GPS NMEA data on ER9X

Thanks to Jean-Pierre Parisy who helped me with the unloved C++

With this modification, you can display some life GPS Data on the Tx display.
UTC-Date and Time,
altitude, maximum altitude and lift,
ground speed and course over ground,
latitude and longitude,

You **enter** the NMEA screen 1/4 with **DOWN[long]** from the normal system screens.

You **step through** the screens with **LEFT[short]** or **RIGHT[short].**

You **leave** any NMEA screen with **UP[long]** to statistics screen or with **DOWN[long]** to previous system screen.

Every screen shows the **number of satellites**, that are currently “in view” by the GPS.

As the **first valid information** that the GPS module receives is **time and date**, this screen (1/4) is entered first .
Stepping right, my most important information is displayed.
Screen 2/4 shows **actual Altitude, Lift, Home altitude, Maximum** (positive) **altitude** and the status of the **lift beep**.
Beep is activated by default and its status is shown on the display. Whenever lift is positiv, that is, greater or equal zero, you hear a short beep, once a second, to inform you that the model is not sinking.
You can deactivate or reactivate the beep with the + and – keys (**LEFT[long]** or **RIGHT[long]**).

Altitudes are generally displayed **relative to home altitude**. At any time you can set home altitude from the current altitude. Whenever **home altitude equals Zero**, all displayed altitudes are **relative to height above sea level**.

Menu[short] toggles the home altitude between the saved home altitude value and Zero (and sets home altitude if it was not yet set before).

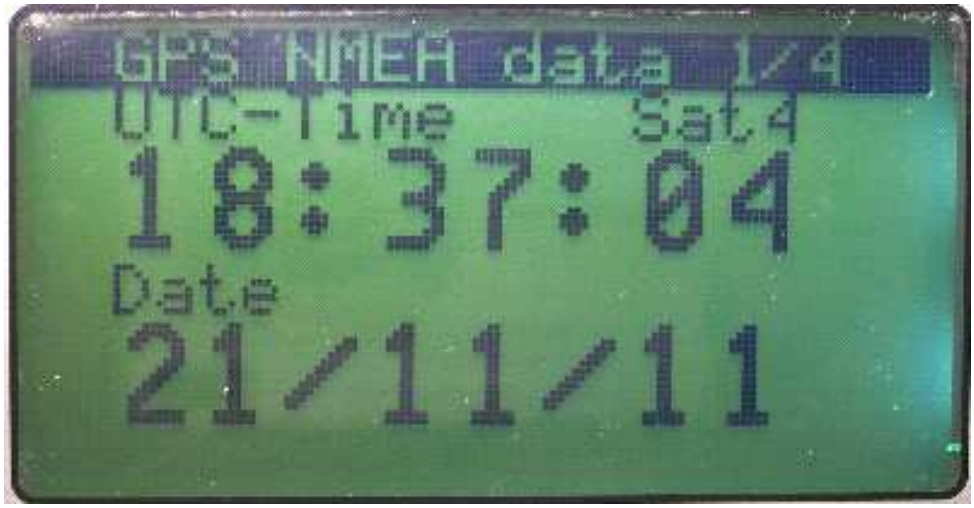
Menu[long] immediately sets home altitude from the current altitude.
Exit [long] resets maximum altitude.

Next screen (3/4) shows **speed and course over ground**, maybe useful for some gliders .
Last screen (4/4) is **latitude and longitude**, not really needed during flight,but supplied from GPS.

The screens are shown in detail on the following pages.

UTC-Time and Date

RIGHT[short] → NMEA 2/4
LEFT[short] ← NMEA 4/4
MENU[short] → switch from UTC-Date to Timer and vice versa
UP[long] → return to statistics screen
DOWN[long] → return to previous system screen



MENU[short] will show

UTC-Time and Timer

RIGHT[short] → NMEA 2/4
LEFT[short] ← NMEA 4/4
UP[long] → return to statistics screen
DOWN[long] → return to previous system screen

MENU[short] → switch from Timer to UTC-Date
EXIT[short] → reset timer to 00:00

Absolute altitude above sea level (Home=0.0) **and up / down lift**

All altitudes are displayed relative to Home altitude.
Whenever Home equals zero, all altitudes are relative to height above sea level.
Lift has a **beeper** for positive lift, that can be enabled / disabled

RIGHT[short] → NMEA 3/4
LEFT[short] ← NMEA 1/4
UP[long] → return to statistics screen
DOWN[long] → return to previous system screen

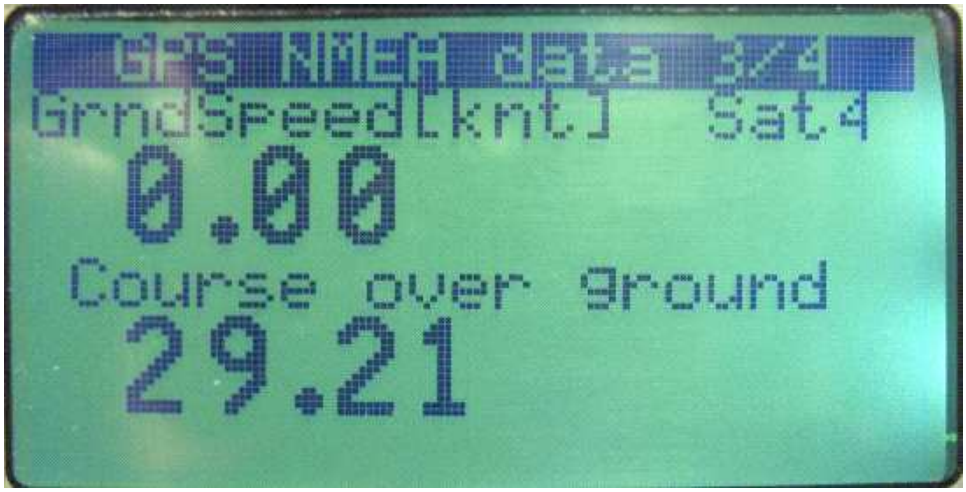
LEFT[long] → Up-Lift beeper off
RIGHT[long] → Up-Lift beeper on

MENU[short] → alternate between absolute (sea level) and relative to home altitude
MENU[long] → Set actual altitude as home altitude
EXIT[long] → reset max altitude to zero

Altitude relative to home altitude (Home<>0.0)

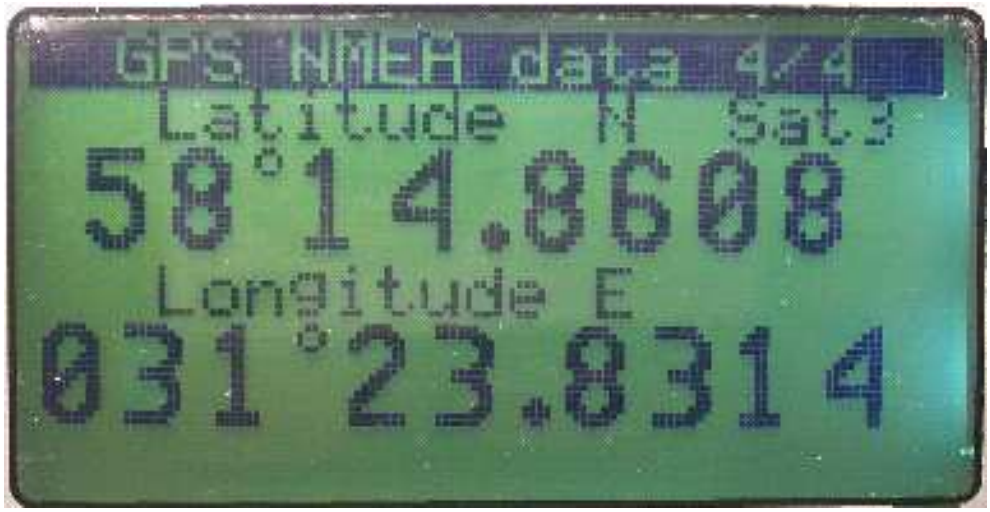
Speed over Ground and Course over Ground

