OK I measured this, and the Multiplex MSB protocol is straight forward

semi-duplex 38.4kbps (one-wire) at 3.3V
Byte structure 10-bit 0xx1, where x = nibble (4 bits), LSB first

RX polls sequentially for addresses 2-F (sends 02h - 0Fh) at 5ms intervals (Cockpit SX supports only up to 7)
If a sensor is present on the bus, it commences its response 300us after the polling byte. The user has to ensure that there are no address conflicts.

A sensor response consists of 3 bytes (6 nibbles)
Nibble 0: Unit type, i.e. [mAh] (table below)
Nibble 1: Address assigned to parameter (same as RX polled)
Nibble 2-5: LSbit is the alarm bit (the one first transmitted, as it's LSB first)
Nibble 2-5: Rest of bits is 15 bit signed
(nibbles 2-5 I have not verified negative numbers, nor alarm bit, but with positive values it works. Depending on unit selected, the decimal point is placed in a fixed position, i.e. Ampere is always indicated with one decimal)

I refer to Stoeckli on nibbles 2-5 as well as the unit types. I have verified mAh and A readings on my Cockpit SX. Once I get my own MCU programming done, I'll run through the rest.

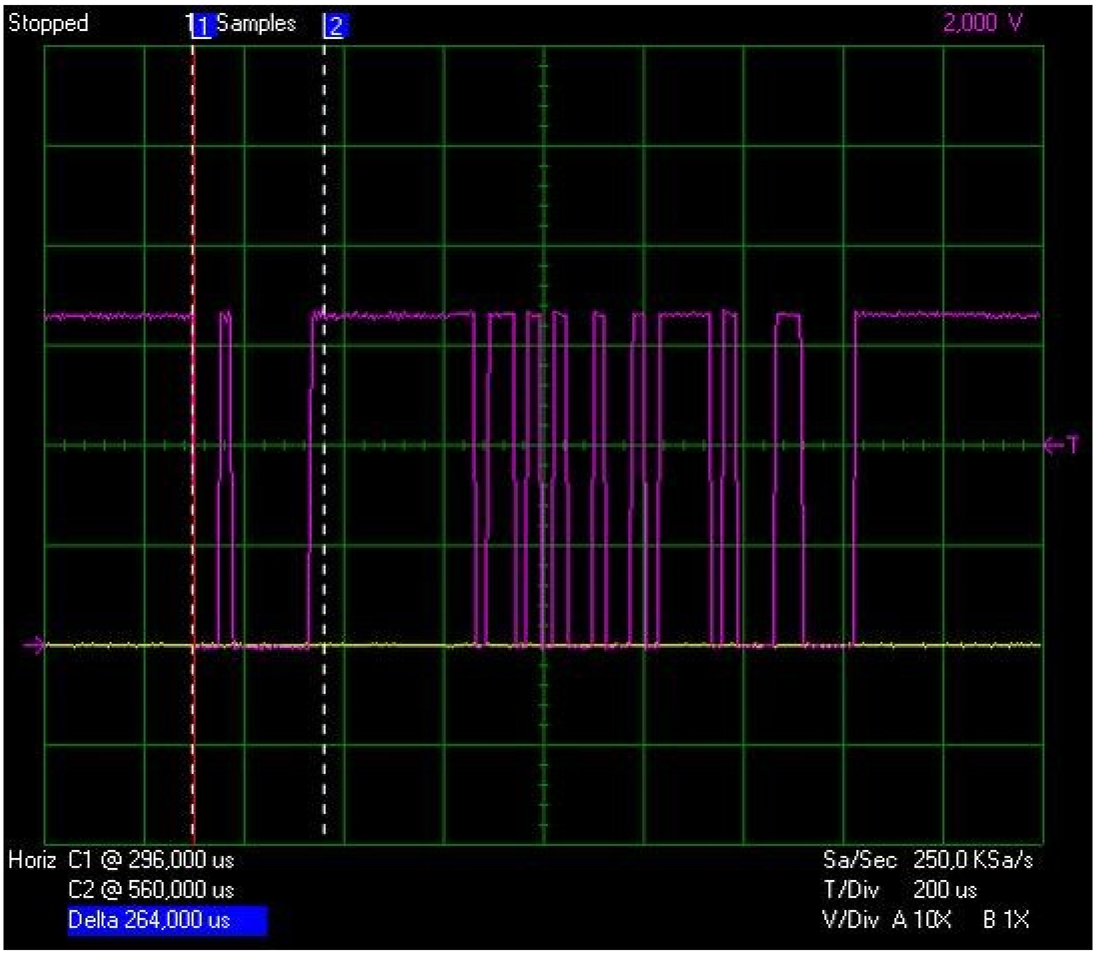
Copied from Stoeckli's blog (unit table, I have indicated precision when known):
01: V (one decimal)
02: A (one decimal)
03: m/s
04: km/h
05: rpm
06: °C
07: °F
08: m
09: % Fuel
10: % LQI (integer percentages)
11: mAh (integer value)
12: mL
13: <none>

