PJRC Store

- Teensy 4.1, $31.50
- Teensy 4.0, $23.80

Teensy

- Main Page
- **Hardware**
- **Getting Started**
- **Tutorial**
- **How-To Tips**
- **Code Library**
- Projects
- **Teensyduino**
  - Main
  - Download+Install
  - Basic Usage
  - Digital I/O
  - PWM & Tone
  - **Timing**
  - Code Security
  - Startup
  - USB Serial
  - USB Keyboard
  - USB Mouse
  - USB Joystick
  - USB MIDI
  - USB Flight Sim
  - Serial
  - **Libraries**
    - Main List
    - GLCD
    - LiquidCrystal
    - OctoWS2811

AltSoftSerial Library

AltSoftSerial emulates an additional serial port, allowing you to communicate with another serial device.

Download: Included with the Teensyduino Installer
Latest Developments on Github

AltSoftSerial is particularly useful when simultaneous data flows are needed. More details below.

AltSoftSerial is capable of running up to 31250 baud on 16 MHz AVR, or up to 400000 baud on Teensy 3.2 at 96 MHz. Slower baud rates are recommended when other code may delay AltSoftSerial's interrupt response.

Serial Port Options

1. **HardwareSerial** - Best performance. Always use this first, if available! Teensy and Teensy++ have one HardwareSerial port which is available (not used for uploading or the Arduino Serial Monitor). Arduino Mega has 3 extra HardwareSerial ports. Arduino Uno has none.
2. **AltSoftSerial** - Can simultaneously transmit and receive. Minimal interference with simultaneous use of HardwareSerial and other libraries. Consumes a 16 bit timer (and will not work with any libraries which need that timer) and disables some PWM pins. Can be sensitive to interrupt usage by other libraries.
3. **SoftwareSerial**(formerly "NewSoftSerial") - Can have multiple instances on almost any pins, but only 1 can be active at a time. Can not simultaneously transmit and receive. Can interfere with other libraries or HardwareSerial if used at slower baud rates. Can be sensitive to interrupt usage by other libraries.
4. **Old SoftwareSerial** (SoftwareSerial in Arduino 0023 & earlier) - Very poor performance.

Hardware Requirements

Board	Transmit Pin	Receive Pin	Unusable PWM
Teensy 3.5 / 3.6	21	20	22
Teensy 3.0 / 3.1 / 3.2	21	20	22

- TimerOne
- MsTimer2
- EEPROM

Reference

Teensy 2.0	9	10	(none)
Teensy++ 2.0	25	4	26, 27
Arduino Uno, Duemilanove, LilyPad, Mini (& other ATMEGA328)	9	8	10
Arduino Leonardo, Yun, Micro	5	13	(none)
Arduino Mega	46	48	44, 45
Wiring-S	5	6	4
Sanguino	13	14	12

The "Unusable PWM" pins can be used normally, with digitalRead() or digitalWrite(), but their PWM function controlled by analogWrite() will not work properly, because AltSoftSerial uses the timer which controls that pin's PWM feature.

Yet another alternate software serial exists for only Arduino Uno, using timer2 and pins 3 and 4. It appears to only work with the ATMEGA328 chip on Uno.

Basic Usage

AltSoftSerial mySerial;

Create the AltSoftSerial object. Only one AltSoftSerial can be used, with the fixed pin assignments shown above.

mySerial.begin(baud);

Initialize the port to communicate at a specific baud rate.

mySerial.print(anything);

Print a number or text. This works the same as Serial.print().

mySerial.available();

Returns the number of bytes received, which can be read.

