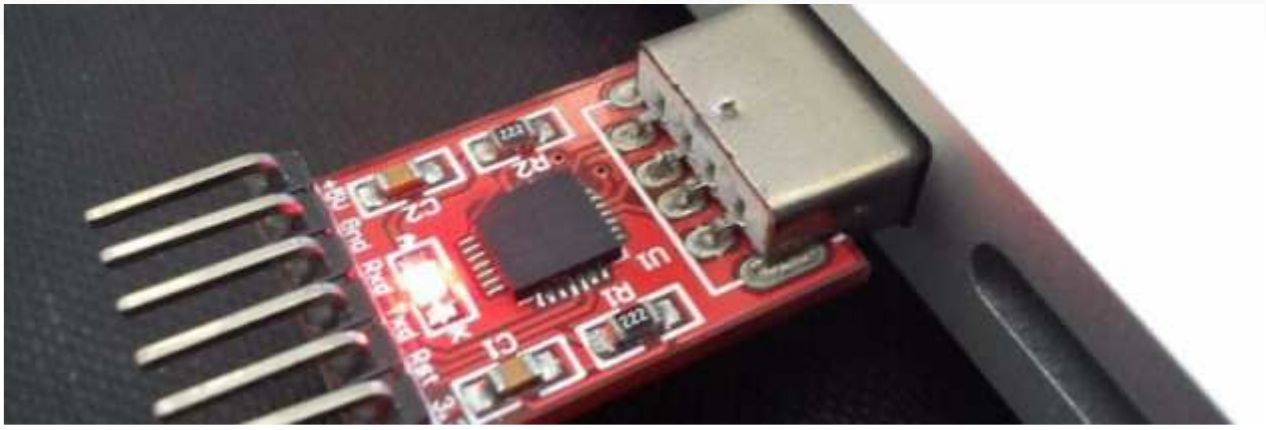


Serial Communication – RS232 Basics

In the previous post , we discussed about the basics of serial communication. In this post, we will learn about the RS-232 protocol of serial communication. This is the protocol you will be using the most when involving microcontrollers like AVR. As we proceed ahead in this post, we will deal with the concept of level conversion and towards the end, we have something interesting and practical for you – the loopback test!

RS-232 Basics

RS-232 (Recommended Standard – 232) is a standard interface approved by the Electronic Industries Association (EIA) for connecting serial devices. In other words, RS-232 is a long established standard that describes the physical interface and protocol for relatively low-speed serial data communication between computers and related devices. RS-232 is the interface that your computer uses to “talk” to and exchange data with your modem and other serial devices. The serial ports on most computers use a subset of the RS-232C standard. RS-232 protocol is mostly used over the DB9 port (commonly known as *serial port*), however earlier it was used over the DB25 port (also known as *parallel port*). We will have a look at both of them here.

RS-232 over DB-9

The pin configuration of DB-9 port is as follows. Yes, it looks exactly like (in fact it is) the serial port you would find in older computers.

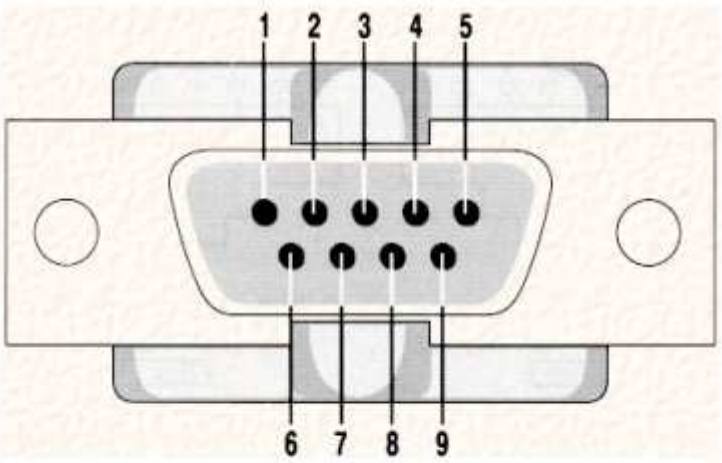


Image Courtesy DB9 Connector
The 8051 Microcontroller and Embedded Systems by Mazidi and Mazidi

RS232 DB9 Connector

Pin	Description
1	Data carrier detect (-DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (-DSR)
7	Request to send (-RTS)
8	Clear to send (-CTS)
9	Ring indicator (RI)

Courtesy DB9 Connector Pins
The 8051 Microcontroller and Embedded Systems
by Mazidi and Mazidi

DB9 Connector Pins

The pin description for the RS-232 pins is as follows:

- **DTR** (data terminal ready): When terminal is turned on, it sends out signal DTR to indicate that it is ready for communication.
- **DSR** (data set ready): When DCE is turned on and has gone through the self-test, it assert DSR to indicate that it is ready to communicate.
- **RTS** (request to send): When the DTE device has byte to transmit, it assert RTS to signal the modem that it has a byte of data to transmit.