Oscilloscope picture below of polling of address 02h and response 0=start bit, 1101 = B (unit = mAh), 0100 = 2 (address = 2), 1=end bit, 0=start bit, 0101 1110, 1=end, 0=start, 0011 0000. The four last nibbles start with the lsb=0 (no alarm). The rest of the bits equal 1597, resulting in a display value of 2: mAh 1597. Q.E.D.

The two values that the rx supplies (receiver voltage and LQI) are transmitted on the bus by the rx. There's no polling for addresses 0 and 1; instead the rx just writes the three byte structure onto the bus

(0 & 1 addresses assuming the rx sends the parameters on the default addresses. If you reprogram the addresses with a MULTImate, the addresses will or course be the regrogrammed ones)

I use the LQI data, and send it to the DAC in the PIC to emulate an RSSI signal. Works great!

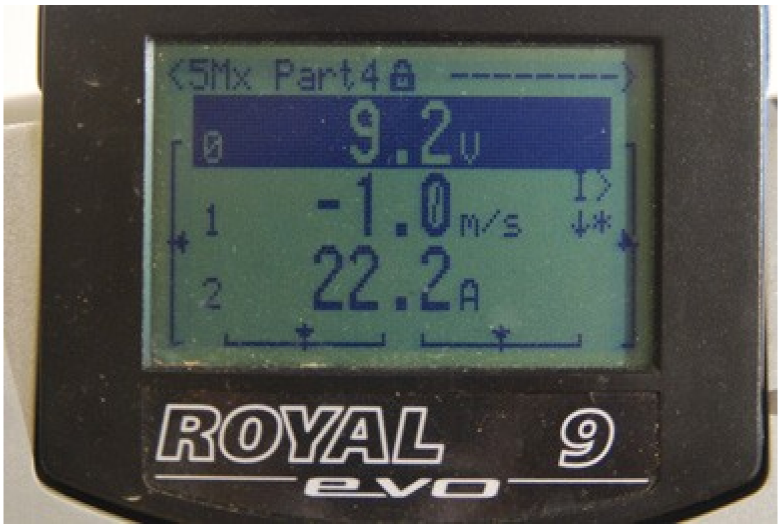


Lost and Found

This page contains content from a previous website which was not properly migrated to its new place. Please excuse the mess...