RIGHT[short] → NMEA 4/4
LEFT[short] ← NMEA 2/4
UP[long] → return to statistics screen
DOWN[long] → return to previous system screen



Latitude and Longitude

RIGHT[short] → NMEA 1/4
LEFT[short] ← NMEA 3/4
UP[long] → return to statistics screen
DOWN[long] → return to previous system screen



Modifications of the TX

To use the NMEA build of ER9x to display GPS data, you will have to connect the GPS Mouse with the transmitter module, attach the receiver module to the radio, configure GPS Mouse and transceiver modules and flash the radio with the NMEA version of ER9X. (It's supposed that you know how to flash your TX)

The modifications in the radio require soldering of small tracks on the main circuit board. Be warned that it is possible to damage the board permanently if care is not taken.

Hardware and software requirements

You will need a GPS module and two transceiver modules (or one transmitter and one receiver module).

You must have the programm to configure your GPS mouse (e.g. SIRFDemo336XP.exe, depends on your GPS mouse) and you must have the program to configure the APC modules (RF-Magic-42.exe from sureelectronics.net).

To program GPS mouse and APC200 modules, they must be attached to a real COM port of a PC. You probably have to solder some temporary cables, that allow you to connect them to the COM port of your computer

I have used the following circuits:

APC200

Holux GM-210

A GPS mouse **Holux GM210** (http://www.ebay.de/itm/180653870348) combined with an **APC200** (http://www.sureelectronics.net/goods.php?id=1053) transceiver module (431MHz - 478 MHz) as transmitter (to be attached to the plane) and another APC200 transceiver as ground receiver that feeds the GPS data signal into the ATMega64 of the TX.

GM210 as well as APC200 work with 5 volt supply voltage and have UART input and output with TTL and RS232 level. This way, they can be connected easily with each other and/or to any COM port. But despite of this, to configure the APC200 module, you need a RS232-to-TTL level converter, as this module must be configured via its TTL interface.
GPS receivers normally (or can be set up to) output their data as records in NMEA format. This is pure ASCII data and is commonly used.

There are many different NMEA record types available that contain different data. But not all record types are supported by every module.

Normally you can configure the module, which record types and at what repetition rate these records are to be output from the GPS receiver.

I have configured the GM210 to output **$GPGGA and $GPRMC records once a second at 4800 Bd.**

The APC200 modules that transmit at a selectable RF of 431 - 478 MHz and a maximum output of 10dbm are mid range modules (to my opinion). With an RF rate of 2400 Baud I verified a ground range of over 800 m with nearly (but not really) free line of sight.

The maximum range of the APC200 modules increases with lower over the air transmission speed. Therefore I configured them for the lowest acceptable speed.

Over the air transmission speed and UART serial input / output speed can be configured independently.

I have set up the **RF transmission speed to 2400 Bd and the serial UART speed to 4800 Bd.**
That means, that the GPS mouse has to send with 4800 Bd, the APC200 transmitter sends over the air at 2400 Bd, the APC200 receiver outputs with UART speed of 4800 Bd and the TX reads the data with 4800 Bd.

This is possible without data loss, because the APC200 has a buffer of 512 bytes and the selected NMEA records have a total of about 150 Bytes only, once per second.

The ER9x NMEA modification reads the data via UART, extracts the required fields and displays them on four different screens as shown above. For performance reason, the GPS data is not checksum checked but used as received.

Steps to do:

- **Verify, that you have everything available**
- **Configuring the GPS mouse.**
- **Configuring the APC200**
- **Removing the housing of the GPS Mouse**
- **Connecting the GPS mouse and the APC200 transceiver**
- **Verify, that all the switches do work.**
- **Modify the transmitter hardware (rewire switches)**
- **Flash the TX**
- **Connect the APC200 module with the ER9X TX**
- **Final Check**

Starting the modifications

Step 1:

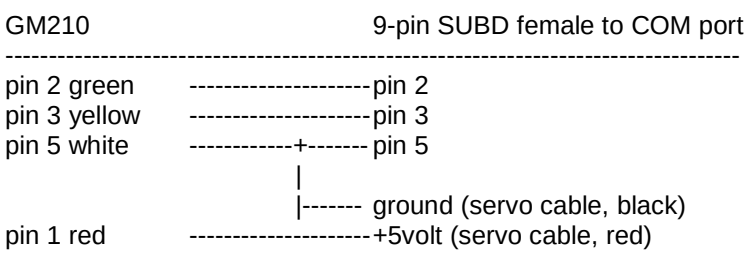
Verify, that you have everything available

- A PC that has a real COM port
- the program to configure the GPS mouse (e.g. Sirfdemo336xp.exe)
- the program to configure the APC200 transceivers (e.g. Rf-Magic42.exe)
- the GPS mouse
- two APC200 transceivers
- an RS232-to-TTL converter
- The resistors for the telemetrie mod (2 x 220 Ohm)
- one or two SUBD female plugs
- one or two servo cables

Step 2:

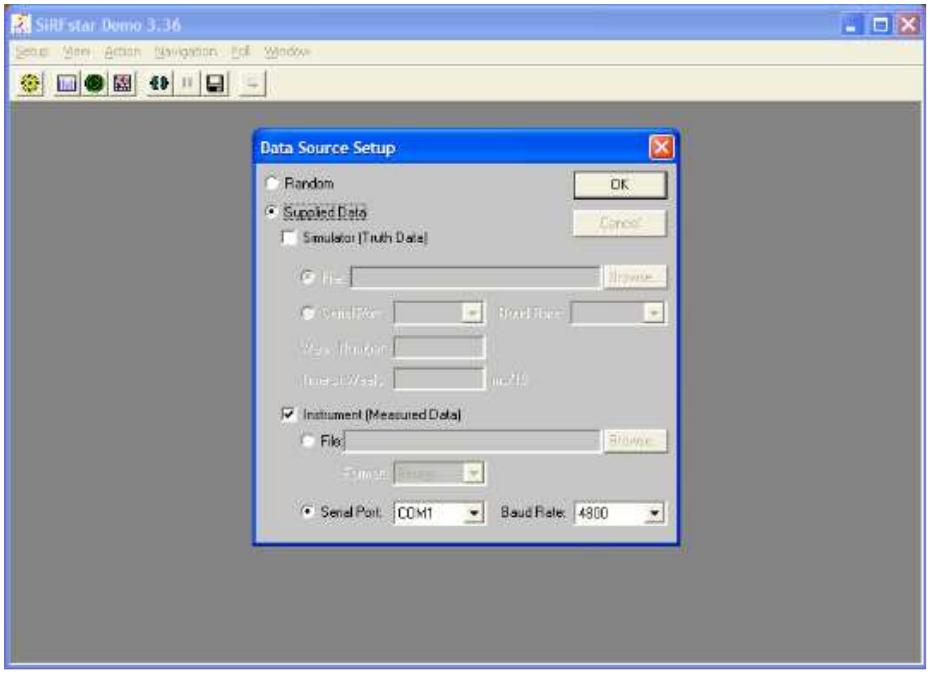
Configuring the GPS mouse.