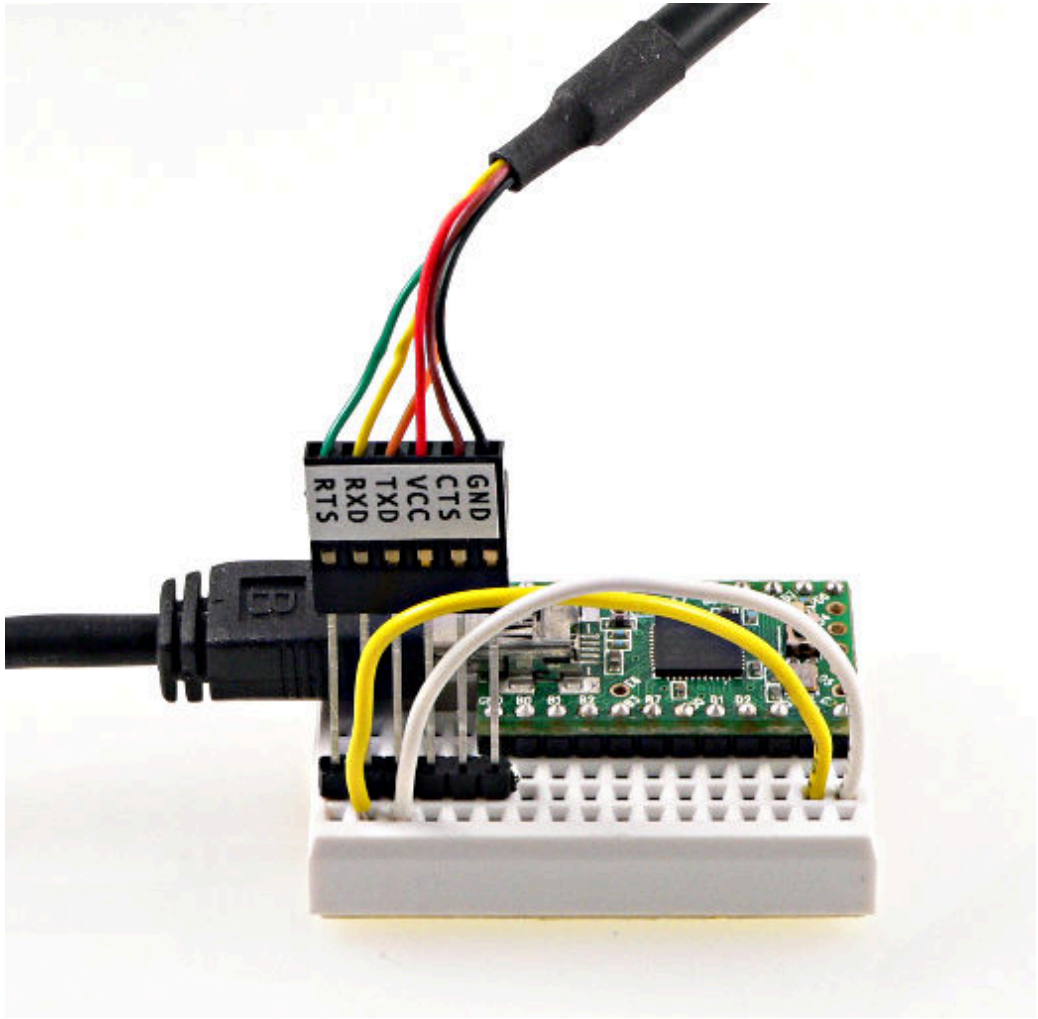
mySerial.read();