10/13/2009

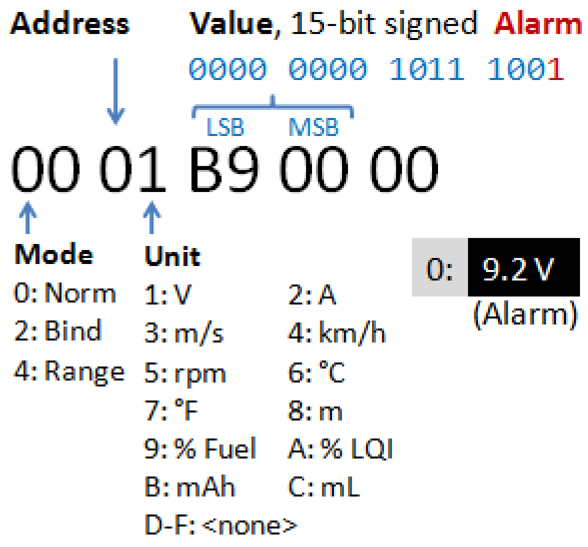
Multiplex Royal Pro (EVO) with Telemetry Display



After decoding the servo pulses from the EVO (see previous post), I started working on the telemetry data.
The TX module sends packages of 5 bytes. If no sensors are found, 00 FF 00 00 00 is sent. The use of the first and last byte is not fully known. The second byte is split into two nibbles. The higher one contains the sensor number, the lower one the unit. Unit symbols are:

01:	V
02:	A
03:	m/s
04:	km/h
05:	rpm
06:	°C
07:	°F
08:	m
09:	% Fuel
10:	% LQI
11:	mAh
12:	mL
13:	<none>

The next two bytes contain the actual data, 15-bit signed, LSB first and the alarm bit. When set, the display highlights the value and an alarm sounds.
The above sample was created by sending:
00 01 B9 00 00
00 13 EC FF 00
00 22 BC 01 00
When climb rate is shown (line two), one can activate an acoustic feedback (switch I). Two beeps high-high or high-low indicate climb or descent, and the pause between the signals gives an indication of the amount.
And here's a description of the full protocol:



How did I find this? I wrote a small C# application for my PC and instead of mounting a Rf-module, I hooked the connector over a USB-TTL converter (FTDI TTL-232-3V3) to the PC... Using this setup, the PC is emulating the Rf-module. Since this is independent of Multiplex M-Link, my EVO (or should I call it ROYALpro?) is now named S-Link ;-) (Thanks, Michael!)

Posted at 23:50 by Markus Stoeckli

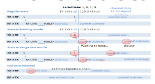
9/25/2009

Communication protocol between Multiplex ROYALpro and M-LINK module HFM4

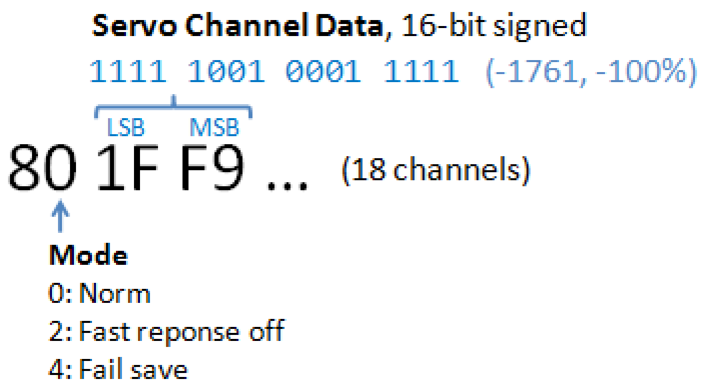
After publishing the communication between the transmitter and the MHz modules, I decoded the protocol with the M-LINK module HFM4. Here are the changes.
The module has a new identifier:

Then, the data is exchanged using the same protocol as previously described, but with a different scaling on the stick values.

The module itself sends 5 status bytes: 0,255,0,0,0
And here's the summary:



And below is a summary of the channel data (thanks to the comments by Yannick).



