ad hocumentation • n. fast documentation of ideas and solutions.

Writing bytes to a serial port in C

Posted on February 13, 2013 by batchloaf

This is a (relatively) simple example of a C program to send five bytes to a serial port in Windows. In this case, I'm sending the five characters "hello" via COM22 at 38400 baud, but of course the program can easily be modified to send a different string, or to use a different serial port or baudrate.

This program can be compiled using cl.exe (the C++ compiler used in Visual C++ / Visual Studio) or using gcc (the C compiler in MinGW). In the former case, the file should be saved as "serial.cpp" and compiled as C++, whereas in the latter case, it can simply be saved as "serial.c" and compiled as plain C. I show an example of each approach in the screenshots at the end of this post. Either way, the resulting program "serial.exe" works the exact same.

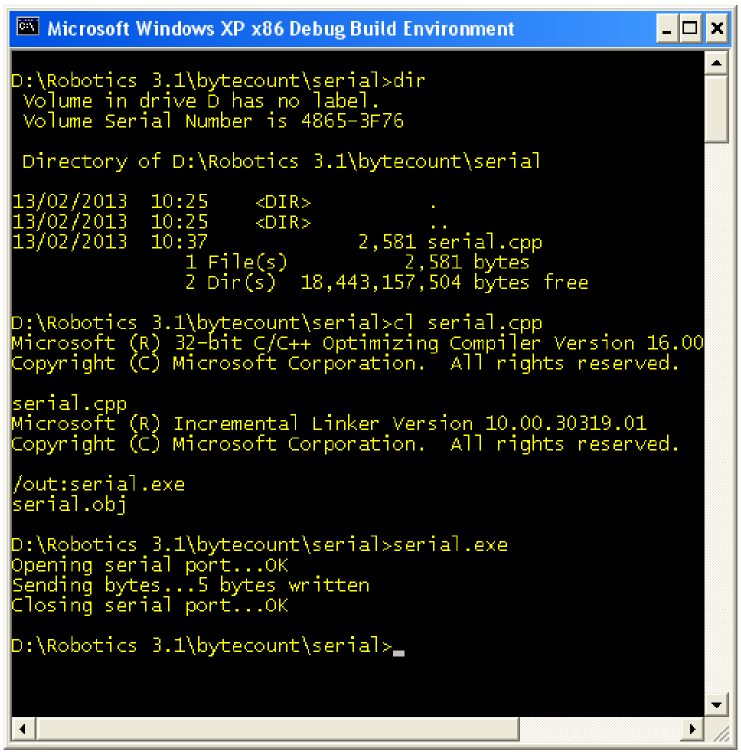
By the way, for a simpler way to send arbitrary characters to a serial port without compiling anything at all, see my previous post.

```
//
// serial.c / serial.cpp
// A simple serial port writing example
// Written by Ted Burke - last updated 13-2-2013
//
// To compile with MinGW:
//
//      gcc -o serial.exe serial.c
//
// To compile with cl, the Microsoft compiler:
//
//      cl serial.cpp
//
// To run:
//
//      serial.exe
//

#include <windows.h>
#include <stdio.h>