You need a program like SirfDemo336XP.exe (depends on your GPS mouse) and a cable, that connects the RS232 signals from the mouse to the COM port of your computer.

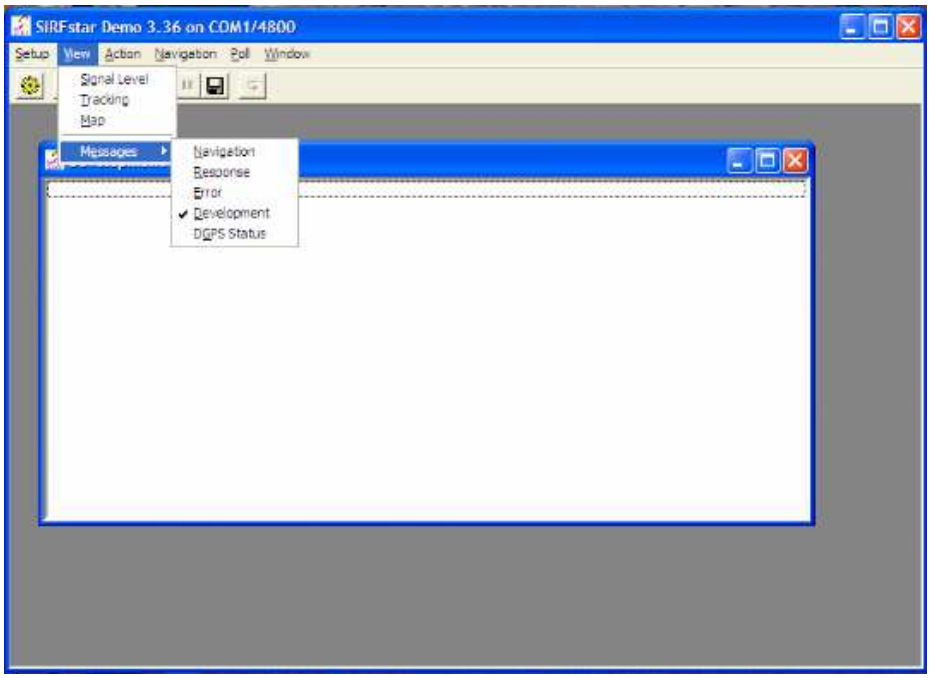


Setting up the GPS mouse to output $GPGGA and $GPRMC records once a second at 4800 Bd.

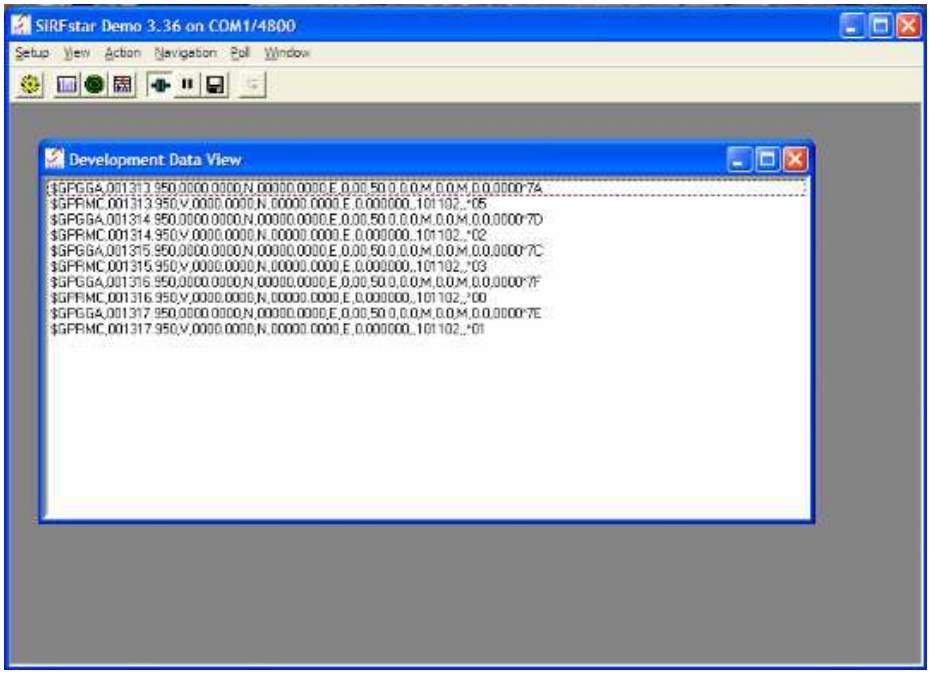
Connect your GPS Mouse to the COM Port e.g. **COM1:** and power it on.
Start the program **'sirfdemo336xp.exe'**
Set parameters as shown below



In the menu, **click** on '**View**' - '**Messages**' - '**Development**' to open the **'Development Data View**' window

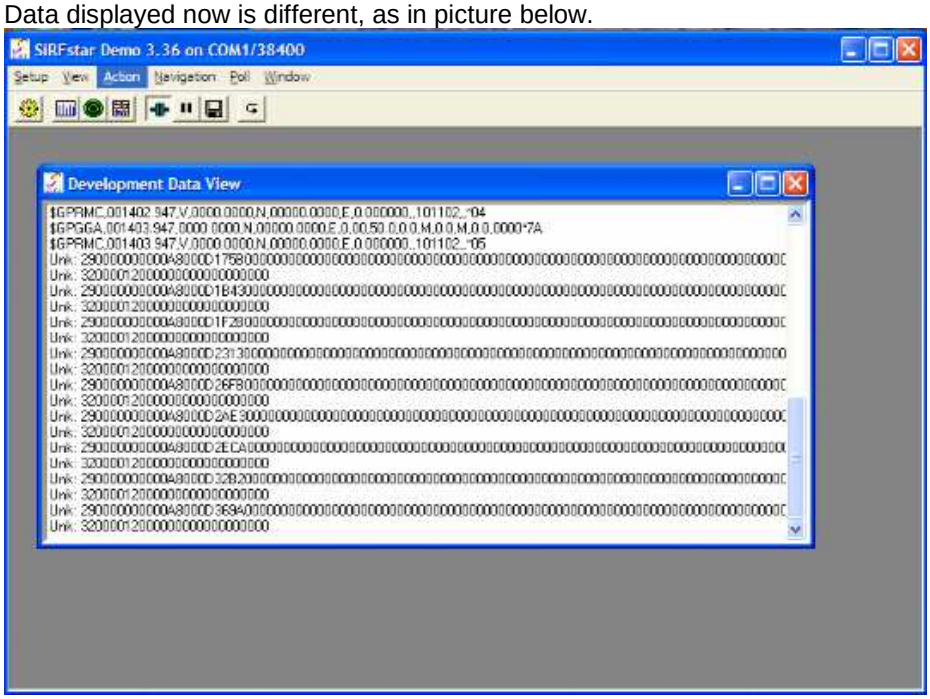


If your GPS Mouse is sending data (a GPS mouse is permanently sending data by default), they will be displayed in this window. If data is readable and looks similar to the screen below, your mouse is already sending with NMEA data protocol.

