Specifications

General

- 433 MHz frequency band
- Max 80 bytes per data packet
- 180 msec needed to transmit 80 bytes
- UART transmission at 57600 baud
- Wireless transmission at 9600 baud

Wireless UART PC Interface

- Powered from USB
- Tx & Rx status LEDs
- FTDI USB interface chip & drivers
- USB Type B connector
- Standard COM port in Window / Linux / Mac

Wireless UART PIC/AVR Interface

- Powered with +5V from PIC/AVR/µP
- Tx & Rx status LEDs
- Logic level UART signals, no MAX232 is needed.

PC Drivers

- Royalty free FTDI drivers for Window / Mac / Linux http://www.ftdichip.com/

Introduction

With the wireless UART / RS-232 link it is possible to connect a PIC / AVR microcontroller wirelessly to the PC.

The data communication is semi-duplex. Max data packet which can be transmitted at once, is 80 bytes. After writing 80 bytes to one of the modules, 180 msec is needed for all data to be transmitted. Smaller packets of course need less time. See calculation further on in this manual.

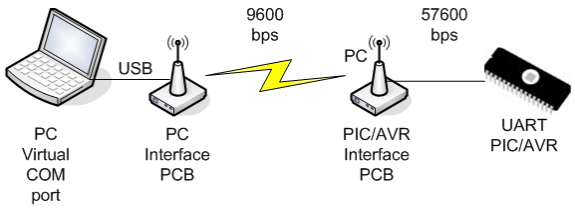

Figure 1, Wireless UART layout

The system consists of 2 modules, both containing a RFM12 wireless chip and a PIC microcontroller to handle the data transmissions.

Wireless UART with PC interface

Figure 2 shows the PCB which has an onboard FTDI chip and can be connected to the PC using the USB connector.

Figure 2, Wireless UART with USB PC interface

Red LED = Transmitting data
Blue LED = Receiving data

Wireless UART with PIC/AVR/uP interface

Figure 3 shows the PCB which is connected to the PIC/AVR microcontroller and directly outputs the UART signals. A level shifter like a MAX323 is NOT needed.

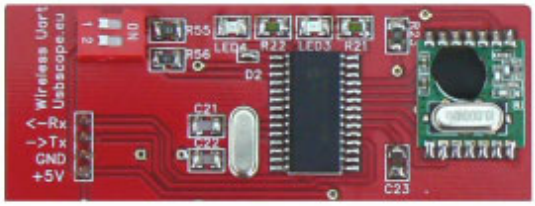

Figure 3, Wireless UART with PIC/AVR interface

The module needs +5V supply which can taken from the PIC/AVR. The UART (Tx, Rx) signals are also 5V logic signals. At 3V3 the module will not work!

Red LED = Transmitting data
Blue LED = Receiving data

Installation Guide

Step1: download the correct drivers for your operating system from the FTDI homepage and install them to you computer. http://www.ftdichip.com/

Step 2: connect the PCB with the USB connector to a free USB port. Your operating system now will detect the new hardware and install the correct FTDI drivers. You will now have a new COM port on your computer.

In windows this looks like shown in figure below (windows will select a number for the port automatically) :

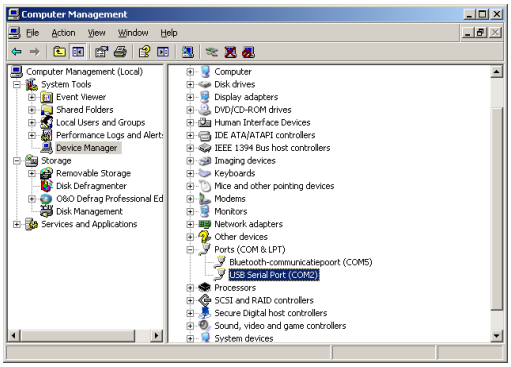

Figure 4, System manger with COM port

Step 3: Use the demo program from our website to send some data to the wireless UART.
http://www.usbscope.eu/wireless_uart/index.html

Programming Guide

Step 1: Use the FDTI COM port in your software just like any other COM port. Configure the port as 57600 baud, 8n1, no handshake, no flow control.

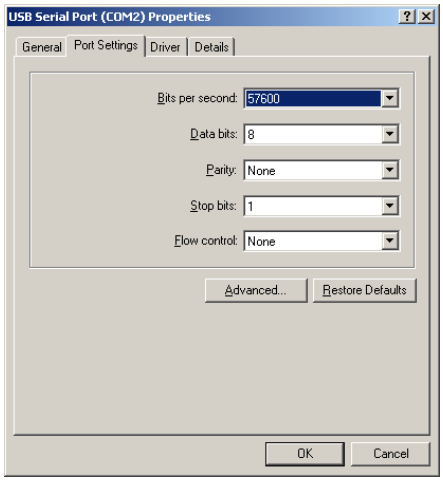

Figure 5, COM port settings

Step 2: In your software, make a command string which has max 80 bytes (shorter string = faster transmission = faster response). Write this string as 1 data packet to the COM port.

Limitation: The data packet send to either one of the modules can NOT start with a byte which has value 254 or 255. These 2 values are reserved for configuration words for the modules and will lead to incorrect transmission of the rest of the data packet. Of course the other bytes in the data packet can have values 254 & 255 only the 1st one not.

Step 3: Wireless transmission of the string to the receiver will take: Time = (20 + 2* nr bytes) msec (80 bytes take 180 msec; 10 bytes take 40 msec).

The receiving PIC/AVR/uP then needs time to process the message and send an answer back. The same formula can be used to calculate the transmission time for the answer.

Step 4: After waiting long enough (1 sec is typically long enough), read the answer back from the COM port and process the received data. Then back to step 2.

Protocol

These wireless modules provide a low level transparent UART interface between PC and PIC/AVR. There is no high level protocol implemented. You probably will need some sort of protocol to "talk" to the microprocessor on the receiving end of this datalink.

I would suggest to use a protocol with readable characters from the ASCII set (0..9;a..Z) to keep things easy to debug. There are a lot of protocol examples on the internet.

Frequencies

The module(s) support 4 different frequencies.

Band	Frequency
0	434.700 MHz
1	434.400 MHz
2	434.100 MHz
3	433.200 MHz

DIP Switches to select frequencies
After power-up, the DIP switch settings are read by the modules and the frequencie band (0..3) is selected.

Software configuration to select frequencies
The frequency also can be changed from the software with a special command data packet. To change the frequency band, send a data packet to the module with size = 2 bytes.

Byte #1 = value 255
Byte #2 = *see table below*

Band	Byte #2 value
0	value 48 (= ascii character "0")
1	value 49 (= ascii character "1")
2	value 50 (= ascii character "2")
3	value 51 (= ascii character "3")

After sending the 2 byte packet, wait for 20 msec before sending new data to the module.

Important: Both modules have to be configured with a 2 byte packet if you want to change the frequency from software. So both the PC software as well as the PIC/AVR/uP should send a 2 byte configuration packet to change the frequency.

Do's and Don't's

Do NOT connect Rx to Tx on the receiver. This will lead to a "loop back" and is a problem on a semi-duplex system as the receiving module will start to transmit too. On a semi-duplex system, only 1 (!) module can be transmitting, the other module(s) must be receiving. Otherwise the data is lost.

This also implies that the receiving PIC/AVR/uP must wait for the complete data packet to be finished, before sending any data back. Therefore it is advised to use a special character sequence to signal the end of the data packet (<CR><LF> is often used).