```

```
    char bytes_to_send[5];
    bytes_to_send[0] = 104;
    bytes_to_send[1] = 101;
    bytes_to_send[2] = 108;
    bytes_to_send[3] = 108;
    bytes_to_send[4] = 111;

    // Declare variables and structures
    HANDLE hSerial;
    DCB dcbSerialParams = {0};
    COMMTIMEOUTS timeouts = {0};

    // Open the highest available serial port number
    fprintf(stderr, "Opening serial port...");
    hSerial = CreateFile(
                "\\\\.\\COM22", GENERIC_READ|GENERIC_WRITE, 0, NULL,
                OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL );
    if (hSerial == INVALID_HANDLE_VALUE)
    {
            fprintf(stderr, "Error\n");
            return 1;
    }
    else fprintf(stderr, "OK\n");

    // Set device parameters (38400 baud, 1 start bit,
    // 1 stop bit, no parity)
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
    if (GetCommState(hSerial, &dcbSerialParams) == 0)
    {
        fprintf(stderr, "Error getting device state\n");
        CloseHandle(hSerial);
        return 1;
    }

    dcbSerialParams.BaudRate = CBR_38400;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;
    if(SetCommState(hSerial, &dcbSerialParams) == 0)
    {
        fprintf(stderr, "Error setting device parameters\n");
        CloseHandle(hSerial);
        return 1;
    }

    // Set COM port timeout settings
    timeouts.ReadIntervalTimeout = 50;
    timeouts.ReadTotalTimeoutConstant = 50;
    timeouts.ReadTotalTimeoutMultiplier = 10;
    timeouts.WriteTotalTimeoutConstant = 50;
    timeouts.WriteTotalTimeoutMultiplier = 10;
    if(SetCommTimeouts(hSerial, &timeouts) == 0)
    {
        fprintf(stderr, "Error setting timeouts\n");
        CloseHandle(hSerial);
        return 1;
    }

    // Send specified text (remaining command line arguments)
    DWORD bytes_written, total_bytes_written = 0;
    fprintf(stderr, "Sending bytes...");
    if(!WriteFile(hSerial, bytes_to_send, 5, &bytes_written, NULL))
    {
        fprintf(stderr, "Error\n");
        CloseHandle(hSerial);
        return 1;
    }
    fprintf(stderr, "%d bytes written\n", bytes_written);
```

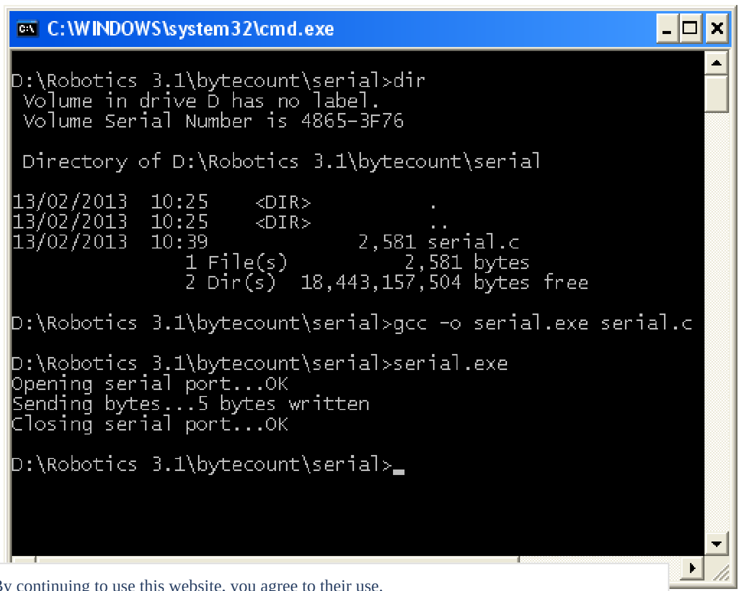
```
    fprintf(stderr, "Closing serial port...");
    if (CloseHandle(hSerial) == 0)
    {
        fprintf(stderr, "Error\n");
        return 1;
    }
    fprintf(stderr, "OK\n");

    // exit normally
    return 0;
}
```

Here's how it looked when I compiled and ran it using the cl compiler (that's the C++ compiler used by Visual C++ / Visual Studio).



```
D:\Robotics 3.1\bytecount\serial>dir
 Volume in drive D has no label.
 Volume Serial Number is 4865-3F76

 Directory of D:\Robotics 3.1\bytecount\serial

13/02/2013  10:25    <DIR>          .
13/02/2013  10:25    <DIR>          ..
13/02/2013  10:37             2,581 serial.cpp
               1 File(s)          2,581 bytes
               2 Dir(s)  18,443,157,504 bytes free

D:\Robotics 3.1\bytecount\serial>cl serial.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 16.00
Copyright (C) Microsoft Corporation.  All rights reserved.

serial.cpp
Microsoft (R) Incremental Linker Version 10.00.30319.01
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:serial.exe
serial.obj

D:\Robotics 3.1\bytecount\serial>serial.exe
Opening serial port...OK
Sending bytes...5 bytes written
Closing serial port...OK

D:\Robotics 3.1\bytecount\serial>_
```