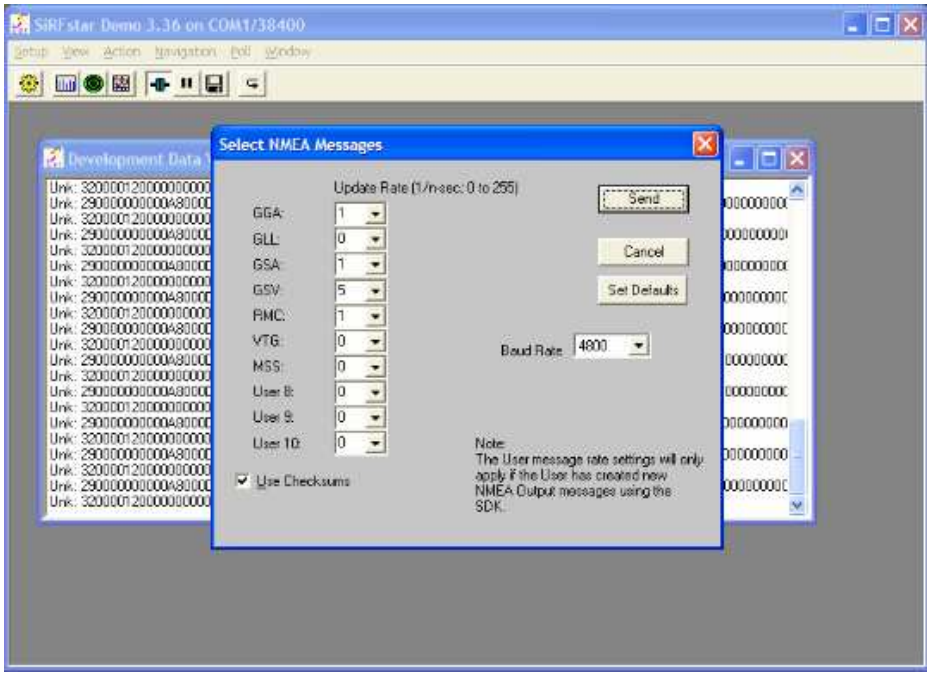


You probably will see more and different record types displayed from your GPS Mouse.
The selection, which recordsets and at which repetition rates they are sent, can only be modified when you switch from SiRF mode to NMEA mode. Therefore we firstly have to switch to SiRF mode and then back to NMEA mode

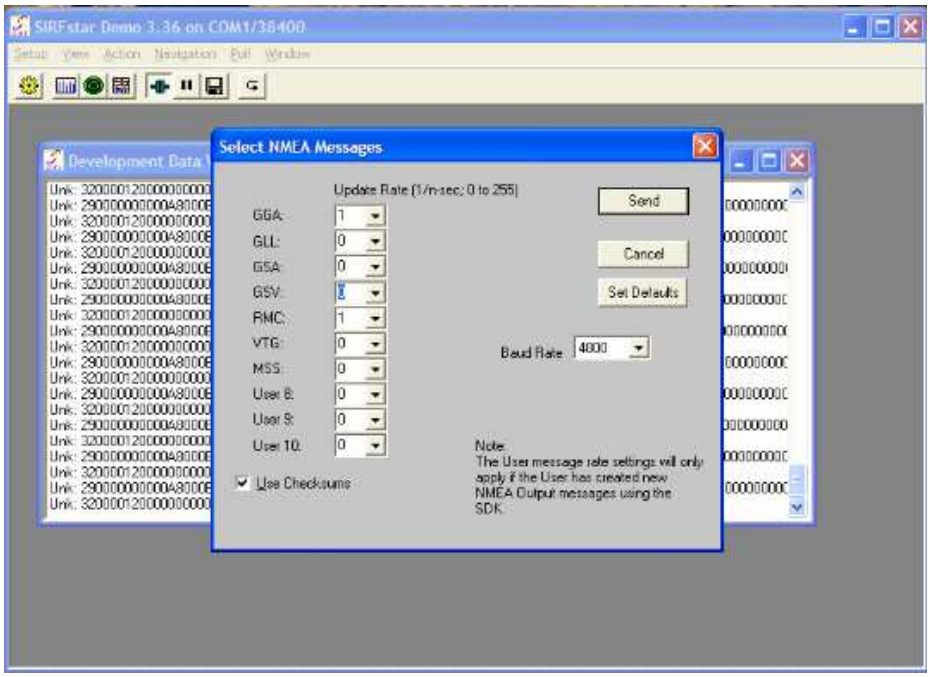
To get this, do the following steps:
In the menu, **click** on '**Action**' - **'Switch to SiRF Protocol'**
Data displayed now is different, as in picture below.



Click on **'Action' - 'Switch to NMEA Protocol'**
The fields will be set as below

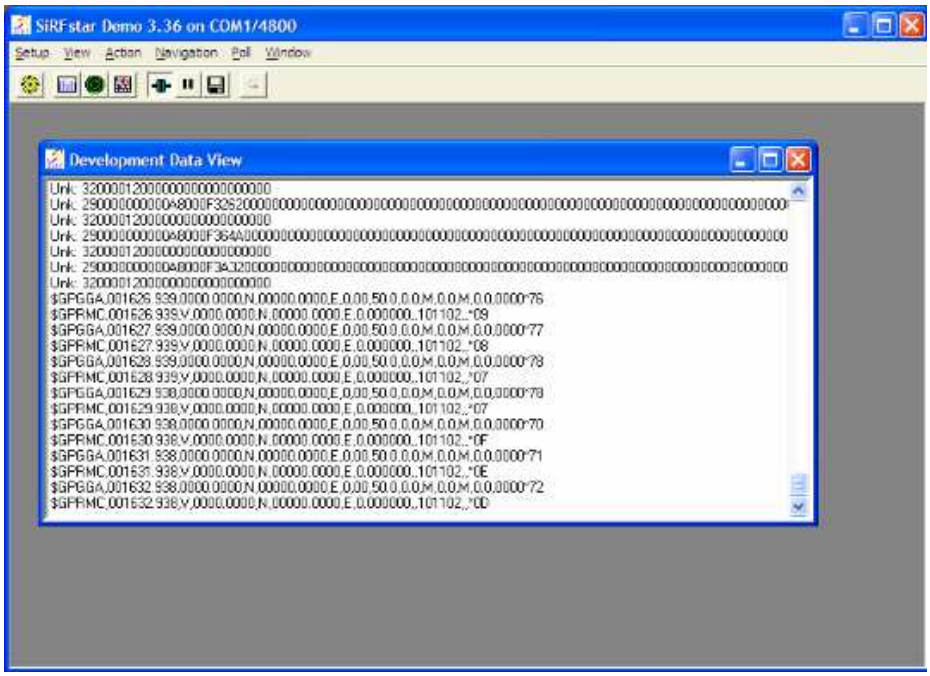


Now **set** the fields according to the picture below
This will select GGA and RMC records at a repetition rate of one record for each recordtype per second
Click 'Send'



You now should see nothing else but records that start with $GPGGA or $GPRMC. And they should appear once per second.

