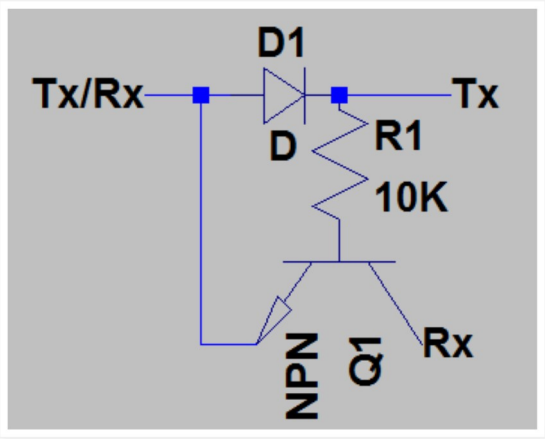
Nerd Ralph

science and technology stuff

Saturday, January 11, 2014

AVR half-duplex software UART supporting single pin operation

Many of the ATtiny MCUs have no hardware UART limited number of pins. Arduino tiny cores uses the TinyDebugSerial class which is output only, so using serial input requires extra code. I have written a small software serial UART which can share a single pin for Rx & Tx when a simple external circuit is used.

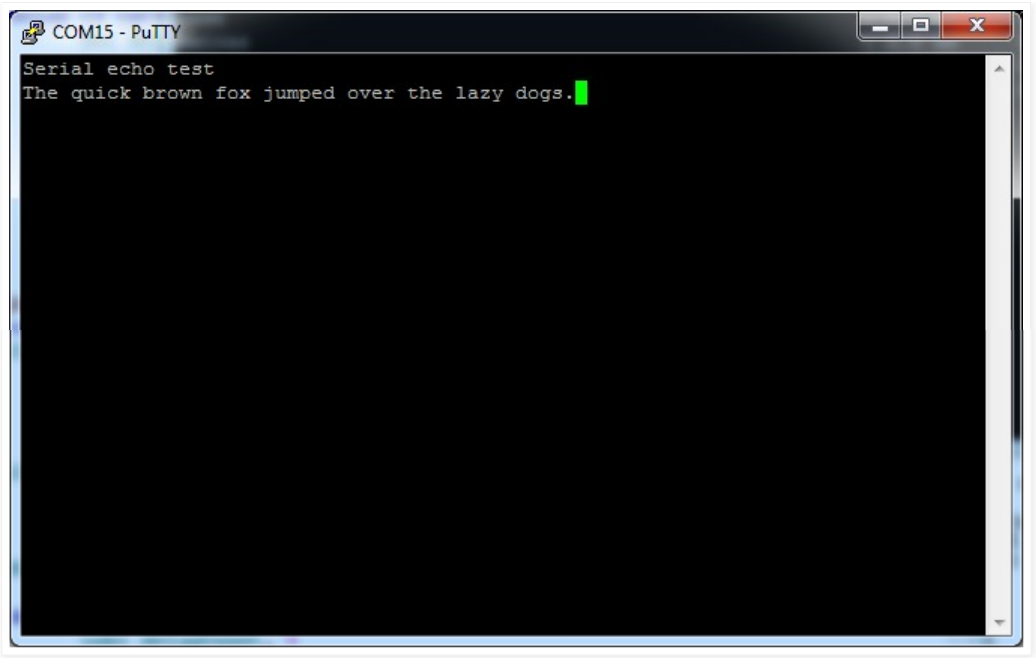


To understand the circuit, a tutorial on TTL serial communication may help. When the TTL serial adapter is not transmitting, the voltage from the Tx pin keeps Q1 turned on, and the AVR pin (Tx/Rx) will sense a high voltage, indicating idle state. When the AVR transmits a 0, with Q1 on, Rx will get pulled low indicating a 0. R1 ensures current flow through the base of Q1 is kept below 1mA. When the AVR transmits a 1, Rx will no longer be pulled low, and Rx will return to high state. When the serial adapter is transmitting a 0, D1 will allow it to pull the AVR pin low. With no base current, Q1 will be turned off, and the Rx line on the serial adapter will be disconnected from the transmission.