Here’s how it looked when I compiled and ran it with gcc (from MinGW).



```
D:\Robotics 3.1\bytecount\serial>dir
 Volume in drive D has no label.
 Volume Serial Number is 4865-3F76

 Directory of D:\Robotics 3.1\bytecount\serial

13/02/2013  10:25    <DIR>          .
13/02/2013  10:25    <DIR>          ..
13/02/2013  10:39             2,581 serial.c
               1 File(s)          2,581 bytes
               2 Dir(s)  18,443,157,504 bytes free

D:\Robotics 3.1\bytecount\serial>gcc -o serial.exe serial.c

D:\Robotics 3.1\bytecount\serial>serial.exe
Opening serial port...OK
Sending bytes...5 bytes written
Closing serial port...OK

D:\Robotics 3.1\bytecount\serial>_
```

c, send, serial, serial port, string, Windows,

46 Responses to *Writing bytes to a serial port in C*

Abdullah Tahir *says:*

September 19, 2013 at 7:25 am

Thats very fine sir. I am also trying to write data on serial port. But in my code WriteFile() function accepts data only in CString type. I wish to send data in integer format. Can you please help me?

Reply

batchloaf *says:*

September 19, 2013 at 11:07 am

Hi Abdullah,

So, you're programming in C++ using Microsoft Visual Studio or Visual C++ – is that right? If so, perhaps you just need to create a CString object on the fly containing the characters you want to send.

Ted

Reply

Abdullah Tahir *says:*

September 19, 2013 at 6:54 pm

Thank you very much sir, your whole article is very informative.

Abdullah Tahir *says:*

September 19, 2013 at 7:38 pm

And one more thing, this program is to write data onto the COM port, what if I've to read data from COM port? How to do that?

batchloaf *says:*

September 21, 2013 at 4:00 pm

Hi Abdullah,

If you need to receive data from a serial port, it's probably possible to do it using an approach similar to what I describe in the following article for output, but I haven't actually tried that myself:

Simple command line trick for sending characters to a serial port in Windows

What I normally use to receive data from the serial port on the command line is my other program, called ComPrinter – it just opens the specified serial port and prints the incoming characters out on the screen, but they could be redirected into a file as well if required. Here's the link:

https://batchloaf.wordpress.com/comprinter/

Ted

John Campbell *says:*

February 24, 2014 at 4:37 am

This code looks like exactly what I need… BUT. When I try to run your above code in the WIN7 command box it can't find the header files. I guess I need a compiler and code to run in a WIN7 machine.Do you have any ideas for serial port access on a

Reply

batchloaf *says:*

February 24, 2014 at 11:26 pm

Hi John,

You'll need a compiler to compile the above example. Either cl.exe (the Microsoft compiler in Visual C++) or gcc (part of the MinGW software suite) will do fine. Both are available to download for free. You would need to install a compiler whether or not you were using Windows 7.

Depending what you want to send / receive via serial port, it may be overkill using C to do it. Can you explain a little more about what you're trying to do? For really simple serial output, you can sometimes get away with something like this:

Simple command line trick for sending characters to a serial port in Windows

Alternatively, you could use Python to do it, which is what I normally use when I need to quickly automate a bit of serial port chit chat. It's also free to download and most people who use it to solve one problem end up finding it useful for lots of other things, so it's a really useful tool to add to your collection.

Ted

John Campbell *says:*

February 25, 2014 at 1:46 pm

Thanks for the reply.
Well, what I have in mind is writing a small C program to send commands to and receive data back from the RS232 port, and save that data into a file. I have a QBasic program (from the 1980s) that works fine for this task on systems before and through WINXP. Now I have WIN7 and it no longer works.
I did download Microsoft Visual Studio, which I suspect will work, but it seems like huge overkill and the user documentation is awful – I can't even figure out how to do "Hello World" in it!
Visual Studio does come with CL but you can't use it in a Command Window.
Just as an aside – Ii seems really silly to me that Microsoft does not provide a simple C compiler with every OS installation!

Thanks,
John

batchloaf *says:*

February 26, 2014 at 11:50 am

Hi John,

Based on your description and your previous use of QBASIC, I definitely recommend looking at Python to solve your problem. Have a look at this comment I wrote to Wim Bruyn who was trying to do some serial comms:

https://batchloaf.wordpress.com/serialsend/#comment-2452

If you decide to give it a shot with Python, I'm sure you'll get it sorted out. I'll be happy to lend a hand if you get stuck. Also, if you're not sure where to get started, you can send me the QBASIC program and I'll see if it can be converted to Python easily.