- **CTS** (clear to send): When the modem has room for storing the data it is to receive, it sends out signal CTS to DTE to indicate that it can receive the data now.
- **DCD** (data carrier detect): The modem asserts signal DCD to inform the DTE that a valid carrier has been detected and that contact between it and the other modem is established.
- **RI** (ring indicator): An output from the modem and an input to a PC indicates that the telephone is ringing. It goes on and off in synchronous with the ringing sound.
- **RxD** (Received data): The RxD pin is the Data Receive pin. This is the pin where the receiver receives data.
- **TxD** (Transmitted data): The TxD pin is the Data Transmit pin. This is the pin through which data is transmitted to the receiver.
- **GND:** Ground pin.

RS-232 over DB-25

The pin configuration of DB-25 port is as follows. Yes, it looks exactly like the parallel (printer) port that you would find in older computers! It is however worth noting that the DB-25 port could transfer data either serially (RS-232) or parallely. The pinouts are different in each case. The pinout for RS-232 over DB-25 is shown below (thanks Thomas and Aaron for clarification).

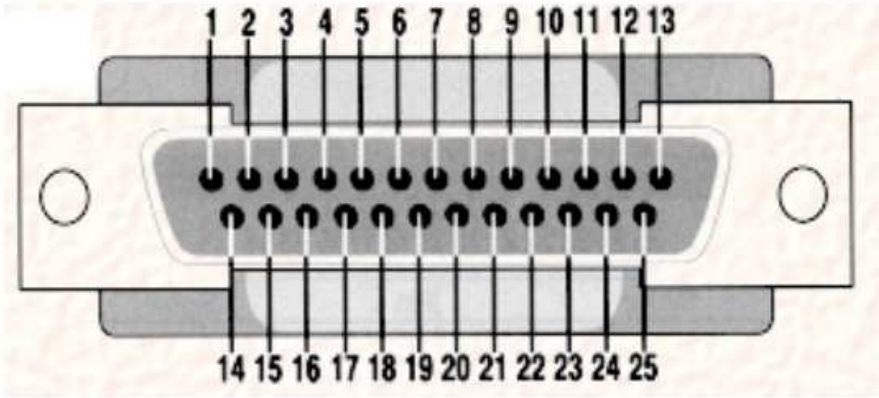


Image Courtesy
The 8051 Microcontroller and Embedded Systems by Mazidi and Mazidi

DB25 Connector

RS232 DB25 Connector

Pin	Description	Pin	Description
1	Protective ground	14	Secondary transmitted data
2	Transmitted data (TxD)	15	Transmitted signal element timing
3	Received data (RxD)	16	Secondary receive data
4	Request to send (-RTS)	17	Receive signal element timing
5	Clear to send (-CTS)	18	Unassigned
6	Data set ready (-DSR)	19	Secondary receive data
7	Signal ground (GND)	20	Data terminal ready (-DTR)
8	Data carrier detect (-DCD)	21	Signal quality detector
9/10	Reserved for data testing	22	Ring indicator (RI)
11	Unassigned	23	Data signal rate select
12	Secondary data carrier detect	24	Transmit signal element timing
13	Secondary clear to send	25	Unassigned

Courtesy
The 8051 Microcontroller and Embedded Systems by Mazidi and Mazidi

DB25 Connector Pins

DB25 Connector Pins

As we can see, most of the pins are similar to that of a DB9 port. If you notice, we see that in DB25 connector there are **two** TxD and RxD pairs of pins. Now what does this mean? In simple words, it means that serial communication through the DB-25 connector could take place through two channels simultaneously (thanks again, Thomas!).

NOTE: Another important thing to note is that the simplest way to in which a microcontroller can communicate to a PC is through RxD, TxD, and Ground Pins. And this is what we will be doing here and hence forth in upcoming posts. The other pins are not of much use to us, for now. Now this was something about RS-232. Our next topic is level conversion. Btw, have you heard of TTL? Sounds familiar, but what is TTL? Lets read on!

