyright (c) 2012 Dimension Engineering LLC
// See license.txt for license details.

// WARNING: This sample makes changes that will
    persist between restarts.
// NOTE: The setBaudRate function will only have an
    effect on V2 controllers (2x12, 2x25 V2, 2x60,
    SyRen 50).
//        Earlier controllers automatically detect
    the baud rate you choose in Serial.begin
//        when you call the autobaud function.
    Autobaud was replaced in V2 controllers for
    reliability
//        in the event that the Sabertooth lost
    power.
#include <Sabertooth.h>

Sabertooth ST(128);

void setup()
{
 // This sample will tell the Sabertooth *at 9600
```

```
    baud* to *switch to 2400 baud*.
 // Keep in mind you must send the command to
    change the baud rate *at the baud rate
 // the Sabertooth is listening at* (factory
    default is 9600). After that, if it works,
 // you will be able to communicate using the new
    baud rate.
 //
 // Options are:
 //   2400
 //   9600
 //   19200
 //   38400
 //   115200 (only supported by some devices such
    as 2X60 -- check the device's datasheet)
 //
 // WARNING: The Sabertooth remembers this command
    between restarts.
 // To change your Sabertooth back to its default,
    you must *be at the baud rate you've
 // set the Sabertooth to*, and then call
    ST.setBaudRate(9600)
  SabertoothTXPinSerial.begin(9600);
  ST.setBaudRate(2400);
  SabertoothTXPinSerial.end();

 // OK, we're at 2400. Let's talk to the Sabertooth
    at that speed.
  SabertoothTXPinSerial.begin(2400);
}

void loop()
{
  ST.drive(0);
  ST.turn(20);
  delay(2000);

  ST.turn(-20);
```

```
  delay(2000);
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

2.Settings/Persistent/Deadband/Deadband.ino

Modifies the deadband.

WARNING: This sample makes changes that will persist between restarts.

```
// Set Deadband Sample for Packet Serial
// Copyright (c) 2012 Dimension Engineering LLC
// See license.txt for license details.

// WARNING: This sample makes changes that will
    persist between restarts AND in all modes.
#include <Sabertooth.h>

Sabertooth ST(128);

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

 // This makes the deadband from -20 to 20 (of
    127).
 // If your commands for a motor stay entirely
    within the deadband for more than
 // a second, the motor driver will stop the motor.
 // WARNING: The Sabertooth remembers this command
    between restarts AND in all modes.
 // To change your Sabertooth back to its default,
```

```
    call ST.setDeadband(0)
  ST.setDeadband(20);
}

void loop()
{
 // 50 is greater than 20, so the motor moves.
  ST.motor(1, 50);
  delay(5000);

 // 10 is NOT, so the motor does not move.
  ST.motor(1, 10);
  delay(5000);
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

2.Settings/Persistent/MaxVoltage/MaxVoltage.inc

Modifies the maximum voltage.

WARNING: This sample makes changes that will persist between restarts.

```
// Set Maximum Voltage Sample for Packet Serial
// Copyright (c) 2012 Dimension Engineering LLC
// See license.txt for license details.

// WARNING: This sample makes changes that will
    persist between restarts AND in all modes.
//          The values in this sample are
    specifically for the Sabertooth 2x25, and may
//          not have the same effect on other
    models.
#include <Sabertooth.h>

Sabertooth ST(128);

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

 // See the Packet Serial section of the
    documentation for what values to use
 // for the maximum voltage command. It may vary
    between Sabertooth models
```

```
 // (2x25, 2x60, etc.).
 //
 // On a Sabertooth 2x25, the value is (Desired
    Volts) X 5.12.
 // In this sample, we'll cap the max voltage
    before the motor driver does
 // a hard brake at 14V. For a 12V ATX power supply
    this might be reasonable --
 // at 16V they tend to shut off. Here, if the
    voltage climbs above
 // 14V due to regenerative braking, the Sabertooth
    will go into hard brake instead.
 // While this is occuring, the red Error LED will
    turn on.
 //
 // 14 X 5.12 = 71.68, so we'll go with 71, cutting
    off slightly below 14V.
 //
 // WARNING: This setting persists between power
    cycles.
  ST.setMaxVoltage(71);
}

void loop()
{
  ST.motor(1, 50);
  delay(5000);

  ST.motor(1, -50);
  delay(5000);
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

2.Settings/Persistent/Ramping/Ramping.ino

Modifies the ramp time.

WARNING: This sample makes changes that will persist between restarts.