I've written the serial code to work as an Arduino library. Compiled it uses just 62 bytes of flash, and does not require any RAM (there is no buffering). As it is written in AVR assembly, it will support very high baud rates - up to 460.8kbps at 16Mhz. The default baud rate of 115.2kbps is defined in BasicSerial3.h, and can be changed with the BAUD_RATE define. The library defaults to use PB5 for both Tx and Rx. It can be changed with the UART_Tx and UART_Rx defines in BasicSerial3.S. Here's an example sketch to that uses it:

```
#include <BasicSerial3.h>

// sketch to test Serial

void setup()
{
}

void serOut(const char* str)
{
   while (*str) TxByte (*str++);
}

void loop(){
  byte c;
  serOut("Serial echo test\n\r");
  while ( c = RxByte() ){
    TxByte(c);
  }
  delay(1000);
}
```

Here's a screen shot of putty running the example:



2020 Update
Over the past several years I've written many updated and improved bit-bang UARTs. Instead of the BasicSerial code in this post, I recommend using picoUART. It supports a wider range of baud rates, and the bit timing is accurate to the cycle.

Posted by Ralph Doncaster at 7:47 PM

99 comments:

Sancho January 13, 2014 at 12:15 PM

Amazingly simple and clever solution :-) Thanks for sharing.

Reply

mdb January 13, 2014 at 12:39 PM

Absolutely great! Thanks for sharing!

Reply

Torchris January 13, 2014 at 1:25 PM

Wow! This is awesome. I will try this with my DigiSparks!

Reply

Replies

Ralph Doncaster January 14, 2014 at 4:52 AM

I tested it on a DigiSpark going to a PL-2303HX USB-serial adapter. It's great for the digisparks, trinkets, and other USB-connected ATtiny85s since 2 of their pins are crippled by being permanantly wired to the USB D+ & D-.

Reply

Anonymous January 14, 2014 at 2:33 AM

Great idea!

Thanks Ralph

Reply

Anonymous January 14, 2014 at 2:44 AM

Quick question Ralph,

Would it work to connect for instance two Arduino Uno?

Thanks
Max

Reply

Replies

Ralph Doncaster January 14, 2014 at 4:54 AM

It should be even easier connecting two arduinos, with both of them running the half-duplex UART code. They could be both connected directly. At most you might need an external pullup resistor on the Tx/Rx line

Anonymous January 14, 2014 at 7:27 AM

Hi Ralph,

Thanks for the quick answer!

Interesting... I thought about a direct connection tx/rx to tx/rx but I didn't think it was possible. I expected it required some software tweaking or some extra interface circuit.

Max

Anonymous January 16, 2014 at 1:59 AM

Hi Ralph,

Just to be sure I understand...

Let's assume that two Arduinos (e.g. Uno) are connected using a single tx/rx line with a pull-up resistor. Is the pin used in the Arduinos defined as UART_Tx and UART_Rx in BasicSerial3.S ?

Thanks
Max

Reply

Unknown January 14, 2014 at 9:33 PM

Can this be used for a one line Interrupt / Enable?

Reply

Replies

Ralph Doncaster January 15, 2014 at 10:02 AM

Not exactly sure what you're asking, but if you mean can the RxByte code used in an ISR, it can with a few code changes. You'd need to remove the cli instruction and add the typical ISR code to save/restore SREG & the used registers (r22 & r24).

Reply

Unknown January 16, 2014 at 9:21 AM

I could this be used for gps data use?

Reply

Replies

Ralph Doncaster January 16, 2014 at 2:18 PM

It can be used with any ttl serial interface.

Reply

Anonymous January 18, 2014 at 7:14 AM

Does not compile with Arduino 1.5.2 or 1.5.5.

In function `loop':
undefined reference to `TxTimedByte'
undefined reference to `RxTimedByte'

Reply

Replies

Ralph Doncaster January 19, 2014 at 10:54 PM

I used 1.04, and 1.05 should work the same.
When 1.5.x goes out of beta, I'll try it.

Unknown February 5, 2016 at 8:41 AM

I have the same error. 1.6.6 do you have any news=

Ralph Doncaster February 8, 2016 at 1:34 PM

I don't think I'll be supporting it with the Ardruino IDE any more, as I rarely use the IDE (I do most of my development from the cli). I have been thinking about expanding http://nerdralph.blogspot.ca/2015/10/beta-picowiring-arduino-compatible.html and making it work as an Arduino Attiny core, which would include my asm soft serial implementation. The board manager seems to make this a lot easier than it used to be.

Reply

Guillaume January 18, 2014 at 4:25 PM

Clever trick! thank you to share it. I have added your article to the last embedded systems weekly issue http://embedsysweekly.com/issue13-microcontroller-8051-pic/
I hope you'll like how I presented it.

Reply

Replies

Ralph Doncaster January 19, 2014 at 10:58 PM

Good, though your English is not quite perfect.

