Wireless: transparently simple?

Application Note 004

By Myk Dormer
Senior RF design engineer
Radiometrix

So you've decided that somewhere in your 'new product' a **short range wireless interface** would provide a great selling feature (maybe for a remote handset, or a short range interface to a diagnostic tool, or perhaps a sensor head or tool needs to operate without the awkward, trailing cables: who knows ?).

At the feasibility stage you just use a simple asynchronous serial datastream, (something like good old RS232, but probably without the awkward voltage levels, eh?), and over the few meters of test cable it works just fine.

But now you need to prove the radio link. So you look at your data rate , your budget and your available space and try to find a radio.

There are some excellent looking 'turn key' **radio modems** in the market, most offering Hayes/AT modem emulation at 9600 baud or more. But those diecast boxes are as big as house bricks, the range extends over several miles (when you need 25m) , and the price! Hundreds of pounds or more.

Something much simpler is called for. So you get a pair of the numerous short range wideband transceivers (marketed by many firms and all much the same, in their neat little 33x25mm screen cans), add power supplies and aerials copied from the datasheet, connect your data stream and hope.

The specifications always state ranges of 50-250m (most use 1-5mW in the 433 or 868MHz unlicensed bands), for data rates upwards of 10kbps.

So why doesn't it work ?

Ah! Transmitter enable. A bit like RTS , you need to provide this other signal line to tell the transceiver that a data burst is coming. And you need to assert this signal with enough time to let the unit power up and settle.

So re-write your software to drive the line, being really careful with the timing (maybe you have to FIFO buffer the datastream to allow for tx power-up times?), and try again. Does it work properly?

No !

Something is coming out of the receive data pin, and on the scope it looks a bit like data, but if you turn your transmitter off, or even stop sending data with the transmitter on, then all hell breaks loose, and random bytes and framing/overrun errors start spewing out of your uart.

What's going on here ? Believe it or not, it's quite normal. The output of any simple fm/fsk demodulator in the absence of a signal is high level white noise. And telling that from the wanted data is your problem.

So you re-write the software to add some sync and address bytes onto the data bursts, and maybe a checksum or CRC code too, so the (now quite complex) decoder routine has something to look for to discriminate between real data and noise, or interference.

Reliable data communications at last ?

Unlikely.

Data errors are appearing, and they seem to depend on the actual composition of your datastream: send random bytes, or streams of 'U' or '5' and things seem fine. Send 'space' characters, or include large gaps between bytes and nothing works at all.

Back at the datasheet a subtle little detail raises it's head: the transmission path of all these simple radios is ac-coupled (at the receive end), the response rarely falling below a few hundred Hz. And an asynchronous datastream requires a dc-coupled link, or the unbalance between high and low bit durations accumulates into a crippling mark-space distortion.

And so you re-write the software (again!). Long preamble sequences (of 101010 etc) are added to the transmit bursts to settle the receive end, and either careful byte coding and bit number balancing algorithms included, or the whole asynchronous format is thrown out (along with the uarts) and a biphase (Manchester) bit level coding scheme gets adopted.

Once you can get the necessary software routines written, anyhow.

Seems like a lot of work, when all you wanted to do was send a few bytes of data over a few yards of space. Isn't low power radio supposed to be simple to use ?

It is now.

The Radiometrix TDL2A transparent data link module does it ALL for you!

Housed in an industry standard 33x23x7mm case and footprint, the TDL2A combines a sophisticated 433MHz band multi channel transceiver with a dedicated data communication processor to produce a foolproof data link, achieving a range of 100-200m from a transmit power of 10mW. Unlicensed operation is permitted throughout Europe under EN300 220-3

For multiple radio systems (polled networks) a TDL2A can be set to 1 of 8 unique addresses. As well as having unique addresses, the TDL2A allows operation on one of 5 pre-set frequencies in the 433MHz band. These frequencies are non-overlapping and simultaneous operation of TDL2As in the same area on different channels will be possible. Units are supplied on 433.925MHz (Ch0) as default.

It is only necessary to add an aerial and a 5v power supply, and the TDL2A deals with any 9600 baud asynchronous datastream without further user intervention or interface, providing all necessary buffering, packetisation, framing, coding, decoding and reconstitution.

If the user remains aware of the 'half duplex' restriction (no simultaneous transmissions), then for 9600 baud data, the TDL2A behaves **just like a wire**.

There is no 'transmit enable' pin. At idle the TDL2A constantly monitors the channel for valid data. When the first user data byte appears on TXD, the TDL2A enters 'send' mode. The internal transmitter is enabled, and while it settles (a process taking up to 4mS) the first and subsequent bytes of the data stream are stored in an internal FIFO. Once the transmitter is settled, data is read form the FIFO to form a packet, to which is added address and synchronizing bits. This is then biphase coded at the bit level and transmitted, at 16kbits/sec.

When the TDL2A at the other end of the link detects a valid packet, the data bytes are decoded and stored in the receive FIFO, and transferred to the internal uart and appear on the RXD pin in 9600 baud asynchronous format. The 'rx_busy' interface pin is asserted on the first valid packet, and is not deactivated until the receive buffer empties fully. It can be considered as a 'channel busy' or CTS indicator.

The overall latency of the link, from first byte in to first byte out, is less than 15mS. This is achieved by using a very short packet size (three data bytes, plus coding and address bits). If only one or two bytes are sent, then the controller waits for three byte periods (about 3mS) and then sends a shorter packet anyway. The user does not need to include 'end of burst' or 'send now' characters in the datastream.

In simple 'network' applications it is desirable for a unit to selectively talk to several subordinate units. To support such operation the TDL2A has a very simple addressing protocol built in. One of eight addresses (0-7) can be selected in test/setup mode by the ADDR0 through ADDR7 commands. The selected address can be written into e2prom as the power-up default by executing the SETPROGRAM command. A TDL2A only responds to packets containing it's selected address. All units are supplied 'out of the box' set to address 0.

For more complex multi-point systems the user will need to embed the addressing information in the datastream.

So TDL2As are perfect ? They solve every problem ?

Well, no. Like any device, there are limitations which the user must observe when implementing a design based on the TDL2A:

- Short range: half duplex, single channel. Don't expect much over 30m indoors, or 150m outside. And within these operating areas remember that only one TDL can be transmitting at a given time, and only in one direction.

- 5v only: The TDL2A requires a regulated 5v supply, and has 5v logic compatible ports. To connect to a 9 or 25 pin RS232 interface a level shifter/inverter like MAX232 will be needed. If desired, the unit can be supplied fitted to the TDI interface board, which adds a true RS232 buffer, a 9 pin D type socket, a 5v regulator and an RF connector (an SMA type). This board increases the footprint to 61x33x15mm, and has M3 holes for chassis mounting.

- 9600 baud: The only format this unit supports is 9600 baud, 1 start, 8 data, 1 stop, with interface idling in 'mark' condition. (so tie unused TXE pins to 5v).

- No re-transmit: In the presence of interference or at the edge of usable range data packets will be lost. Due to the degree of coding used incorrect data is rarely observed, but bytes will be missing. If the user requires absolute data reliability then they must provide some form of file checking and a 'request for re-transmission' or 'transmit/acknowledge' protocol.

In Radiometrix tests the 'zmodem' file transfer software was found to work extremely well with TDL2A links, allowing very large files to be sent without errors.