Problems with USB Soundcards

Andy Talbot G4JNT

One of the perennial complaints to be heard on the various Groups where data mode operators congregate is the effect the latest Windows update has had on their setup. Sound cards are no longer recognised, or their name has changed or some other catastrophe has arisen. (At this point Linux users will often chip in with unhelpful comments about how much easier it would be if …) It is probably safe to say that the vast majority of these issues relate to USB peripherals. Internal hardware is not usually so badly affected – or is it?

The main problem is the way USB was designed in the first place. The originators of this very flexible interface aimed to make it plug-and-play and simple for the average user. A mouse, keyboard, external printer or whatever could just be plugged in and it would go. There was to be no defining of ports, the sort of thing COM and LPT users had to worry about in the past. To make this possible every USB interfaced device, when it is plugged in goes into a negotiation informing the host PC of what it is. This is fine, provided the peripheral does actually have a unique identity the host can recognise. Windows remembers this identity and subsequently, whichever USB port it is plugged into in the future, can properly identify the device and provide the right driver. Problems come when generic devices with no unique identity are used. And this can include modern rigs and SDRs.

Sound cards chips are typical of this. What happens is that two functionally similar devices are plugged into their respective USB ports. At start up, Windows polls the USB ports in numerical order, and the first Sound card found gets an identity, let's call it 'Sound Device 1', allocated to it. The second one 'Sound Device 2' etc. This identity is what you then see in your software when you allocate a particular soundcard to your WSJT-X suite, another to your SDR output etc. If you then unplug the one allocated 'SD1', the next time Windows starts, it polls everything again and the one that you thought was 'SD2' is now allocated 'SD1'. Physical soundcards now have a different name and your software gets confused. Even unplugging and plugging into another USB port can have the same effect. There is an added annoyance that the Windows 'default' device, the one that outputs all the bings and bongs, may end up allocated to sound device you don't want – like your transmitter. So all in all, an interface that was meant to make life simple for the majority of users is anything but if you want to change things around – as we often do.

The problem is resolved if the soundcard is given a unique name stored in its hardware. The FDM-Duo, for example, when plugged into any USB port on a PC appears as two new soundcards, "*Line (FDM-DUO Audio v1.04)*" and "*Microphone (FDM-DUO Audio v1.04)*". This is recognised whichever PC USB port it is plugged into. And Windows remembers this name for next time, even when plugged into a different USB port. So user software sees this name and it all usually works quite nicely. The same cannot be said for the several devices I have that all use the PCM2900 USB soundcard chip. This chip has a unique USB identity "*USB AUDIO CODEC*", and every chip has the same name [1]. This is what appears in the pull-down audio selection menu in any software. The real confusion sets in when two of these identically named devices are both plugged in. Each time the USB ports are polled, Windows sees two similar devices, so very kindly adds its own serial number, giving the two devices what it thinks are unique names, like "*1 USB AUDIO CODEC*" and "*2 USB AUDIO CODEC*" Windows then remembers these names – you will find them in device manager and they appear in your pull-down selection menus. Now take one of those codecs, and hot-swap it into another USB port. Windows repolls the USB ports, sees what it thinks is another codec and allocates it "*3 USB AUDIO CODEC*". It doesn't care that the first one was unplugged and might actually be the same hardware. You can keep playing this game with the number increasing each time you generate a new combination of device and port – even one you've used before – so the serial number grows uncontrollably. All are stored and will appear in device manager. Remember, the choice is made depending on which USB port any one is plugged into, and

the polling order, so you can see the problem. If the peripheral has no unique identity to separate it from any other similar one, windows will make one up – and it will probably not be what you intended and will change sporadically [2]. So the golden rule is, if you use USB soundcard devices without unique identities, always use the same USB port for each, and don't unplug them while the system is hot, to avoid a repoll and reallocation.

COM Ports and other devices

USB COM (serial) ports are generally less problematical. Manufacturers of the better ones, like FTDI-Chip, give each chip a unique number, a random text string. Windows sees this string and remembers it, matching against a COM port allocation. So your Tx/Rx switch will always be allocated the same COMx whichever USB port it is plugged into. This can give its own issues, but these are usually more amusing than serious. At one point I was supplying a fully built module that used an FTDI-232 device. The final test before shipping involved plugging it into the shack PC and making sure it worked. OF course, every unit got allocated a new COM port number. When these eventually reached COM200, I thought I'd better do something and went into device manager - advanced, to delete all those obsolete and redundant settings. Even now, at the last count that PC has 13 different COM ports allocated, all corresponding to different bits of kit I have lying around and all bar one using the FTDI-232 device. (The exception is the internal COM1, a real old-fashioned on-board RS232 port). I keep a list pinned on the wall as to which COM port is allocated to which bit of hardware. If only soundcards could be organised the same way.

Some USB peripheral chips include the option of connecting a serial EEPROM to allow users to store their own USB identifiers. The FTDI-232 device allows this, so you are not forced keep to the random identifier. However, the added complexity of doing so is not needed if a simple COMx allocation is sufficient. A customised ID might be useful for manufacturers who want to implement an elegant complete plug-n-play user experience, where customised software searches ports for a specific returned name, and not have to rely on remembering COMx or whatever. Such unique naming is probably more appropriate to situations where other USB interfacing protocols like the HID (Human Interface Device) is adopted and all sorts of odd devices like mice and keyboards need to be separated-out.

References

[1] To be entirely fair to the manufacturers of the PCM2900 chip, they do offer a batch customisation service. If you want to order several thousand devices, they will programme a user-defined string into each device of that batch. But it is the same string for each chip, so is only useful for manufacturers shipping products to multiple customers who only use one at a time.

[2] As soundcard naming is so hit and miss, any software I write that needs to allocate a soundcard forces the user to make a choice right at the start. The pull-down menu to choose a soundcard is presented first, highlighted in red. The rest of the programme will not start until the user has selected one.