Ted

Dorian *says:*

is very educational since this was the first time I compiled anything with Visual Studio. The experience has led me to wonder what is behind the failure of serial commands from Octave.

I have a serial device that looks for a simple 4-character command, e.g. 'abcd' to change the kind of data it sends. With the python client associated with the device the commands can be successfully sent via keyboard input. Using Realterm with chardly=100ms I can also successfully send the commands. Using fread in Octave for data analysis I was able to receive the serial data but had no success with fprintf or fwrite to send the commands, even with pausing between characters. Why? Then I came across your simple command line trick but it seemed to hang the device. I was disappointed and mystified. Then I came across your serial program and adapted it, after about a dozen re-compiles as I learned C program syntax, to write one character at a time with a delay in between. And it worked !

I can next try to run serial.exe as a system command from within Octave, but I am wondering if you or anyone know why fprintf or fwrite doesn't work on the commands but fread does fine on the receive. It would be great if the command line trick worked; it could be a good patch to provide Octave full serial data functionality. Matlab apparently handles serial data with fprintf. Octave is open source so it may be possible to spot the issue in the source code but as I said I am just a beginner.

Reply

batchloaf *says:*

September 29, 2013 at 10:19 pm

Hello again Dorian,

I'm not too sure about using fwrite etc in Octave since I haven't tried it before. However, I'm interested to see if there's a straightforward solution since I'm back teaching programming in MATLAB this semester and I encourage my students in that class to install Octave if they want to practice at home. If I get a chance during the week, I'll install Octave and try to sending out a few characters via the serial port with fwrite.

I don't really know why it's not working for you, but one thing did spring to mind that you should try if you haven't already. The input and output streams for serial port devices are typically buffered by default. In other words, when you fwrite characters to the serial port, they're not normally transmitted immediately via the serial port. Instead, they're written to a memory buffer which acts as a kind of staging area for transmission. Sometimes, they're shuffled out of the buffer one-by-one without any delay, but more often they just sit there waiting in the buffer until a newline character ('\n') is written to the serial port (i.e. the end of each line of text in the transmission). If you're only sending the four characters you mentioned, there's probably no newline character at the end (and you don't actually want to send one), so you may need to explicitly "flush" the output buffer to get the characters stored there moving out over the serial link. To flush the output stream, just use the fflush function, as follows:

```
fflush(fid)
```

where fid is the file descriptor for the serial port device, which you presumably already used in your fwrite function call.

Hope that helps!

Ted

Reply

Markus *says:*

November 5, 2013 at 3:02 pm

Very good intro, and it works!

Reply

batchloaf *says:*

November 5, 2013 at 4:10 pm

Ted

Reply

Reddy Parvathi *says:*

December 30, 2013 at 5:04 am

Hello everyone , its good………….. I need to write and read a file using serial port ,can you please explain me a sample code for creating, opening, writing, reading and closing a file using serial port .

Reply

batchloaf *says:*

January 1, 2014 at 6:39 pm

Hi Reddy,

Sorry, I don't understand exactly what you want to do. Do you mean that you want to read data from a device connected to a serial port, store that data in a file, and then later on retrieve the data from the file and transmit it back out through the serial port?

Ted

Reply

parvathi *says:*

January 17, 2014 at 7:05 am

yes, you are correct. Its my exact question

yes i want to read data from a device connected to a serial port, store that data in a file, and then later on retrieve the data from the file and transmit it back out through the serial port?

batchloaf *says:*

January 17, 2014 at 10:53 am

And is the length of the data to be recorded fixed, or is there some other way to know when all the data has arrived so that the PC can start re-transmitting it?

Sergio Davila *says:*

September 12, 2014 at 12:04 am

Hi!

I'm trying to run this example in Eclipse (Keppler) on Windows with MinGW.

However, I am getting a lot of undefined symbols, beginning with "HANDLE."

Is there an include file that I am missing? Perhaps my versions of and do not have these definitions (or the includes that they reference are different).

Thank you!

Reply

batchloaf *says:*

September 12, 2014 at 12:41 am

If you've included the header files I included then that should be adequate. However, you could still receive an error of that type from the **linker** (rather than the *compiler*). Maybe you know C/C++ inside out, in which case that's probably enough to explain what I think the problem is. However, in case you're not (or you're somebody other than Sergio reading this reply), I'll try to explain what I mean.