You can verify this from the first field following the record id, that is the UTC time (hhmmss with 3 digits following the decimal point e.g. 0016226.939 = 00:16:26.939)



The GPS Mouse now is successfully set up to work with ER9X-NMEA.

Terminate the program and **power off** and **disconnect** your GPS Mouse

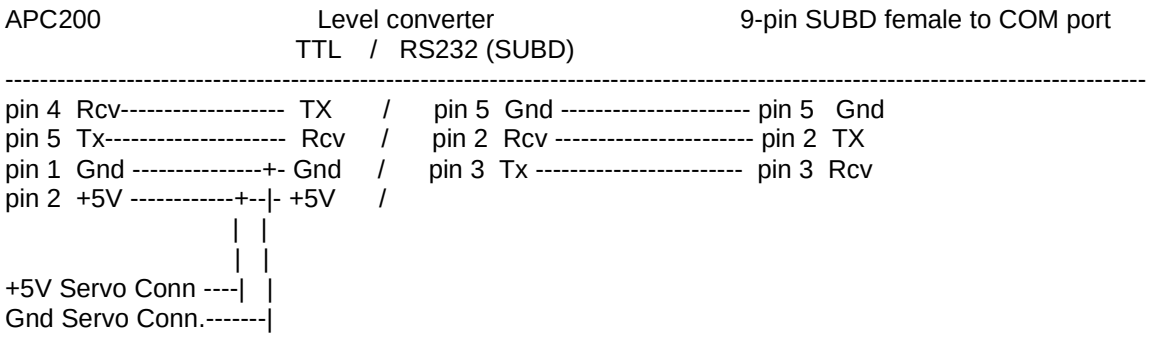
Step 3:
Configuring the APC200

Programming of the APC200 module can be done only via the serial TTL interface. To program it, the module has to be switched into configuration mode. This happens, when the module receives special data while it is powering on.

When the configuration program RF-Magic is started, it permanently sends certain data over the serial port to the attached APC200 module. When the module is powered on while this data is supplied to its serial TTL input line, it switches into configuration mode and stays there, as long as there is configuration data traffic. It switches back to normal mode within parts of a second, when this data traffic is stopped.

To set the APC200 module into configuration mode, you have to start the configuration program RF-Magic.exe and while the program is running, power on the connected APC200 module. If RF-Magic does not immediately recognize the module, retry several times (power off / power on), until the status line shows successful connection.

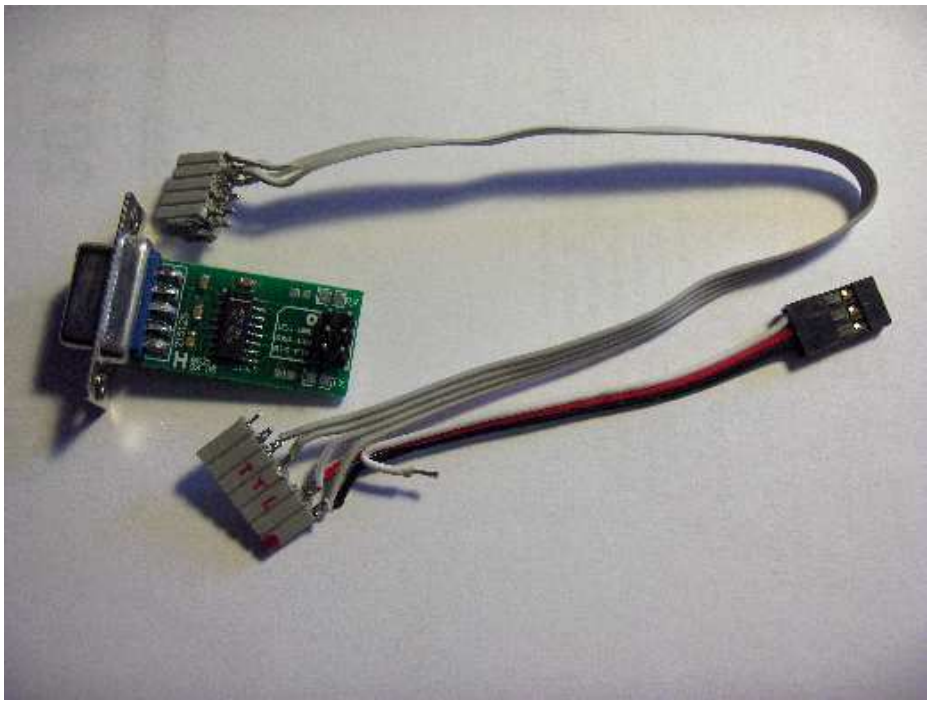
Preparing the connection cable



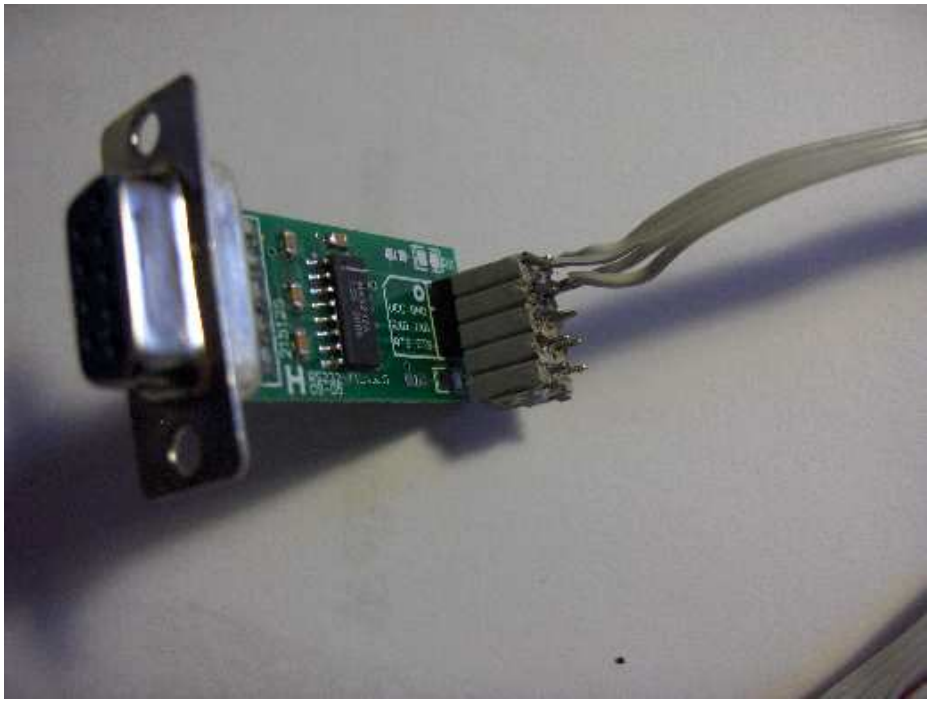
APC200 and the level converter are supplied with 5 volt via a servo connector, attached between APC200 and converter.

