GPS NMEA data on ER9X

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

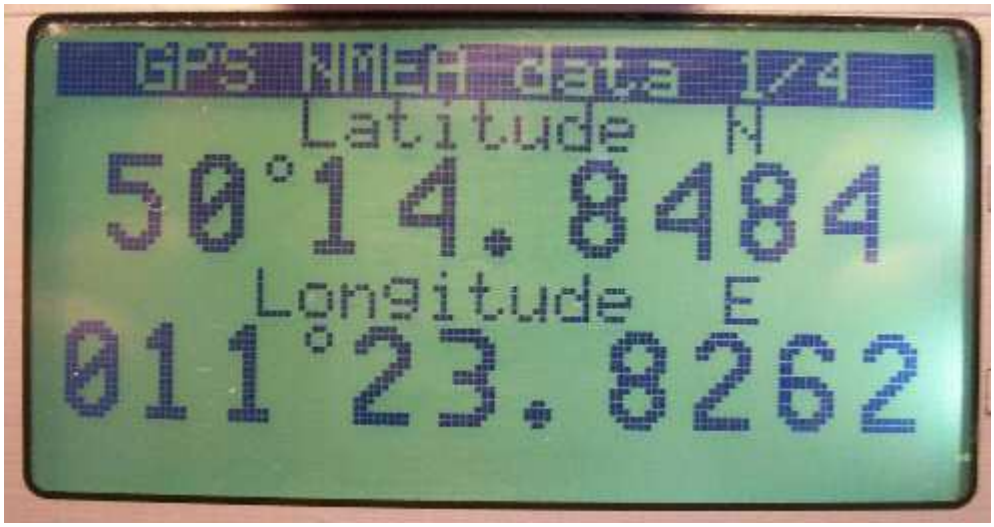
With this modification, you can display some life GPS Data on the Tx display.

You enter the NMEA screen 1/4 with DOWN[long] from the normal system screens.
You step through the screens with DOWN[short] or UP[short]

According to my setup, the TX displays the following data:

Latitude and Longitude

UP[short] --> NMEA 4/4
DOWN[short] --> NMEA 2/4
HOME[short] --> STAT screen
EXIT[short] --> system screens



Speed over Ground and Course over Ground

UP[short] --> NMEA 1/4
DOWN[short] --> NMEA 2/4
EXIT[short] --> system screens

Absolute altitude over sea level and up / down lift

Lift has a **beeper**, that can be enabled / disabled for positive lift

UP[short] --> NMEA 2/4
DOWN[short] --> NMEA 4/4
LEFT[short] --> Up-Lift beeper off
RIGHT[short] --> Up-Lift beeper on

MENU[short] --> alternate between absolute and relative altitude
EXIT[short] --> reset home altitude to zero
EXIT[short],MENU[long] --> Set actual altitude as home altitude
EXIT[long] --> system screens

Altitude relative to a settable home altitude

UTC-Time and Date

UP[short] --> NMEA 3/4
DOWN[short] --> NMEA 1/4
MENU[short] --> switch from UTC-Date to timer and vice versa
EXIT[short] --> system screens

UTC-Time and timer

UP[short] --> NMEA 3/4
DOWN[short] --> NMEA 1/4
MENU[short] --> switch from timer to UTC-Date
EXIT[short] --> reset timer to 00:00
EXIT[long] → system screens

To use the NMEA build of er9x you will need to make some modifications to the hardware in your radio. These modifications require soldering of small tracks on the radio circuit board. Be warned that it is possible to damage the board permanently if care is not taken.

You must have the programm to configure your GPS mouse (e.g. SIRFDemo.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 attach the GPS mouse and the APC200 modules to the PC, you probably have to solder some temporary cables, that allow you to attach them to the COM port of your computer.

It requires, that you apply the telemetrie mod to the TX to free up the input pin on the ATMega64 of the TX
http://code.google.com/p/gruvin9x/wiki/FrskyInterfacing

Note: You only need to follow the rewire instructions. No need to build the interface with the level converter.

I have used the following circuits:

APC200

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 a PC for reprogramming without the need of any level change adapter.

GPS receivers normally output their data as records in NMEA format. This is pure ASCII data and is commonly used.

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

Normally you can configure the module, which records 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 10bdm are mid range modules (to my opinion). 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.

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 shown above.

Step 1
Verify, that you have everything available

A PC that has a COM port
the program to configure the GPS mouse
the program to configure the APC200 transceivers

the GPS mouse
two APC200 transceivers
The resistors for the telemetrie mod
one or two SUBD female plugs
one or two servo cables

Step 2:
program the GPS mouse.

You need a program like SirfDemo.exe and a cable, that connects the RS232 signals from the mouse to the com port of your computer.

```
GM210                           SUBD female to COM port
-------------------------------------------------------------------------
pin 2 green     ---------------------pin 3
pin 3 yellow    ---------------------pin 2
pin 1 red       ----------------------          +5volt (servo cable)
pin 5 white     ----------+---------            pin 5
                          |--------  ground (servo cable)
```

Step 3:
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 4:
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 5:

Verify, that all the switches do work.

Apply the telemetrie mod to the TX to free up the input pin on the ATMega64 of the TX according to this description
http://code.google.com/p/gruvin9x/wiki/FrskyInterfacing

Note: You only need to follow the rewire instructions. No need to build the interface with the level converter.

Verify, that two of the switches do no longer work.

Flash the TX with the NMEA version of ER9x

Verify, that all switches work correctly.

That verifies, that the telemetrie mod has been done correctly

Press down [long] to enter the telemetrie display
Press up or down [short] to step through the screens
All the values should show a big questionmark
Press exit to leave the telemetrie screens

Step 6:
Connect the APC200 module to the TX

APC200		TGY 9X
TTL Tx Pin 5	----------	via resistor of 1K to Pin 2 of ATMega
+5Volt Pin 2	----------	+5 Volt
Ground Pin 1	----------	Ground

Step 7:
power on the GPS-transmitter and the TX
Most of the fields will be similar to 0000.000
UTC-Time and Date will be the first values to display real values.
When the GPS mouse gets a fix, all the fields (except course over ground) will show the correct values.
Course over ground requires, that the receiver moves.

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

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

NMEA Records:

==========

The length of the individual 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
```