When you compile a C program, each C file (e.g. "main.c", "blah.c", etc) forms the basis of a separate "module". The module consists of not only what's in the C file itself, but also whatever declarations are hauled in by the header files included in the C file. These declarations are mostly just prototypes for library functions and that kind of thing, and they are normally declared as `"extern"`, which tells the compiler that these functions *exist* (along with the data type of the arguments and return value) and that the linker should be able to track down the actual definition of each function later on. In other words, the extern declaration says something like: *"This function exists but is defined somewhere else so don't bother looking for it just yet. For the time being, all you need to know is how to call it."*

So, in the meantime, it's ok for the compiler to compile each module of the program in isolation to create an *object* file. An object file contains compiled code – i.e. it's a non-readable (by humans) binary file – but it's not an executable program. It normally has a file extension like `.o` or `.obj`.

Once all of the program's C files have been compiled, the compiler is left with a bunch of these object files. At this point, the linker takes over. The role of the linker is to combine the separate object files into a single working executable, by resolving any unresolved references in the object files and connecting each such reference to the correct point in another object file or in a library file (e.g. a file with the `.lib` extension).

Compiling and linking are regarded as two separate steps in the build process and many C compilers use two separate tools to perform the two steps. In the case of GCC however, both steps are intitated by running the same `gcc` executable. Depending on the what command line arguments are specified, gcc can either compile or link or both. Similarly, `cl.exe` which is the C compiler under the bonnet of Visual C++ both compiles *and* links.

The linker's input files are the object files and library files, rather than the `.c` and `.h` files, which are no longer needed at this point in the build process. Therefore, if there error you're seeing is generated by the linker, then header files are not the problem. Instead, it's probably due to the linker options not specifying all of the required libraries. If you're compiling at the command line, this would normally be done using linker flags in the command line arguments.

In the case of this program, if you do need to explicitly link a specific library, my money is on "user32.lib" which is required for compiling most Win32 platform Windows programs. When I run gcc (from MinGW) on the command line, my understanding is that "user32.lib" is linked automatically, but if that's not the default behaviour in Eclipse, then you could dig around for a menu option to specify it manually. Alternatively, if Eclipse allows you to edit the compiler command line, you could try adding the following argument to it: "-luser32". That's a minus sign, followed by "l" for lemon, followed by "user32".

A complete command line might look like this:

```
gcc -o serial.exe serial.c -luser32
```

Hope that helps!

Ted
Reply

Sergio Davila *says:*
September 12, 2014 at 4:31 pm

Thank you for your very prompt reply!

I believe Eclipse is actually issuing compiler errors. I don't think it gets to the stage of linking in libraries. In fact, in the IDE, "HANDLE" and other identifiers ("DCB","COMMTIMEOUTS","GENERIC_READ",etc) are flagged (red underlined) as missing.

he identifiers in WinBase.h and other includes), so your code works great. The only modification I had to make was to change CreateFile to CreateFileA (because in my configuration CreateFile is redefined to CreateFileW which uses a LPWCSTR instead of a LPCSTR for the first argument — don't ask me why or what that means!)

However, my reason for using Eclipse is that the next step will be to run it under linux on a virtual box (as an intermediate step itself towards using it in an embedded ARM processor).

Again, thank you for your help!

batchloaf *says:*
September 12, 2014 at 11:38 pm

Hi Sergio,

I'll come back to the undefined symbols in a minute, but before that… the CreateFileA vs CreateFileW thing:

Somewhere in the project settings dialog box in Visual C++ (or perhaps in the Project Wizard when you create a new project), there is a check box to set it as a Unicode project. If the program uses Unicode by default, that means that each character in a text string anywhere in the program will be represented by more than one byte. This is because Unicode provides a lot more characters than the upper and lower case alphabet and punctuation marks we have on our keyboard. These multi-byte characters are sometimes referred to as *wide characters*.

Microsoft define all kinds of data types for different things (I'm not sure why), including LPCSTR which I think is a long pointer to a C string. In other words it's the memory address (pointer) of an array of characters (a C string) which can contain some text. Historically, a string of this kind would have consisted of an array of single-byte characters. Nowadays however, a string could contain multi-byte unicode characters, in which case the type LPWCSRT would be used instead. It's basically the same thing as LPCSTR – the memory address of an array of text characters – but the individual characters in the string are *wide characters* (multi-byte unicode). Hence the addition of the letter "W".