Reply

Unknown February 17, 2014 at 1:15 AM

Hi Ralph,

is it possible to use the code without sharing a single pin or does it cause any problems??

Thanks for sharing your code!

Reply

Replies

Ralph Doncaster February 19, 2014 at 8:02 PM

It should work fine Johann - just change the defines for the tx and rx pins.
If you're looking for something you can use with plain AVR code instead of Arduino, I have a similar soft UART here:
https://code.google.com/p/picoboot/source/browse/trunk/BBUart.S

Martii April 28, 2014 at 1:07 AM

Hey, would this work with lets say Attiny13A? I don't need anything fast - 57600 or even 28800 bps will be more than enough - I have tried your BBUart.S but avr-gcc complains about this code.

Ralph Doncaster April 28, 2014 at 4:25 AM

Someone on the Arduino forums said it worked on the tiny13 for him:
http://forum.arduino.cc/index.php?topic=207467.0
I'll have a better version of the UART finished soon, along with a decent Makefile. It will work by defining BAUDRATE in the application code (i.e. where your main is) so the uart code doesn't have to be modified.

Martii April 28, 2014 at 5:37 AM

Ok so adding:

#include
#define UART_Port (PORTB-0x20)

helped - all ASM errors are gone now. The only problem I have now is that somehow my main.c code is unable to see TxByte - even if there is extern directive pointing that it's defined in ASM (I want to use TxByte only as I need some UART just to debug ADC values

sometimes and using terminal seems to be the best way).

Martii April 28, 2014 at 5:53 AM

This comment has been removed by the author.

Martii April 28, 2014 at 6:52 AM

This comment has been removed by the author.

Martii April 28, 2014 at 7:40 AM

OK I have tried TXDELAY 25 (as it was said on the forums) - I get garbage on my screen regardless of the speed I set in minicom (115200, 57600, etc) different garbage for different speeds - Tx pin of tiny13 is connected directly to Rx of my USB serial converter.

Ralph Doncaster April 28, 2014 at 12:46 PM

I looked at the tiny13 datasheet, and it says, "The device is shipped with CKSEL = “10”, SUT = “10”, and CKDIV8 programmed." That means 9.6Mhz/8 = 1.2Mhz. That's going to be to slow for 115.2, and even 57.6 would be tricky. At 38.4kbps there's 31.25 cycles per bit, so 8 for the delay counter would give a total of 31 cycles per bit (<1% timing error) if you use the arduino-compatible version of the code (linked above). If you're using a modified version of BBUart, (the one that calls delay3Cycle), TXDELAY should be 6, which would give a total of 32 cycles and 2.4% timing error. Most UARTs can handle up to 5% timing error.

Martii April 28, 2014 at 11:25 PM

Thanks Ralph - unfortunately this didn't help - I tried both 1.2Mhz and 9.6MHz - no luck - still garbage - but at the moment I power this via USB - so I'll do some proper stabilization and filtering via external battery bank to see if this helps. I'll also try to add decoupling capacitor and will let you know :) - thanks for all help and this brilliant tiny code.

Martii April 30, 2014 at 1:33 AM

My bad - this is tiny13a - this comes with 9.6MHz by default - but it should be no different to tiny13 except it's less power hungry.

Martii May 4, 2014 at 9:31 AM

So what is the formula for TXDELAY in BBUart.S? I might have to try few different speeds to see if any of them will work at all.

Ralph Doncaster May 5, 2014 at 4:48 AM

If you're using the latest version with the nop after the out, its:
Its ((BAUDRATE/F_CPU) -15) /3).
The nop is there just to make the TxByte and RxByte timing in sync, so if you're only using TxByte there's no benefit to the nop. Without it subtract 14 cycles from BAUDRATE/F_CPU instead of 15.

If you have a logic analyzer you could check the output timing. If you don't, you could use the audio input on a computer as a low-speed scope to do the same thing. If you can wait a few days I'll be writing an article on wiring a 3.5mm jack and about which software I like.

Martii May 6, 2014 at 12:46 PM

Thanks - all I need is TX from AVR to PC - that way I can see for ex. ADC values and anything else I desire - I'm happy to wait for any extra solution that might help - no logic analyzer at hand :(

Martii May 6, 2014 at 1:43 PM

Hmmm BAUDRATE/F_CPU gives me very small number :) so I assume it should be F_CPU/BAUDRATE?

Also I use BBUart.S from the link you left in comments and I can't see nop after out at the end of TXLoop.