The connectors for the level converter and the APC module in detail.

Level converter, plugs for level converter and APC200 module and servo connector for powering both modules

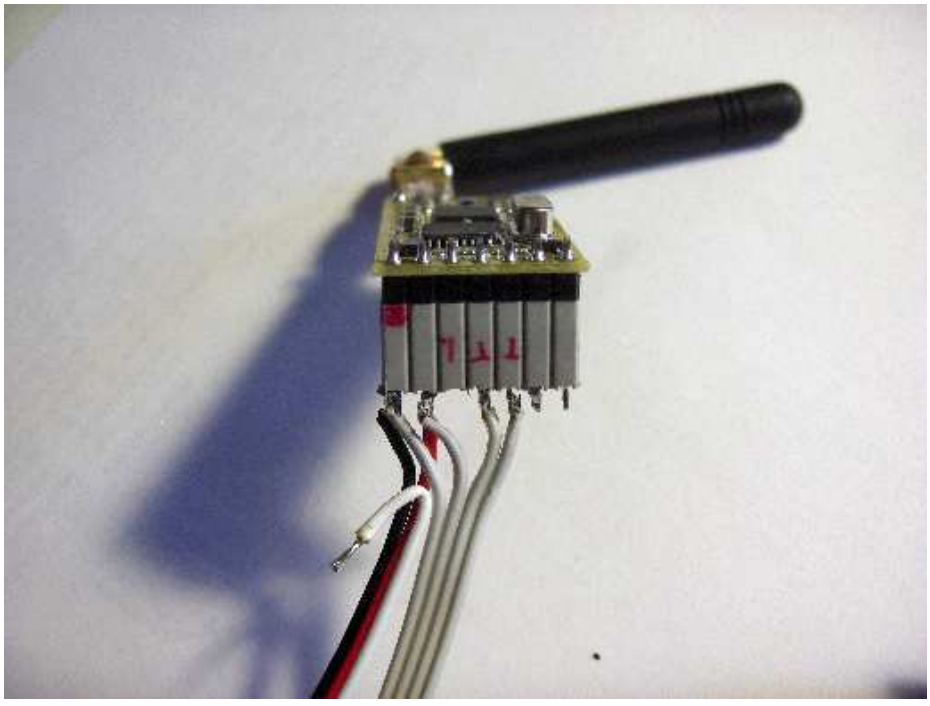


Pins on converter are top to bottom, front to rear:
+5 volt, ground, RxD (TTL), TxD (TTL)

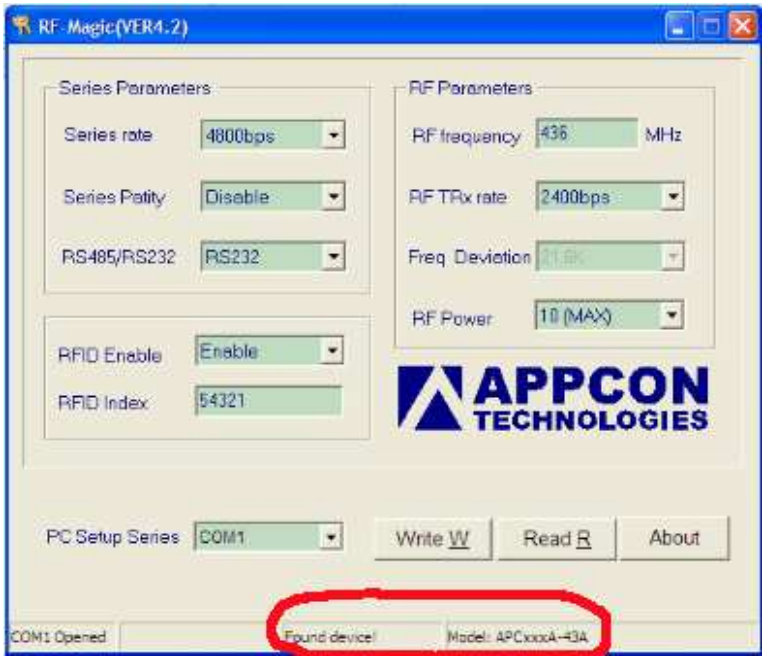


APC connector marked with 'TTL' has ground on outer pin marked with the red dot.
The signals of the 4 wire cable for the APC200 from top to bottom are:
Tx (TTL), Rx (TTL), +5volt, ground (0volt)

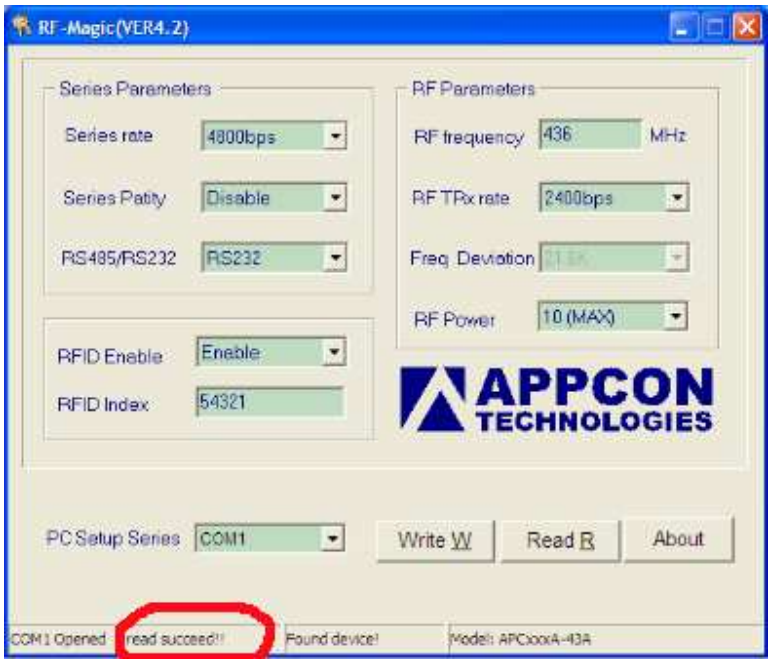
Pins on APC200 module are left to right:
Ground, +5 volt, RxD (TTL), TxD (TTL)



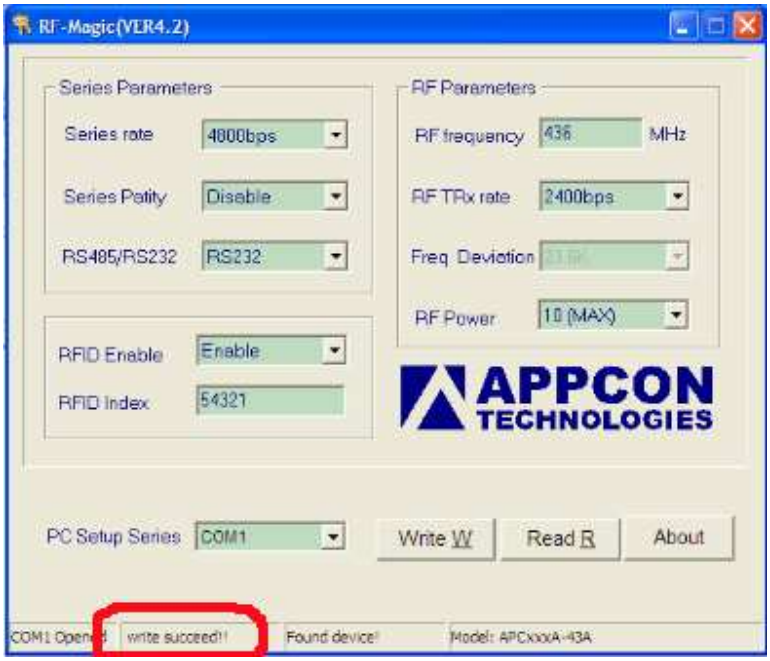
Connect the APC200 module via the level converter to a real COM port of your PC
Don't power up the module
Start the configuration program RF-Magic
Power on the APC200 module
The program should immediately detect the module and confirm it with a status message