Because a lot of Microsoft's Win32 functions use text strings as arguments and return values, they created a second version of every function that used unicode strings rather than single-byte ascii strings. However, they made it so that you can still use the same function name as always and the appropriate version of the function (unicode or non-unicode) will be used according to your project settings. For example, let's suppose you use the function `CreateFile`. If your project is set to use unicode, the function CreateFileW will actually be used and things like the filename will be specified as unicode strings. If the project is not set to use unicode, CreateFileA will be used instead and the filename will be specified as a string of single-byte ascii characters.

I'm afraid I can't be sure why you're getting those undefined symbol errors. Could you post the exact text of the errors you're getting so that I can google them?

Ted

Adi *says:*

December 22, 2015 at 10:30 pm

Hey Ted,

Thanks for this amazing blog. I tried out this code, and even though everything compiles correcly, I get an error while opening port. I tried it with several different port numbers, but the rror persisits and the program closes. Any Ideas?

Adi

Reply

batchloaf *says:*

What kind of serial port are you writing to – is it a USB-to-serial adapter of some kind?

What's the error you're getting?

Are you able to connect to the serial port from another program?

Ted

Reply

grottyBits *says:*

February 20, 2016 at 4:04 pm

I'd like to send/receive bytes with an adruino over a USB cable.
I was hoping to write to it using your sample code. I am able to compile it, but CreateFileA() always fails. "System can not find the file

specified"
The filename I'm passing in is "\\\\.\\COM22"
Reply

batchloaf *says:*
February 21, 2016 at 12:12 pm

Hi grottyBits,

It sounds like your serial device is showing up as a different numbered COM port. My one just happened to show up as COM22 when I plugged it in, but that number won't stay the same (especially between different devices and different computers). If you haven't already done so, please check what COM port number your Arduino is appearing as. You might be able to see this from the menus in the Arduino IDE software. Failing that, you could try using my ComPrinter program which automatically scans for the highest numbered COM port present on the system. Once you know the correct COM port number you can change the code example above and hopefully it will work.

Ted

Reply

Ravish *says:*
March 11, 2016 at 12:50 pm

Hello Sir,
A very informative post indeed. Thanx a lot for that at the outset.

I am stuck at a problem and am very new to the world of programming. As a part of a project, I have to receive data from a sensor with only one way communication on RS 422 port.
The packets being sent are 81 byte long and are structured as follows:

Byte Parameter Reference

1-2 5Ah A5h Header
3 48h NUMDATA
4 02h IDENT

5-9 STATUS 1-5

10-12 Time

16-18 Data2

19-21 Data3
.
.
.
.
80 XXh CHECKBYTE

81 AAh Terminator

Now, I have to receive these data packets, take the data from data field and arrange it in an array.

I have been able to achieve serial communication between my 2 PCs using the following code :

#include
#include

```
#ifdef _WIN32
#include
#else
#include
#endif

#include “rs232.h”
int main()
{
int i, n,
cport_num=0, /* /dev/ttyS0 (COM1 on windows) */
bdrate=9600; /* 9600 baud */

unsigned char buf[100000];

char mode[]={‘8′,’N’,’1′,0};

if(RS232_OpenComport(cport_num, bdrate, mode))
{
printf(“Can not open comport\n”);

return(0);
}

while(1)
{
n = RS232_PollComport(cport_num, buf, 4095);

if(n > 0)
{
buf[n] = 0;

for(i=0; i < n; i++)
{
if(buf[i] < 32) /* replacing unreadable control-codes by dots */
{
buf[i] = '.';
}
}
printf("received %i bytes: %s\n", n, (char *)buf);
}
```

```
#else
usleep(100000); /* sleep for 100 milliSeconds */
#endif
}
return(0);
}
```

With this code am able to receive the text sent over on comport by following program :

```
#include
#include

#ifdef _WIN32
#include
#else
#include
#endif
```