Also I had to change brne delay3Cycle to brne Delay3Cycle - as I was getting an error (avr-gcc linux - is of course case sensitive)

Ralph Doncaster May 6, 2014 at 6:37 PM

This afternoon I finished the post on using the sound card as a logic analyzer.
You're correct about F_CPU/BAUDRATE. You can see the forumula from the macros in BBUart.h. I've done a few updates to the BBUart code, but if you use the matching BBUart.h you'll get the right timing.
For the version I think you're using, TXDELAY should work out to 23 for 115.2kbps @9.6Mhz (with 0.4% timing error). The 25 that you tried would have been close enough that it would see characters but they would be corrupted.
In the most recent version I've tweaked things a bit and added the code for echo test (uart_echo.c).
https://code.google.com/p/nerdralph/source/browse/#svn%2Ftrunk%2Favr

Martii May 7, 2014 at 1:06 AM

This comment has been removed by the author.

Martii May 7, 2014 at 1:09 AM

Still no luck - the only thing I can say is that my rs-usb converter (based on pl2303) is playing important role here.
Looks like you're Windows user - so I can't use your latest entry tips on sound card logic analyzer :(

Ralph Doncaster May 7, 2014 at 7:25 AM

I do most of my development on Linux, and use Windows for client-side stuff. Under Linux any sound recorder like KDE has would do the trick for using the audio input as a logic analyzer.

Martii May 14, 2014 at 2:26 AM

Just follow-up - the problem was with usb-serial cable - my pl2303 is not good. I visited my friend who has FTDI cable and surprise surprise - it WORKS just fine! - thanks for help :)

Reply

sergio ambort February 27, 2014 at 3:49 AM

This comment has been removed by the author.

Reply

Anonymous February 27, 2014 at 3:50 AM

sorry, i get
avr-gcc: error: unrecognized command line option '-assembler-with-cpp'
any idea?

Reply

Replies

Ralph Doncaster February 27, 2014 at 3:58 AM

You should have at least version 4.3.2 of avr-gcc. Check it with avr-gcc --version.

Anonymous February 27, 2014 at 6:22 AM

ok, thanks ralph!
if i use digispark-arduino-1.0.4 ide instead of arduino-1.0.5, it compile well!!!
please could you suggest the appropriate diodes and resistance for use with attiny85 in 8 mhz 3.3V mode or 16 mhz mode 5V mode?
best regards, pescadito

Ralph Doncaster February 27, 2014 at 7:11 PM

That's surprising. I'll try to see what the difference is with the 1.0.5 arduino ide.
I used a 1n4148 diode, and also tested it with a schottky diode; just about any diode should work with both 3.3 & 5v. For the resistor anything from 4.7K to 47K should work - I did most of my testing with a 10K resistor.

Reply

sergio ambort February 28, 2014 at 1:09 AM

thank ralph, very kind of you!!!
i use digispark-arduino-1.0.4,micronucleous, 1n4148 and 222a and i work, downloading for digispark 16 mhz work better in 5v, but with few errors if 3.3v is provided (rx char is printed corrupted),
but them, in the loop, any sentence after TxByte(c); doesn' t execute every cicle,...only execute..sometime.....
i'm referint to

```
while ( c = RxByte() ){
TxByte(c);
serOut(".....\n\r");
}
```

any idea?

Reply

Ralph Doncaster February 28, 2014 at 6:22 AM

I can think of a couple possibilities:
1) if you're using an ATtinyx5 on a breadboard, make sure you have a decoupling capacitor (anything between 0.1 and 1uF). I've had problems go away when I used a 0.1uF cap.

2) Try dropping the baud rate to 57600 in BasicSerial3.h
#define BAUD_RATE 57600
The macros to calculate the bit delays are only within a few cycles of the correct timing. In my own code I calculate the delays by hand to get the best timing. For example, at 115200kbps each bit is 8.681 us long. At 8Mhz, that's 69.444 cycles. The delay loop is 3 cycles per itteration, and if the delay calculation macro has a rounding error the timing could be off by 4 or 5 cycles per bit.

Reply

Anonymous February 28, 2014 at 7:55 PM

thank again ralph!!
i tryed a 0.1 uF ceramic cap bewten vcc and gnd, also a 1F electrolytic cap and also 57600 and rx/tx worked well
but proble is that any sentence after TxByte(c); in the while is not executed!!!
i try to add support for some characters commands in rx but without result
it's a software problem or a hardware problem? another idea?
best regards, pescadito

Reply

Replies

Anonymous March 3, 2014 at 2:22 AM

wooo, it seems that's a voltage problem, to reduce space i was testing without a regulator, then i tested 3 differents, one return tx gabarage, other not execute some sentece, last do both without problem: was a regulable booster with a 3.4V output.....