Click on the **'Read R'** button
This displays the current setting of the module



Set the parameters as shown below, but set **'RF frequency'** and **'RFID Index'** to your requirements
Click on the **'Write W'** button



This writes the configuration data permanently to the APC200 module
(You can modify the data at any time later)
Take a note of your settings, as you have to configure the second APC200 identically.

Power off and remove the module

Connect the second APC200 module and **configure** it identically.

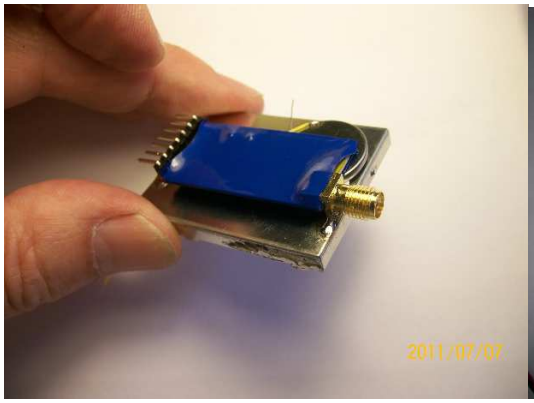
When powered on, both APC200 modules will link together and work like a wireless RS232 cable.
Data put in on one end will drop out at the other end.

Step 4:
Removing the housing of the GPS Mouse

To save some weight on the GPS mouse, remove the housing. You have to, more or less, destroy the housing as upper and lower half are glued together and must be pried open with some force.

Step 5:
Connecting the GPS mouse and the APC200 transceiver

Cut the original mouse cable to still have enough length to connect the mouse with the APC200
As power supply, use a servo cable and solder it to pins 1 and 2 of the APC200 (watch polarity)



GM210		APC200	
pin 1 red	----------	pin 2	+ 5Volt
pin 4 blue	----------	pin 5	TTL Tx
pin 5 white	----------	pin 1	ground
pin 6 black	----------	pin 4	TTL Rx

To power your GPS with transmitter, plug the servo cable into an unused servo position of the receiver or use a Y-cable splitter.

Check, that the LED of the GPS mouse lights up.

Step 6:
Verify, that all the switches do work.

Throttle cut and **Aileron-D/R switch** will be rerouted in the next step. Go to the system screen that displays the switches or to the diagnostic screen 4/6 and verify, that all switches are working.

Step 7:
Modify the transmitter according to the following description (rewire switches)

This modification is based on the very good instructions by Gruvin9x.

http://code.google.com/p/gruvin9x/wiki/FrskyInterfacing

Accessing TXD and RXD on the ATmega\l

The ATmega RXD and TXD special-purpose pins are used for simple switch inputs in the factory '9X -- namely, Throttle-Cut and Aileron-D/R, respectively.
To free these pins up for use with serial data, we need to move those two switches elsewhere. The two pins chosen for their new home were pins 41 and 42, or PC6 and PC7 respectively.
The modification requires the use of *really fine* wire. I used 0.2mm, enamelled copper wire. You can also use fine, insulated wire, salvaged from the newer (yet now old) high-density computer IDE cables. Each of these wires is a *single-core*, PVC insulated wire. You most definitely do *not* want to try and use any kind of multi-strand wire -- no matter how fine it it is.

IMPORTANT: The main reason to use *really* fine wire may not be what you think. It is certainly possible (though more difficult) to use heavier, stronger wire. But doing so will increase the risk of accidentally damaging the delicate ATmega chip's pins. (Think of it as using a wrecking bar to pick your teeth -- or something like that! :P) In practice, the slightest accidental tug by 'big fat, shaky, human hands on a big fat wire', can quite literally rip an ATmega pin right of the chip. Trust me -- *use really fine wire!*

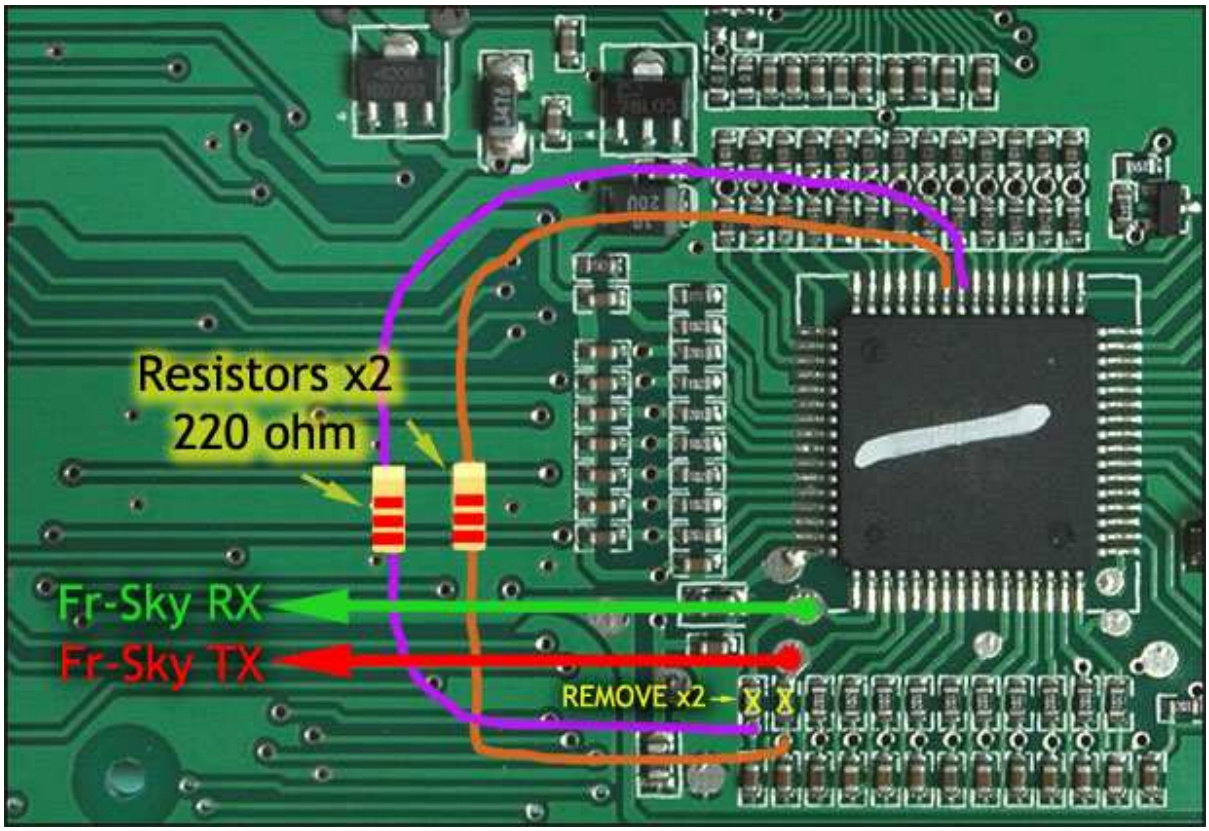
Re-wiring Procedure

Referring to the image below ...