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

// WARNING: This sample makes changes that will
    persist between restarts AND in all modes.
#include <Sabertooth.h>

Sabertooth ST(128);

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

 // See the Sabertooth 2x60 documentation for
    information on ramping values.
 // There are three ranges: 1-10 (Fast), 11-20
    (Slow), and 21-80 (Intermediate).
 // The ramping value 14 used here sets a ramp time
    of 4 seconds for full
 // forward-to-full reverse.
 //
```

```
 // 0 turns off ramping. Turning off ramping
    requires a power cycle.
 //
 // WARNING: The Sabertooth remembers this command
    between restarts AND in all modes.
 // To change your Sabertooth back to its default,
    call ST.setRamping(0)
  ST.setRamping(14);
}

void loop()
{
 // Full forward, both motors.
  ST.motor(1, 127);
  ST.motor(2, 127);
  delay(5000);

 // Full reverse
  ST.motor(1, -127);
  ST.motor(2, -127);
  delay(5000);
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

2.Settings/SerialTimeout/SerialTimeout.ino

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

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

#include <Sabertooth.h>

Sabertooth ST(128);

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

 // setTimeout rounds up to the nearest 100
    milliseconds, so this 950 will actually be 1
    second.
 // A value of 0 disables the serial timeout.
  ST.setTimeout(950);
}

void loop()
{
 // Set motor 1 to reverse 20 (out of 127), and
    sleep for 5 seconds.
 // Notice how it cuts out after 1 second -- this
```

```
    is the serial timeout in action.
 // Since we configured it in setup() for 1 second,
    1 second without any new
 // commands will cause the motors to stop.
  ST.motor(1, -20);
  delay(5000);

 // Why do this?
 // If the S1 wire gets cut for some reason, or if
    your program crashes,
 // the Sabertooth will stop receiving commands
    from the Arduino.
 // With a timeout, your robot will stop. So, it's
    a safety feature mostly.
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

3.Advanced/SharedLine/SharedLine.ino

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

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

#include <Sabertooth.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.
Sabertooth ST1[2] = { Sabertooth(128),
    Sabertooth(129) };
Sabertooth ST2(130);

void setup()
{
  SabertoothTXPinSerial.begin(9600);
 Sabertooth::autobaud(SabertoothTXPinSerial); //
    Autobaud is for the whole serial line -- you
    don't need to do
 // it for each individual motor driver. This is
    the version of
 // the autobaud command that is not tied to a
    particular
```

```
 // Sabertooth object.
 // See the examples in 1.Basics for information on
    whether you
 // need this line at all.
}

void loop()
{
 // ST1[0] (address 128) has power 50 (of 127 max)
    on M1,
 // ST1[1] (address 129) has power 60 (of 127 max)
    on M2, and
 // ST2    (address 130) we'll do tank-style and
    have it drive 20 and turn right 50.
 // Do this for 5 seconds.
  ST1[0].motor(1, 50);
  ST1[1].motor(2, 60);
  ST2.drive(20);
  ST2.turn(50);
  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(-40);
  delay(5000);
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

3.Advanced/SoftwareSerial/SoftwareSerial.ino

Uses a pin other than TX to connect to S1.

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

#include <SoftwareSerial.h>
#include <Sabertooth.h>

SoftwareSerial SWSerial(NOT_A_PIN, 11); // RX on no
    pin (unused), TX on pin 11 (to S1).
Sabertooth ST(128, SWSerial); // Address 128, and
    use SWSerial as the serial port.

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

void loop()
{
 int power;

 // Ramp from -127 to 127 (full reverse to full
    forward), waiting 20 ms (1/50th of a second)
    per value.
 for (power = -127; power <= 127; power ++)
```

```
  {
    ST.motor(1, power);
    delay(20);
  }

 // Now go back the way we came.
 for (power = 127; power >= -127; power --)
  {
    ST.motor(1, power);
    delay(20);
  }
}
```

SyRen/Sabertooth Packet Serial Library for Arduino

Control your SyRen or Sabertooth with reliable Packet Serial.

Sabertooth Member List

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

address() const	S
autobaud(boolean dontWait=false) const	S
autobaud(SabertoothStream &port, boolean dontWait=false)	S
command(byte command, byte value) const	S
drive(int power) const	S
motor(int power) const	S
motor(byte motor, int power) const	S
port() const	S
Sabertooth(byte address)	S
Sabertooth(byte address, SabertoothStream &port)	S
setBaudRate(long baudRate) const	S
setDeadband(byte value) const	S
setMaxVoltage(byte value) const	S
setMinVoltage(byte value) const	S
setRamping(byte value) const	S
setTimeout(int milliseconds) const	S
stop() const	S
turn(int power) const	S