Ralph Doncaster March 3, 2014 at 5:42 AM

Good to hear you figured it out.

Reply

Martii April 29, 2014 at 12:33 PM

Seems I'm still unable to make it work. Regardless of voltage filtering or decoupling I'm getting garbage :(Will try with Attiny85 :) - anyway I'm out of options now.
I can't wait for your code update (I presume based on timer) I hope it will fit tiny 1k flash :) including some of my code ;)

Reply

Rajesh June 30, 2014 at 4:52 AM

Hi there,
I want to use this configuration for communication between two microcontroller by single wire communication.I want to implement this just in reverse way .Do you think its possible?

Reply

Replies

Ralph Doncaster July 1, 2014 at 7:19 AM

Between 2 MCUs should work fine - then you wouldn't need the circuit to combine tx/rx onto one wire.

Reply

Goutham N August 7, 2014 at 6:17 AM

Hi,
Looks like a compact and tidy library, perfect for communicating between two MCUs over a single line.
I am fairly experienced on the Arduino platform and now trying to make two attiny85 (@8MHz) talk to one another (right now, just one way is just fine).

Here is my setup:
With the BasicSerial3 library, the header is modified as:
#define UART_Port (PORTB-0x20)
#define UART_Tx 4
#define UART_Rx 4
to use the attiny's pin 4.

In the code, i have a loop which sends out a string "A" using:
serOut("A");
//every two seconds.

On the second tiny, I have a simple code which (is supposed to) receive the string on pin 4 again (pin 4 of each MCU connected by a single line) and blink an led:
while (c = RxByte()){
//blink an LED to let me know a that a byte has been received

}

I am stuck at this point when I am not able to get it working. I know there is something simple being missed out, but unable to figure it out!
Any pointers on what might be going wrong?

Reply

Replies

Ralph Doncaster August 8, 2014 at 2:33 PM

You shouldn't subtract 32 from the port address, unless you remove the following line too:
#define __SFR_OFFSET 0

Reply

Goutham N August 9, 2014 at 12:27 AM

Hi Ralph,
Thanks for your reply.
I checked with the following:
#define UART_Port (PORTB4) // to use IO pin 4 on the ATTiny85
#define UART_Tx 4 // should i use PB4 instead of just 4 here?
#define UART_Rx 4

I tried the same code hooked on to a Mega with physical connection (from i/o pin 4 of the tiny) to digital pin 4 on the Mega
and trying to print the characters received thru:
while (c = RxByte()){
Serial.print(c);
}

Sitll no luck.
Any idea?

Reply

Replies

Ralph Doncaster August 9, 2014 at 5:31 PM

I'd try each end independently with a USB-ttl adapter. Most of them have LEDs for RX, so you'll know if any data is flowing. If you have a multimeter you can also check that the line is high when no data is being transmitted (and after the first byte is transmitted since the port is set to output mode in the TxByte code).

Reply

Goutham N August 11, 2014 at 10:18 AM

Thanks!

Reply

Erni August 19, 2014 at 7:46 AM

I have used this library on a ATtiny85 with succes
For testing purpose I used 2 pins (3 and 4) as tx and rx
The Tiny was running @8MHz and baudrate 57600.
I think that using two pins (atleast for testing) makes it eayer, not so much that can go wrong

Reply

Unknown August 26, 2014 at 6:19 AM

Hello Ralph, thank you so much for sharing your work. it's a nice piece of code that can help many people with low I/O's projects.
I am currently trying to implement your code in a project involving 2 arduino nano that need to communicate over a single wire and I have found your code to be exactly what I needed. I have managed to get the link up and running using PD6 at a nice 57600 baud without having a single byte corrupted (though I had to put some microseconds of delay to avoid having all the bytes sent in a row, since it was causing some corruption from time to time).

I am having some trouble implementing the RxByte() function though. this is because When I call it to receive data, the entire code freezes until the other microcontroller sends his data over the line. at that precise point, the code unfreezes and keeps running without problem until the next RxByte() call (this is happening in both ends).
I was wondering if there's something I am missing, or if there is a possibility of implementing a timeout function to avoid freezing the code for more than X time, which in my particular case would turn in a system reset due to the watchdog timer.
I am afraid I don't have the necessary programming skills to figure it out by myself alone :(
I really appreciate any help you can provide.