#include “rs232.h”
int main()
{
int i=0,
cport_num=0, /* /dev/ttyS0 (COM1 on windows) */
bdrate=9600; /* 9600 baud */

char mode[]={‘8′,’N’,’1′,0},
str[2][512];
strcpy(str[0], “Transmission 1\n”);

strcpy(str[1], “Transmission 2\n”);
if(RS232_OpenComport(cport_num, bdrate, mode))
{
printf(“Can not open comport\n”);

return(0);
}
while(1)
{
RS232_cputs(cport_num, str[i]);

printf(“sent: %s\n”, str[i]);

#ifdef _WIN32
Sleep(1000);
#else
usleep(1000000); /* sleep for 1 Second */
#endif
i++;
i %= 2;
}
return(0);
}

How should i go ahead with this background to receive the 81 byte packet and arrange the 63 byte data field in it in an array?

Regards
Reply

Hi sir,
Your post is very much useful thanks for it. And i need to add an interrupt to the receiving data to the pc. ie, for the ReadFile data in the program. How can i add the interrupt to this data.
Reply

batchloaf *says:*
May 21, 2016 at 4:38 pm

Hi Satheesh,

I guess it’s possible to generate some kind of interrupt or trigger a callback function when new data arrives into the serial buffer of the PC, but I’ve always just used polling myself which seems simpler. You can set up a Win32 timer (using the SetTimer function) to check the serial buffer for data periodically and do something with the data if it finds any waiting. To check if there’s data waiting in the serial input buffer, you can call the Win32 function ClearCommError passing in a pointer to a COMSTAT structure. ClearCommError will fill the COMSTAT structure with information about the current state of the serial port, including the number of bytes received which have not yet been read (in the cbInQue member of the COMSTAT structure).

I know that's not quite what you were looking for, but hopefully it's of some assistance. That's how I would try doing it anyway.

Ted
Reply

sadasir *says:*
July 28, 2016 at 3:58 pm

Hello,
I get this error when I run the code:
Error C2664 'HANDLE CreateFileW(LPCWSTR,DWORD,DWORD,LPSECURITY_ATTRIBUTES,DWORD,DWORD,HANDLE)': cannot convert argument 1 from 'const char [10]' to 'LPCWSTR' \consoleapplication1.cpp 26.
Do you know how can I fix it?

Thanks, great post.

Reply

batchloaf *says:*
July 28, 2016 at 4:24 pm

Hi Sadasir,

I'm guessing you may be compiling in Visual C++ – is that correct? This sounds like one of those weird problems related to the Unicode or non-Unicode versions of the same function. For example, when you use a Win32 function like CreateFile, you're actually using one of two possible versions of the function – a unicode version called CreateFileW or a non-unicode (i.e. ANSI) version called CreateFileA. Which one your program ends up using depends on the project settings you've chosen.

Anyway, to cut a long story short, CreateFileW expects the filename to be provided as a unicode string but you're currently providing it a plain old C string. Try changing the first argument from `"\\\\.\\COM22"` to `L"\\\\.\\COM22"` and recompiling. Adding the letter "L" before the inverted commas tells the compiler to make it a unicode string. So the complete line of code is..

```
hSerial = CreateFile(L"\\\\.\\COM22", GENERIC_READ|GENERIC_WRITE, 0, NULL,
                     OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL );
```

Hopefully that will solve it. Please let me know how you get on.

Ted
Reply

Jim *says:*
November 16, 2016 at 4:18 pm

Caveman Ted,

Thank you for this post. I have need to communicate with my Arduino to gather data from sensors and log the data for analysis. I needed a working example that I could modify to fit my needs. Your program almost worked out of the box on my Windows 10 machine. I had to insert the com port number of my Arduino, and it ran. I was pleased that this took less time than I expected.

I changed the baud rate and added a ReadFile loop, and it's over there collecting data. You saved me a grundle of time. Thank you.

Jim

Reply

batchloaf *says:*

November 16, 2016 at 9:30 pm

Thanks Jim, I'm delighted you found it useful!

Ted

Reply

Usman *says:*

January 2, 2017 at 1:10 pm

Kindly share a code, for reading the COM Port also, Thanking in Advance.

Reply

Prash *says:*

January 5, 2017 at 4:14 pm

Hi Ted,

How can we send binary characters on serial port i.e presently the values get converted to ASCII when they are out on the serial port. For example when I want to transmit 0x08 from the program it transmits 0x38 to a microcontroller board. Can you please help suggest !!