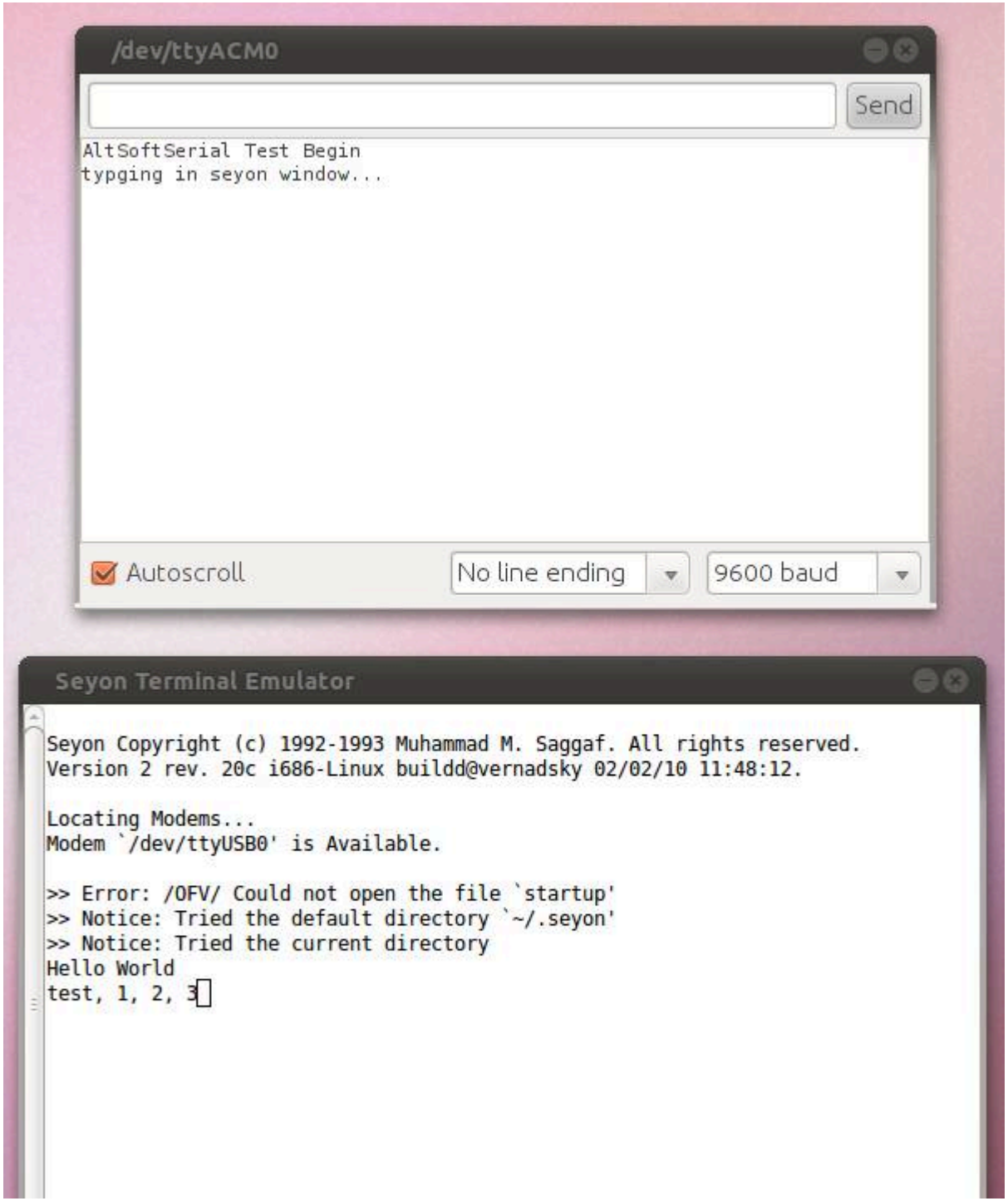
Reads the next byte from the port. If nothing has been received, -1 is returned.

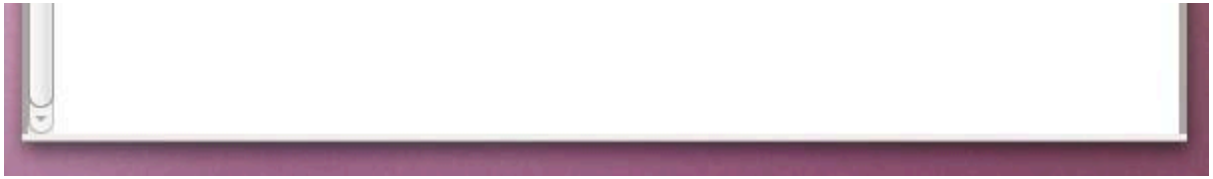
Example Program

```
#include <AltSoftSerial.h>

AltSoftSerial altSerial;
```

```
void setup() {
  Serial.begin(9600);
  Serial.println("AltSoftSerial Test Begin");
  altSerial.begin(9600);
  altSerial.println("Hello World");
}

void loop() {
  char c;

  if (Serial.available()) {
    c = Serial.read();
    altSerial.print(c);
  }
  if (altSerial.available()) {
    c = altSerial.read();
    Serial.print(c);
  }
}
```