Reply

Replies

Ralph Doncaster August 26, 2014 at 11:51 AM

If your receiver doesn't process the incoming bytes as fast as the transmitter, then data will get lost.
The code is blocking for simplicity. You could add an ISR for when the Rx line goes low (start bit), which runs the RxByte function. The RXSTART timing would have to be reduced by around 4 cycles to account for the interrupt latency.

Unknown August 28, 2014 at 4:12 AM

I didn't thought about that. Implementing the ISR has solved the problem. the bad news for me is that I am limited by the arduino IDE and my low skills so I had to lose the UART port since atmega 328p has support for hardware interrupt in TX and RX pins only.
But it's not that important, finally I can now communicate over one channel without using aditional hardware :D
Thank you Ralph!

Reply

Unknown September 15, 2014 at 5:49 AM

Hello,
I would want connect Arduino with servo motor through half duplex single wire.
Can I use your code and connect the pin of arduino directly to pin of servo without diode and resistor?
My motor is Futaba RS303MR.
Data sheet: http://www.futaba.co.jp/en/dbps_data/_material_/radicon_eng/Robot/RS303MR_RS304MD_EN_112.pdf.

Thanks for sharing.

Reply

Replies

Ralph Doncaster September 15, 2014 at 6:37 AM

The serial UART code has nothing to do with servo control.

Unknown September 15, 2014 at 6:45 AM

Ok, but I want implement the protocol of servo.
Servo need to communicate a the single wire half duplex serial port 8bit, Stop bit 1, None Parity, Asynchronous.
My question is: can i use your code to create the comuncation? Can I send and receive the packet using only one wire?
Thanks

Ralph Doncaster September 23, 2014 at 6:12 PM

Sorry, I meant to get back to you when I read your reply but forgot.
I hadn't heard of intelligent servos before, which is why I misunderstood your question initially. Looking at the datasheet, you need RS485 communications. The AVR USART and my soft UART implement ttl rs232, so it would require code and/or hardware modifications to work with RS485.

Unknown September 24, 2014 at 2:39 AM

Hi Ralph,
I have tested your code without any extra component (only two wire, data and ground) and it works very well.
Now I can control the motors at 115200 baud without any problem.
Your assembly code is very optimized and in my case it works better than SoftwareSerial library for Arduino.
Thanks again for your sharing

Reply

Unknown May 5, 2015 at 7:39 PM

Hi, what about if the second device only have 1 line for tx and rx ? and the MCU have individual pins, the oppositve case. Thanks

Reply

biserx August 17, 2015 at 7:24 AM

How to choose D1 and Q1? When I go to the electronic parts shop, what do I ask?

Reply

Replies

Ralph Doncaster August 18, 2015 at 10:33 PM

I already commented about the diode (Feb 27):

"I used a 1n4148 diode, and also tested it with a schottky diode; just about any diode should work with both 3.3 & 5v."
Same goes for the transistor; any NPN will do, so I'd suggest the cheap and common 2n3904.

biserx August 18, 2015 at 11:23 PM

My apologies, I was scrolling twice to find that info, but somehow I skipped it :/
Thank you very much!

Reply

biserx August 17, 2015 at 7:34 AM

This comment has been removed by the author.

Reply

biserx August 17, 2015 at 7:34 AM

This comment has been removed by the author.

Reply

JB_AU September 5, 2015 at 6:15 AM

Sorry if 1st comment went through..
Is it possible for 2 instances, i.e. RX0/TX0 & RX1/TX1

Reply

JB_AU September 5, 2015 at 9:39 AM

I'm guessing , changing UART_Tx to UART_Tx0 & UART_Tx1 and Rx with new instances, I want to send data to a ttl laser & receive from a photodiode, yet transmit from gui tx & receive to gui rx.

Reply

Replies

Ralph Doncaster September 6, 2015 at 6:51 AM

You'd need to duplicate the code, so separate .S files for Uart0 & Uart1.
I've also updated the UART code so the tx and rx timings are matched to the exact cycle, while my original version was within +-1 cycle.
As well, the tx and rx code are in separate sections so if you were only using Tx, the code for Rx would not get linked in (when -ffunction-sections is used with avr-gcc).

Ralph Doncaster September 6, 2015 at 6:52 AM

The updated UART code is in my github repo.
https://github.com/nerdralph/nerdralph/tree/master/avr/libs/bbuart

---

Reply

JB_AU September 6, 2015 at 9:13 AM

Thank you very much.