Reply

batchloaf *says:*

January 5, 2017 at 7:19 pm

Hi Prash,

I'm not sure why your microcontroller is not receiving the bytes you expect, but I don't think there's any way that my example program above can be converting the values to ASCII. There's just nothing in the program that could do that.

All I can think of is that maybe you're writing something like this in your version of the program…

```
bytes_to_send[0] = '8';
```

That would result in the byte value 0x38 being sent.

If all else fails, you could try using my little utility program SerialSend to send the bytes you want. It's what I normally use for sending simple binary values to a microcontroller. It's free to download from here (link also on the menu bar above):

https://batchloaf.wordpress.com/serialsend/

Hope that helps.

Ted

Reply

batchloaf *says:*

January 5, 2017 at 7:21 pm

PS If you have something like “0x08” or ‘0x08’ in your code somewhere that might produce a four-byte block that is somehow being transmitted. The last of the four bytes would be 0x38 (the ASCII value for the character ‘8’).

Adarsh *says:*
May 31, 2017 at 10:37 pm

Hi,

I’m working on communicating with a device using a USB to RS232 cable. I am able to communicate with it perfectly using powershell and the ReadLine(), WriteLine() commands. Using
the program you have generously provided, I am unable to achieve the same results. WriteFile works fine. It returns success and the number of bytes written is what is desired. However, the device unfortunately does not seem to respond as it does with powershell. I have to send the commands in ASCII but I believe that using sprintf(write_buffer, “Hello”) should work.

I have checked the Baudrate, parity bit, Bytesize parameters and they match the device settings. I also tried the ‘simple command line trick’ post that you put up and found out that the device fails to acknowledge commands sent through it too.

Do you have any suggestions on what else I can try? I’d appreciate them very much.

Reply

batchloaf *says:*
May 31, 2017 at 10:44 pm

Could it be something to do with a newline character that’s missing at the end of the string you send?

Ted

Reply

Adarsh *says:*
June 1, 2017 at 4:36 pm

This was exactly it! Microsoft uses CR LF to signify EOL and so I had to append “\r\n” to the end of my strings. Thank you so much for this resource!

Another question I have is about reading. I want to write a CLI for this device but with ReadFile unfortunately, I have to read the response one character a time. I want to write my command to the device with one call and use something like fgets on the device so that I can get the entire response with one call. Would you be able to tell me how to get the associated File *stream or if there are other ways to get the entire contents of the response in one call?

I changed the ‘nNumberOfBytesToRead’ parameter from 1 to MESSAGE_LENGTH and this worked.

In case anyone else is looking, the following is also a good resource:
https://msdn.microsoft.com/en-us/library/windows/desktop/aa365467(v=vs.85).aspx

Again, thanks for this page!

batchloaf *says:*
June 8, 2017 at 4:47 pm

You’re welcome Adarsh! Glad you got your problem solved.

Ted

Rene Rutten *says:*

March 12, 2018 at 1:34 pm

🙂 Works seamless after 1st Compile. Good job.

Reply

batchloaf *says:*

March 15, 2018 at 11:18 am

Thanks Rene!

Ted

Reply

Alias *says:*

December 3, 2019 at 8:53 am

E0167 argument of type "const char *" is incompatible with parameter of type "LPCWSTR" WindowsProject4
hSerial = CreateFile(
"\\\\.\\COM22", GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);

Reply

Alias *says:*

December 3, 2019 at 8:55 am

E0167 argument of type "const char *" is incompatible with parameter of type "LPCWSTR" WindowsProject4
hSerial = CreateFile(
"\\\\.\\COM22", GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
Can you please suggest a solution to rectify this error? Thanks a lot.

Reply

batchloaf *says:*

January 23, 2020 at 3:38 pm

Hi Alias

[illegible]her or not unicode strings are used. As [illegible]acter string". Wide character here refers to the fact that each character of the string is stored using multiple bytes rather than just a single byte. If you're compiling in Visual Studio, I think there's a setting where you can enable/disable unicode support for the whole project and depending whether it's on or off functions which expect strings as arguments will expect either "LPCSTR" or "LPCWSTR". Maybe try changing the unicode setting for the project and see if the error goes away?

Ted

Reply

ad hocumentation • n. fast documentation of ideas and solutions.