/dev/ttyACM0
Send
AltSoftSerial Test Begin
typging in seyon window...
Autoscroll
No line ending
9600 baud
Seyon Terminal Emulator
Seyon Copyright (c) 1992-1993 Muhammad M. Saggaf. All rights reserved.
Version 2 rev. 20c i686-Linux buildd@vernadsky 02/02/10 11:48:12.
Locating Modems...
Modem `/dev/ttyUSB0' is Available.
>> Error: /OFV/ Could not open the file `startup'
>> Notice: Tried the default directory `~/.seyon'
>> Notice: Tried the current directory
Hello World
test, 1, 2, 3



Interrupt Latency Requirements

AltSoftSerial can withstand approximately 1 bit time of interrupt latency from other libraries or functions. Most libraries, HardwareSerial, and the millis() timekeeping interrupt are designed to minimize CPU time spend executing interrupts or accessing data with interrupts disabled.

However, some libraries may disable interrupts for longer than 1 bit time. If they do, AltSoftSerial will not work properly. If you discover erroneous results from AltSoftSerial, please try testing without other libraries.

Timer Usage

AltSoftSerial uses a 16 bit hardware timer to generate the transmit output and measure the receive input waveforms. Any other library which needs the same timer, and the PWM pins which depend upon that timer, will not work.

Some boards, like Teensy, Teensy++ and Arduino Mega, have more than 1 timer which is suitable for AltSoftSerial. You can configure which timer AltSoftSerial uses by editing "config/known_boards.h" within the library. This may allow AltSoftSerial to be used together with other libraries which require the timer which AltSoftSerial uses by default.

AltSoftSerial & SoftwareSerial Usable Speed

A commonly asked question is the maximum baud rate these library can support. Both can work with approximately 1 bit time of interrupt latency from OTHER code. So if other interrupts take a maximum of 15 µs (eg, some libraries), then a baud rate of 57600 ought to be possible.

Without other libraries, on Teensy or Arduino (with the issue 776 fix), interrupt latency is about 3 to 4 µs. 115200 baud is possible.

However, the maximum baud rate is often not the most important question. Each library imposes interrupt latency on other libraries. AltSoftSerial causes approximately 2-3 µs latency. SoftwareSerial causes 10 bit times of latency for other libraries. Running at 57600 baud, that's 174 µs! This latency is the primary difference between AltSoftSerial and SoftwareSerial.

To see this in action, you can try the example that comes with SoftwareSerial in Arduino 1.0. If you type "Goodnight" in the Arduino Serial Monitor, you'll see what actually comes out of pin 3 at 4800 baud is "Goot". The characters "dnigh" are lost. The reason is because while SoftwareSerial is sending the letter "G" at 4800, the letters "oodnigh" arrive at 57600 baud. Only "oo" are held in the UART registers. The rest are lost because interrupts were disabled for too long. AltSoftSerial can handle this test easily, since it does not lock out interrupts for long times.

Using Both SoftwareSerial and AltSoftSerial

It is possible to use both SoftwareSerial and AltSoftSerial, and of course HardwareSerial, to have 3 serial ports! However, the baud rates must be chosen carefully!

Because SoftwareSerial creates 10 bit times of latency for other libraries, it should be used for a device needing high baud rate. SoftwareSerial should NOT be used at slow baud rates, because it will interfere with the other ports. SoftwareSerial can not simultaenously transmit and receive, so it should be used with a device that never sends in both directions at once.

HardwareSerial can tolerate up to 20 bit times latency for receive, or 10 bit times for non-stop transmit. To maintain full transmit speed, HardwareSerial's baud rate should be no greater than SoftwareSerial's. If continuous transmit is not needed, HardwareSerial can use a baud rate almost twice as fast as SoftwareSerial and still reliably receive. But more than twice SoftwareSerial's baud will not be able to receive reliably.

AltSoftSerial can tolerate almost 1 bit time latency, so its baud rate must be at least 10 times less than the baud rate used for SoftwareSerial.

If the baud rates are chosen wisely, all 3 can work together reliably!

Development

New development on AltSoftSerial is done on this GitHub repository:

https://github.com/PaulStoffregen/AltSoftSerial

Please report any bugs or submit pull requests via GitHub!