Reply

Unknown December 10, 2015 at 6:58 AM

Hi Ralph! How are you?
I'm trying to senda data between two micros (atmega328 <-> attiny85) using same pin. I pulled up TX/RX line and have one ISR on each microcontroller for RX. Before sending data to line I disable interrupts because I don't want micro to receive own data being transmited.
So far, so good... But it's not working as expected. After tiny send his packet to mega, mega send some data back and then I can't send data anymore. I don't know if I had to change something in BasicSerial.S and .h

Could you kindly help me please? Thanks in advance, happy holydays!

http://pastie.org/10623141

Reply

Replies

Unknown December 10, 2015 at 9:57 AM

Apparently I'm losing data in the process:

530

0 š ? 607

7 0 › ? 559

9 • š ? 316

6 ™ ? 220

0 ˜ ™ ? 456

6 š š ?

Ralph Doncaster December 10, 2015 at 5:38 PM

Hi Will,
If you review the comments you'll see you're not the first to ask about using the receive code in an interrupt. The delay between triggering the interrupt and calling RxByte within the ISR will throw off the timing of the routine. RxByte is intended to be called before the start bit, as it busy-loops waiting for the start bit. Depending on the baud rate and the delay between the start bit and calling RxByte, you could certainly see corruption.

Since an interrupt-based receive routine seems to be a popular idea, I've been thinking of writing one. I'll likely do a follow-up post, since in the almost 2 years since I wrote this post, I've improved the bit-bang code and added it to my github repo.
https://github.com/nerdralph/nerdralph/tree/master/avr/libs/bbuart
This version was written for separate Rx and Tx lines, but could easily be modified for shared tx/rx.

I'm also learning inline assembler, and may do an inline asm version of bbuart as well.
http://www.nongnu.org/avr-libc/user-manual/inline_asm.html

Reply

Unknown December 11, 2015 at 5:57 AM

Ralph, thanks for your clarifications and many many thanks for writing this very optmized code.
Yesterday, reading all the comments I noticed the other guy in similar situation. I did noticed your update version too and since I'm not skilled in assembler I could not get it working on a single pin.
Considering the fact I'm using Basicserial3 single pin version at 115200 what should be done to get rid of those problema? Or should I be trying to get updated version working on single pin first?
I read something about removing CLI and RETI calls and restoring SReg status, is that right? There is two delays delays on RX, which one should be reduced?
I'm sorry, I had no intention to bother you with so many questions, but I really couldn't get it working in non blocking mode using ISR. Maybe it's time to learn something new!

Thanks again, Ralph!

Reply

Replies

Ralph Doncaster December 11, 2015 at 8:43 AM

I think you'll have to wait until I get around to writing an ISR version. I'll need to recalculate RXSTART to compensate for the interrupt overhead as well as saving used registers on the stack. It involves counting the execution time of each instruction (in/out take 1 cycle, push/pop take 2, branch takes 2 cycles when true, 1 when false...) so it's not a short or simple answer.

It will probably only take me a few hours once I get into it, but if you look at the rest of my blog you'll see there's lots of other irons I have in the fire...

Unknown December 11, 2015 at 9:54 AM

Ralph,I got it. Take your time :)
I didn't knew about your blog until I found your serial implementation. Right now I'm in the middle of the proccess of reading all of it!

Thanks,

--
Willian

Reply

Hans July 19, 2016 at 2:49 PM

Hi!
I'm using this code on an ATtiny13, and it works great! Do you know how I can send integer and/or float values? The code is only made for character arrays, and I'm really flipping' the bits and bytes with this microcontroller. Is there like a super elegant and efficient way to convert lets say the number 123 into characters and send them?

Reply

Replies

Ralph Doncaster July 20, 2016 at 5:46 PM

I have a tiny routine in C to convert a binary 8-bit value to 2 hex characters.
https://github.com/nerdralph/nerdralph/blob/master/avr/u8tohex.h
I used it in my spitest code:
https://github.com/nerdralph/nerdralph/blob/master/avr/spitest.c

Reply

Unknown April 11, 2017 at 10:22 PM

Hi Ralph,
Thank you for sharing your amazing trick. The library and the code you provided does not seem to be working for me (Arduino IDE 1.81; ATTiny85V @ 8Mhz). Having the additionally circuitry and connected to a FTDI's Rx-Tx-GND, I'll only see echo of what I type in but not what I serOut. Could you please give any pointer?