Logic Level Families

By 'Logic Level' one means the range of voltage over which a high bit (1) and a low bit (0) is accepted in a particular IC, gate, etc. Various logic levels have been standardized, out of which the most popular ones are:

1. TTL

TTL stands for T ransistor- T ransistor L ogic. These days TTL is the most widely used logic. TTL is mostly used in ICs and gates, like 74xx logic gates. A major drawback of the TTL logic is that most of the devices working on the TTL Logic consume a lot of current, even individual gates may draw up to 3-4 mA. In TTL Logic, a HIGH (or 1) is +5 volts, whereas a LOW (or 0) is 0 volts.But since attaining exact +5 volts and 0 volt is practically not possible every time, various IC manufactures define TTL logic level range differently, but the usual accepted range for a HIGH is within +3.5 ~ +5.0 volts, and the range for a LOW is 0 ~ +0.8 volts.

2. LVTTL

LVTTL stands for L ow V oltage T ransistor- T ransistor L ogic. LVTTL is increasingly becoming popular these days, because of the nominal HIGH voltages, and hence lesser power consumption. By lowering the power supply from 5v to 3.3v, switching power reduces by almost 60%! There are several transistors and gates, which work on LVTTL logic. Atmel's Atmega microcontrollers are designed to work on both, LVTTL and TTL, depending upon the V_{CC} supplied to the IC. In LVTTL Logic, a LOW is defined for voltages 0V ~ 1.2V, and High for voltages 2.2V ~ 3.3V, making 1.2V~2.2V undefined.

3. RS-232

RS232 is also one of the most popular logic. Though now quite old, it is still in use. In RS-232 logic, a HIGH (1) is represented within -3V ~ -25 V, whereas a LOW (0) is in between +3V ~ +25 V, making -3V to +3V undefined. Weird isn't it? ;) But that's how it is defined! Apart from these, there are many other logic families like ECL, RTL, CMOS, LVCMOS, etc. At present, we are not much concerned about them. You can refer to this article to know more about them.

Level Conversion – TTL/RS232

So what is (logic) level conversion?? To interconnect any two logic level families, their respective HIGHs and LOWs must be same else they wouldn't work. For example, when we want to interconnect two devices, one of which works over TTL and the other over RS232, we need to convert the HIGH of TTL (which is 3.3v~5v) into the HIGH of RS232 (which is -3v ~ -25v) and similarly, the LOW of TTL (0v~0.8v) into the low of RS232 (which is +3v ~ +25v). So you see, here lies the problem! If we do not convert the logic levels (in this case) then the LOW signal of TTL would be interpreted as a HIGH in RS232, making all the data transfer go wrong!!

The Solution

One solution is to use additional pull-up resistors, or to use Zener diodes. A better solution is the use of ICs that directly converts logic levels. Luckily logic level conversion is quite simple these days with the use of ICs like MAX232 and CP2012! We would talk about all these solutions one by one.