- Remove the two left-hand-most resistors as indicated in yellow
- Hot glue a couple of 220ohm resistors (1/8th or 1/4watt are OK) to a nearby, clearish area of the PCB
- Solder your 'super fine' wires to pins 41 and 42 of the ATmega and connect to the top end of the resistors as shown. Use alcohol if required -- to steady your hands! (Just joking -- but if it works for you -- go right ahead. :p)

HINT: Pre-solder the wire end and the ATmega pins. Use a solder sucker or de-soldering wick to clear away excess solder from the two pins and to remove any solder bridge. Then lay the first wire end over the pin and apply the soldering iron directly down on top of it for half a second, then directly straight up -- then let go the wire. Give the wire a gentle tug in the direction directly away from the chip (in line with the pin).If it holds -- you're done. Don't futz with it. If you're right handed, solder the left hand wire first, so the the soldering iron does not lay over the first wire when trying to solder the second.

- Solder wires from the bottom ends of the hot-glued resistors to the little vias (holes) near where the two removed resistors were, as shown in the image.



NOTE: It pays to use small dabs of hot-glue near the ATmega pins and in a couple of places along the wires to hold them in place. The last thing you want is for something to get hooked under them and rip them off, possibly damaging the ATmega pins in the process!

Step 8:
Flash the TX

Go to the system screen that displays the switches or to the diagnostic screen 4/6 and verify, that **Throttle cut** and **Aileron-D/R** switch do no longer work.

Flash the TX with the NMEA version of ER9x

Verify, that all switches work correctly again.

That verifies, that the telemetrie mod has been done correctly

Press DOWN[long] to enter the telemetrie display
Press LEFT[short] or RIGHT[short] to step through the screens
Most of the values should show a big questionmark
Press UP[long] or DOWN[long] to leave the telemetrie screens

Step 9:
Connect the APC200 module with the TGY9X TX

APC200 TGY9X

TTL Tx Pin 5 ----------via resistor of 1K to Pin 2 of ATMega (named 'Fr-Sky RX' on the picture above)
+5Volt Pin 2 ----------+5 Volt
Ground Pin 1 ----------Ground

Step 10:
Final Check

Power on the GPS-transmitter and the TX
Enter the NMEA screens
UTC-Time and Date will be the first fields to display real values.
Most of the other fields will be similar to 0000.000
When the GPS mouse gets a fix (Sat >=3), all the fields will show the received values.
'GroundCourse' and 'Speed over Ground' require, that the GPS Mouse is being moved.

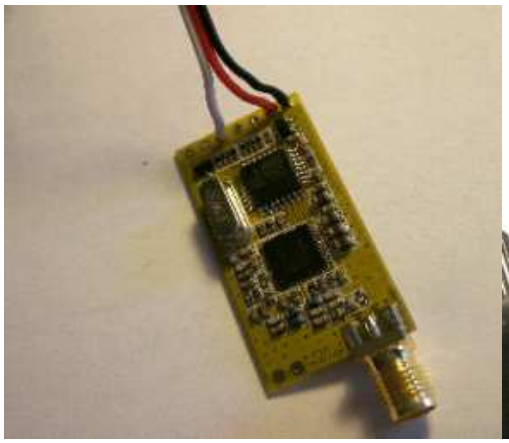
Some Pictures:

NMEA transmitter

GPS module from backside with the RF module and an externally connected backup battery. Modules not yet wired.

NMEA receiver

APC200 Module with pins unsoldered and cable directly attached



Hole for the antenna connector

Antenna connector on the backside of Tx case

Module, switch and servocable, secured with hot glue

Mounted antenna, module and power switch

==========
==========

----------some technical background----------

NMEA Records:

==========

The length of the individal fields may vary (even be empty), but all Fields are always there and are separated by colon.
$GPGGA data is valid when field 6 <> "0" (Null)
$GPRMC data is valid when field 2 <> "V"
UTC-Date and UTC-Time are available some time before the GPS gets a fix

Sample with position not fixed:
==========

$GPGGA,130607.704,5014.8588,N,01123.8331,E,0,01,0.0,0.0,M,47.7,M,0.0,0000*73
$GPRMC,130607.704,V,5014.8588,N,01123.8331,E,0.00,,110211,,*0D

Sample with position fixed:
==========

$GPGGA,130415.711,5014.8480,N,01123.9166,E,1,08,1.0,387.2,M,47.7,M,0.0,0000*79
$GPRMC,130415.711,A,5014.8480,N,01123.9166,E,2.59,312.81,110211,,*0E

Layout of the $GPGGA record

$GPGGA - Global Positioning System Fix Data, Time, Position and fix related data fora GPS receiver.

```
          1           2    3  4         5 6 7   8   9   10 11 12 13   14    15
          |           |    |  |         | | |   |   |   |  |  |  |    |     |
$GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh<CR><LF>
```

Field Number:
1) Universal Time Coordinated (UTC)
2) Latitude
3) N or S (North or South)
4) Longitude
5) E or W (East or West)
6) GPS Quality Indicator,
   0 - fix not available,
   1 - GPS fix,
   2 - Differential GPS fix
7) Number of satellites in view, 00 - 12
8) Horizontal Dilution of precision
9) Antenna Altitude above/below mean-sea-level (geoid)
10) Units of antenna altitude, meters

11) Geoidal separation, the difference between the WGS-84 earth
ellipsoid and mean-sea-level (geoid), "-" means mean-sea-level
below ellipsoid
12) Units of geoidal separation, meters
13) Age of differential GPS data, time in seconds since last SC104
type 1 or 9 update, null field when DGPS is not used
14) Differential reference station ID, 0000-1023
15) Checksum

Layout of the $GPRMC record

$GPRMC - Recommended Minimum Navigation Information

```
          1            2  3    4  5         6  7    8    9      10 11 12
          |            |  |    |  |         |  |    |    |       |  |  |
$GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,xxxx,x.x,a*hh<CR><LF>
```

Field Number:
1) UTC Time
2) Status, V = Navigation receiver warning
3) Latitude
4) N or S
5) Longitude
6) E or W
7) Speed over ground, knots
8) Track made good, degrees true. = = Course over ground (COG)
9) Date, ddmmyy
10) Magnetic Variation, degrees
11) E or W
12) Checksum

Checksum calculation
==========

my_d is that part of the NMEA record that is between "$" and "*"

'All the bytes are XORed

(This is how I calculate the checksum in HB++)

```
for n=1 to len(my_d)
        myhi=mid(my_d,n,1)
        cs=cs xor asc(myhi)
next

CheckSum= right("0" & hex(cs),2)    'to get a 2 character string
```