Reply

Replies

Ralph Doncaster May 5, 2017 at 6:13 AM

Hi Abdul, I just noticed your comment in the moderation queue.
I did most of my development with avr-gcc and makefiles, and just occasionally used the Arduino IDE. I expect if you use an older version of the Arduino IDE it should still work. Or you could try making the jump to command-line development.

Reply

azimishani May 30, 2017 at 11:24 PM

Hi Ralph,
Appreciate your awesome work. Please can you advise whether this can be used for communication between multiple attiny and also can hot swap functionality be added.
Many thanks.

Reply

StefVan June 14, 2017 at 8:00 AM

Hello Ralph,
Many thanks for this very small sized and efficient library!
Is there any way to make this working with lower baud rates like 10400?
#define BAUD_RATE 10400

Reply

celem September 6, 2017 at 8:26 PM

Ralph, I have no pins left on my ATTiny85 design. I have an LED (to ground through a 1k resistor) on PB1. This LED blinks hourly and also blinks patterns in response to a button press duration. Otherwise it is idle. Looking at your circuit it seems that I could use your circuit connecting your Tx/Rx to the same PB1 that the LED is on but tie the LED to Vcc through the resistor and invert the blink logic. Both functions should work but would obviously interfere if used simultaneously. That is ok with me as they won't be used together. What do you think?

Reply

celem September 6, 2017 at 8:43 PM

Ralph, In looking at the schematic again, my LED connected to Tx/Rx would need to remain tied to resistive ground. The resistor's value is currently 1k to dimly light the LED but when used in conjunction with your circuit that might not be the best value.

Reply

celem September 7, 2017 at 11:37 AM

I breadbord'ed your circuit and installed your example code into an ATTiny85, after switching the Rx/Tx port from PB5 (reset) to PB1. It works although I have a timing issue because I'll often, but not always, have to enter a character several times for it to be received.
BTW - I also tried driving an LED from the same port but I was not successful. Either the LED wouldn't light of the serial didn't work, depending on the LED's resistor value.

Reply

Replies

Ralph Doncaster September 18, 2017 at 6:03 AM

There's two possible problems you might be running into. Your LED may be limiting the signal high level. Blue (and some green) LEDs turn on around 3V, and so would be best. Red LEDs turn on around 2V, which could clip the high signal level so that it is sometimes not recognized.
Another issue you may run into depending on the speed is a poorly calibrated internal oscillator. I have some info about adjusting OSCCAL in this post:
http://nerdralph.blogspot.ca/2014/04/attiny85-as-433mhz-transmitter-fail.html

Reply

KevinOfOz March 26, 2018 at 4:40 AM

Hi Ralph, Did you ever find an opportunity to work on the ISR version?

Reply

Replies

Ralph Doncaster April 1, 2018 at 4:19 PM

I did play with an ISR Rx bit a couple years ago, but never finished. The main use I have found for the bitbang UART is debug output, so I haven't got a good use case and hence much motivation for making an ISR-based receive.

Reply

Unknown September 12, 2018 at 11:12 AM

Hi Ralph,
would it be possible for you to publish this code as Arduino library for inverted serial levels where idle is low? I guess it would not be very difficult for you. I unfortunately do not understand any assembly :(

Reply

Unknown September 12, 2018 at 11:15 AM

Hi Ralph,
unfortunately I do not understand any assenly. Could you publish a version of the library with the serial levels inverted (idle is low)?

Reply

Unknown December 5, 2021 at 2:04 AM

Hi Ralph,

Adding a schottky diode from NPN transistor base to collector transforms it into a schottky transistor. This change made my circuit perform well over 1M baud rate with TTL. This change also reduces the need to iterate through different base resistor value for the specific system.

What the schottky diode does is to provide a feedback loop that prevents the NPN deep into saturation. Low base resistor value can drive NPN into saturation, which significantly increased recovery time from ON state to OFF state.

My application: using a 5V TTL FTDI cable to test drive a serial "smart" servo that uses a 1-wire-like communication bus. when a valid packet sent in over 1M baud 5V TTL, the servo replies a packet in 30 us. I have attempted using the RTS pin as a makeshift GPIO, but turned out I could only flip it every 50 us.

The servo has an EEPROM that can configure response time and baud rate, but with your solution, the servo works out of box.

Best,

Dian

Reply

Unknown February 3, 2022 at 9:34 AM

Hi Ralph,
Revisiting this 8 years later. Fascinating mechanism.
Any thoughts on using a N-MOSFET?

Reply

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)