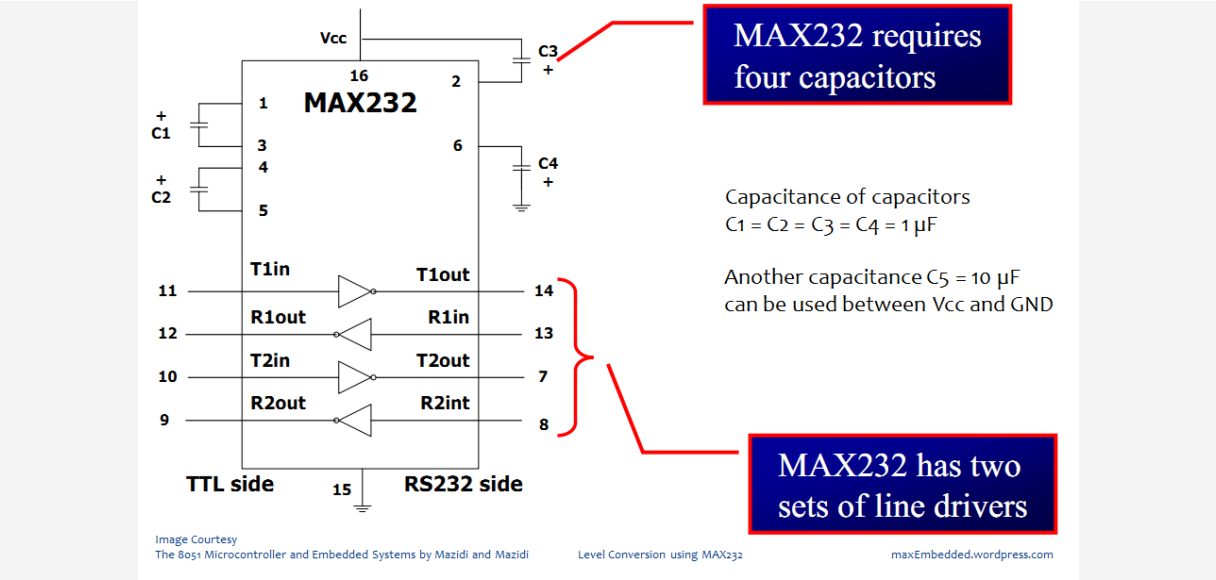
Zener Diodes

Zener diodes are widely used to regulate voltage between two points.When Zener diode is used in reverse bias in series with a suitable resistor, and a voltage > breakdown voltage is applied across the terminals of the Zener-resistor pair, then a voltage V = Zener Voltage appears across the terminals of the Zener diode, while the rest of the voltage appears across the terminals of the resistor. The simple circuit below shows how to use Zener diodes to convert logic levels from TTL to LVTTL (Note that this circuit is only applicable for high to low logic level conversions):

Bidirectional Logic level converters are easily available in the market. Some of the websites selling them are: Adafruit , Freetronics , Sparkfun ,Embedded Market etc.

IC MAX232

MAX232 ICs were invented by *Maxim.* These IC packages are used to convert TTL/CMOS logics to RS232 logic directly! All we need are some passive components, and we are done! Below is the circuit diagram of the MAX232 IC.



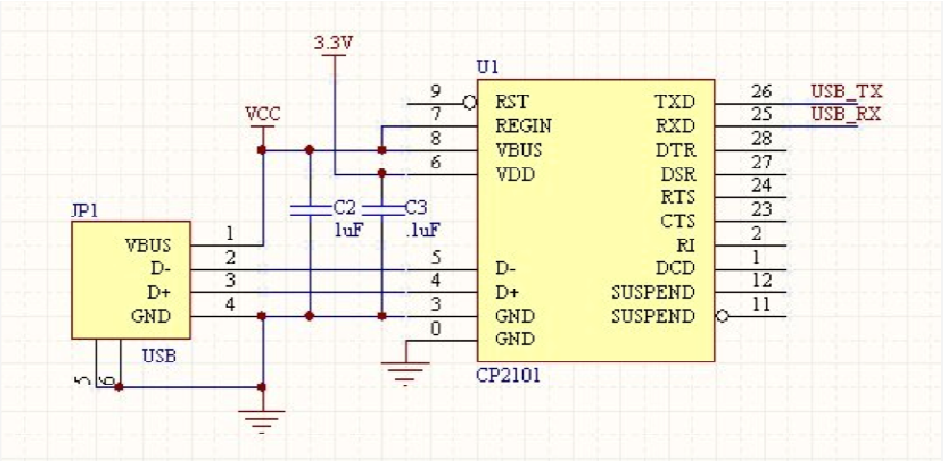
Level Conversion using MAX232

MAX232 is used to convert TTL to RS232, and vice-versa as shown in the above circuit diagram. But these days, USB is the most used protocol!Everything runs on USB – be it printer, scanner, displays or anything! But how to convert USB to UART? One way is USB → TTL → UART. The other way is to use USB-UART bridges which directly convert USB → UART. They are widely and easily available these days. Here are some of the websites:Robokits , eXtreme Electronics , Sparkfun , Adafruit etc.

All these devices work on CP210x based USB-UART conversions.

IC CP210x

CP210x is the series of ICs made by Silicon Labs. These are used to directly convert USB to UART. Below is the circuit schematic of CP2101:



CP2101 Schematics

Though these ICs are not available in DIP Packages, so it is always advisable to buy any one of the modules listed above.

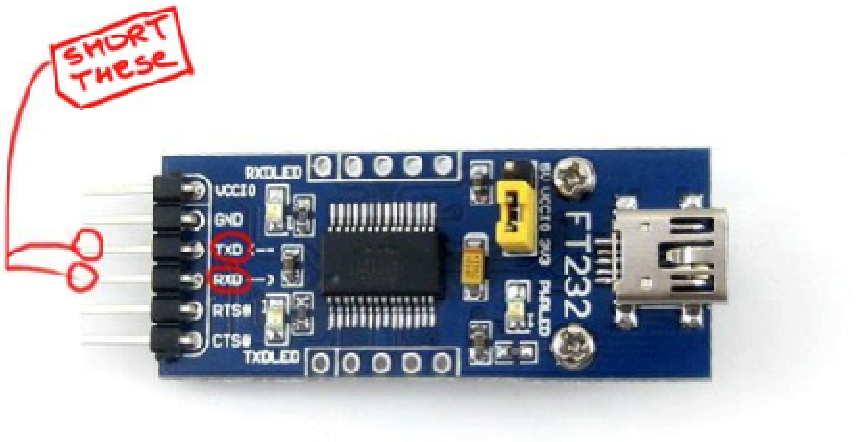
This is how a CP210x based USB-UART Bridge looks like

The drivers of CP210x can be found here . They work with the Windows platform. Drivers for Mac and Linux are also available. We will discuss a little later in the same post as to how to install these drivers and work with them.

