

## Temas

: Más

## CATEGORÍAS

Development Tools

Projects

Official Hardware

Other Hardware

Community

International

Deutsch

Español

Français

Italiano

Todas las categorías

## Teensyduino on Teensy 2.0 - UART Serial

system

ene. 2012 post #1

Since I cannot find an official Teensyduino board, I figured I might post here. Does anyone have experience with UART serial (the hardware serial - not the USB) on the teensy? I'm wondering if write() blocks instruction or uses a buffer. I'm also wondering what the max reliable baud rate is? They list some on their site, but I was wondering if anyone has experience with higher speeds?

Thanks!

Coding\_Badly

ene. 2012 post #2

gcotten:

Does anyone have experience with UART serial (the hardware serial - not the USB) on the teensy?

Yes. It works well (just like the UART on the 328). It's one of the features I really like on the Teensy.

I'm wondering if write() blocks instruction or uses a buffer.

Let's check the source code (my copy is several months old so things may have changed)...

```
void HardwareSerial::write(uint8_t c)
{
    uint8_t i;

    if (!(UCSR1B & (1<<TXEN1))) return;
    if (tx_enable_pin < 255 && !transmitting) {
        digitalWrite(tx_enable_pin, HIGH);
    }
    i = tx_buffer_head + 1;
    if (i >= TX_BUFFER_SIZE) i = 0;
    while (tx_buffer_tail == i) ; // wait until space in buffer
    tx_buffer[i] = c;
    transmitting = 1;
    tx_buffer_head = i;
    UCSR1B = (1<<RXEN1) | (1<<TXCIE1) | (1<<TXEN1) | (1<<RXCIE1) | (1<<UDRIE1);
}
```

Looks very "buffery" to me.

I'm also wondering what the max reliable baud rate is?

"Reliable"? There's no way to answer that question. It depends on: medium quality, medium length, receiver hardware, receiver and transmitter software. And, you have not defined what "reliable" means in your application.

They list some on their site, but I was wondering if anyone has experience with higher speeds?

I have not had any problems at 115200.

If possible, you will have better results using baud rates that are "16 MHz compatible". Go here...

<http://www.wormfood.net/avrbaudcalc.php>

Find the "16 Mhz" table. Choose baud rates in green. Or, use the equations in the datasheet to determine baud rates with zero error (250K is one such choice).

[Saltar al contenido principal](#)

I recently did a project with another non-USB AVR chip sending debug data, and a Teensy receiving it and resending it to a PC. The Teensy was also connected to the reset line and could do ISP reprogramming of that other AVR chip, as well as a few checks of various analog voltages.

I wanted the serial to run as fast as possible, but I didn't want to do a lot of work. I did go to the trouble of reading to a 64 byte buffer, rather than 1-byte-at-a-time, but otherwise I just went with things as they are. Here's the code.

```
void loop() {
  char buf[64];

  if (cpu_is_running) {
    int n = Uart.available();
    if (n > 0) {
      if (n > sizeof(buf)) n = sizeof(buf);
      Uart.readBytes(buf, n);
      Serial.write((uint8_t *)buf, n);
    }
  }
}
```

With the code above, it turned the fastest reliable baud rate was 666667 bits/sec.

```
#define BAUDRATE 666667
// #define BAUDRATE 1000000 // Teensyduino can't keep up...
// #define BAUDRATE 2000000
// #define BAUDRATE 115200
HardwareSerial Uart = HardwareSerial();
```

It did work pretty well for short bursts of data at 1 Mbit/sec. But if the other AVR chip sent sustained maximum rate data, ultimately it couldn't keep up.

If your program were structured in the common 1-byte-at-a-time approach, you'd incur more overhead, especially since each write to the PC would have the overhead of manipulating the USB buffers. It very likely would not keep up at 666667 bps with so much more overhead.





Internally, readBytes() calls the read function for 1 byte at a time. I may someday optimize readBytes and the underlying Stream class for block reads. When/if that happens, the code above would very likely become able to reliably handle 1 Mbit/sec.

Of course, if your program is doing something different with the data, the actual reliable maximum data rate will depend heavily on what you're doing, and especially if you manage data in blocks or 1-byte-at-a-time. But hopefully this 1 use case gives you at least some idea of the underlying capability to move at least 667 kbits/sec speed from the hardware serial port to the USB virtual serial port.



ps: another caveat is this does NOT apply to regular Arduino boards. The HardwareSerial code in Teensyduino is heavily optimized. Arduino's HardwareSerial code is much slower.

Cerrado el 6 may. 2021

### 🔖 Temas relacionados

Tema	Respuestas	Vistas	Actividad
  <a href="#">Teensyduino (direct USB, not serial via FTDI chip)</a>	<a href="#">5</a>	<a href="#">4,0k</a>	<a href="#">ene. 2009</a>
  <a href="#">Reading serial data from Teensy 3.6?</a>	<a href="#">5</a>	<a href="#">1,2k</a>	<a href="#">mar. 2019</a>

[Saltar al contenido principal](#)

Tema	Respuestas	Vistas	Actividad
  <a href="#">ATmega16U4 based arduino</a>	11	7,2k	dic. 2009
  <a href="#">HardwareSerial Update Confusion</a>	3	762	jun. 2013
  <a href="#">Serial interrupts</a>	29	4,6k	sep. 2017

[Back to top](#)

[Help Center](#)  
[Contact Us](#)  
[Trademark & Copyright](#)  
[Brand Guidelines](#)

[Distributors](#)  
[Careers](#)

FOLLOW US