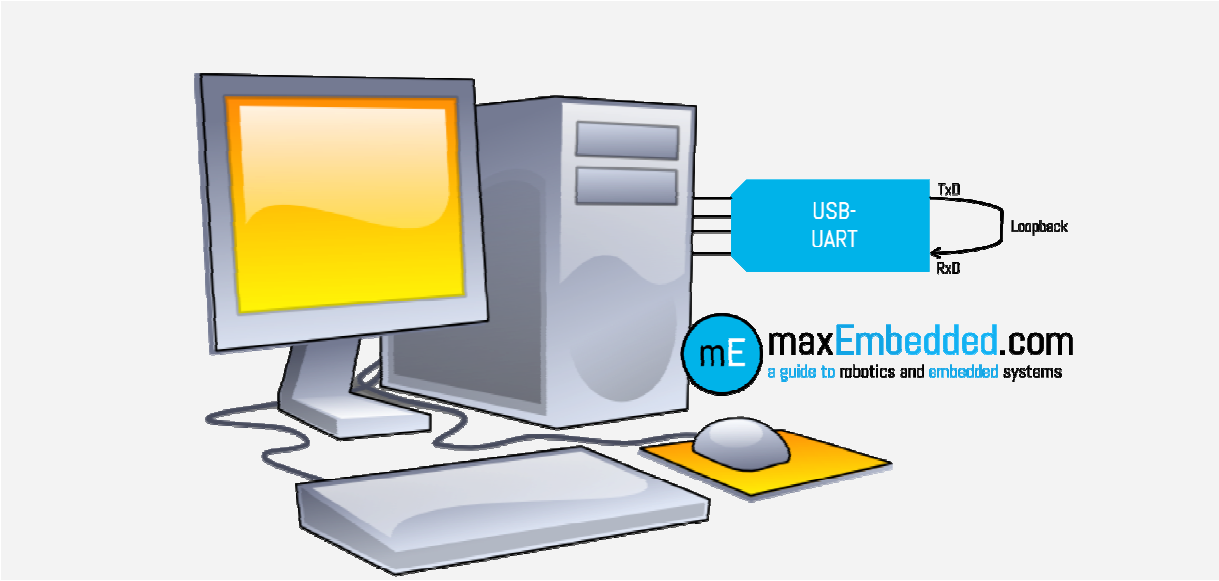
This was all about what is logic level conversion, why we need it, and how to do it. Now its enough of theoretical topics, lets have something practical stuff to do now! Next, we deal how to use the serial communication devices with your PC. So get your PC and USB-UART bridge ready! That's all you need.

Loopback Test

A Loopback Test in serial communication is a test in which we check whether the Rx and Tx are working properly or not. How we do it? Its simple! Just short the Tx and Rx pins of the USB-UART Bridge and connect it to your PC! So it's like transmitting data from your PC and receiving the same through the same port! Have a look at the following block diagram.



Loopback Test Setup for USB-UART Bridges



Loopback Test Block Diagram

Using a Serial Terminal at PC/Mac

So far so good. You have made the hardware connections. Now what? How do you send serial data from your PC (via USB) to any other device? Well, seems like now is time to deal with serial terminals! What is a serial terminal?

A serial terminal is an application, which allows you to directly control the serial ports of your PC/Mac ie it lets you send and receive data directly through your serial port. Bingo!

For various applications, it is best to have a serial terminal. For example, you want to communicate to an AVR microcontroller, or say you wanna control a microcontroller through your PC (a cool example would be control a robotic car wirelessly using your computer, it would be like real world NFS! We organized one such event last year as well !!), so how would you do that? Yes exactly! You will use a serial terminal on your PC!

Another example could be when you want to use a module, say for example, a GSM module, or a Bluetooth module with your PC. You can communicate with those modules only with the help of these serial terminals!

A number of serial terminals are available, both for Windows and Unix. The most popular ones for Windows are: Realterm , PuTTy , ZOC Terminal ,Terminator etc. The most popular ones for Mac OS X are: Coolterm , iTerm ,Terminator , ZOC terminal etc. For Linux, one such application XTerm comes along with the package. As we can see, some of them are cross-platform terminal emulators.