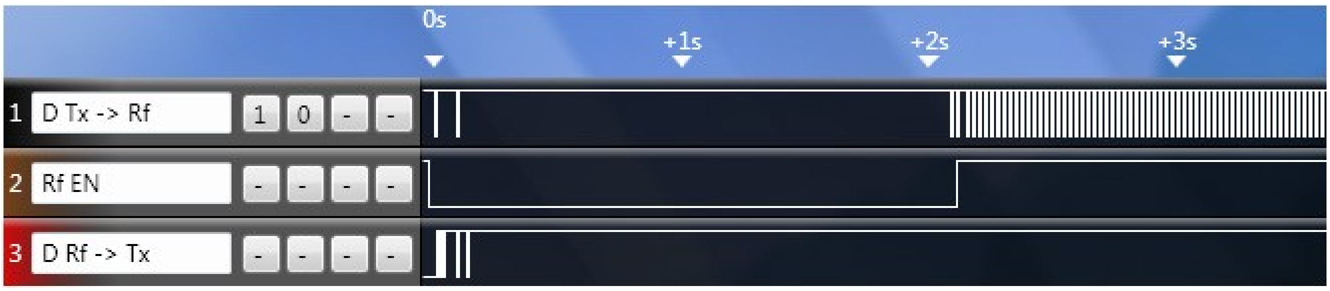
And these are the recorded files to be analyzed with the software from Saleae:
Switching on with a module bound: here
Switching on waiting to bind: here
Waiting and then binding: here
Posted at 16:15 by Markus Stoeckli

2/2/2009

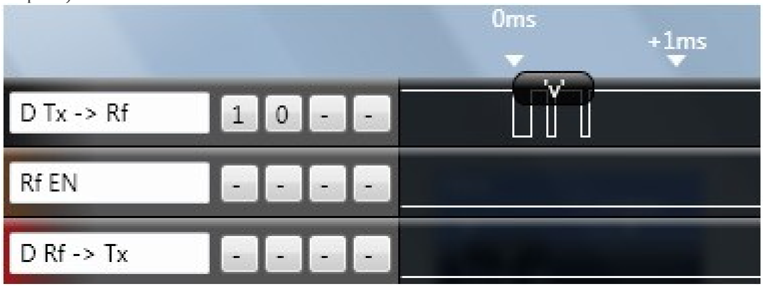
Communication Multiplex EVO to Rf

With the availability of the new 2.4 GHz transmitter modules from various vendors, I was wondering if it would be possible to directly get the channel data from the transmitter instead of reading out the PPM signal. I love my EVO, and so I chose to find more information about it. First thing on the list was a patent search... and I found DE 198 18 919 C 1. This German patent explains how the EVO and its RF module work together. Traditionally, the transmitter would directly send the PPM signal to the rf module, which would then takes this signal to directly modulate the rf. The EVO setup goes a different route. The transmitter send information to the rf module, which generates the modulation signal itself. The patent does not describe the protocol, but that was fairly easy:

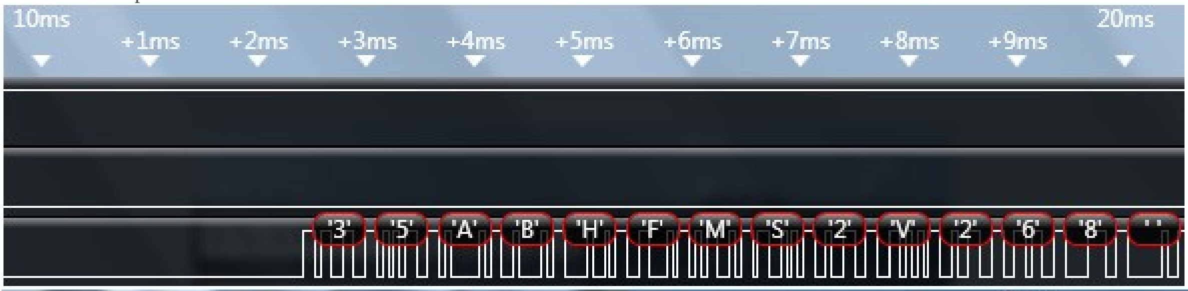
This image shows the overview of the signals (pin-out rf connector, from top: D Tx->Rf, Rf EN, Vcc, GND, D Rf->Tx, Rf)

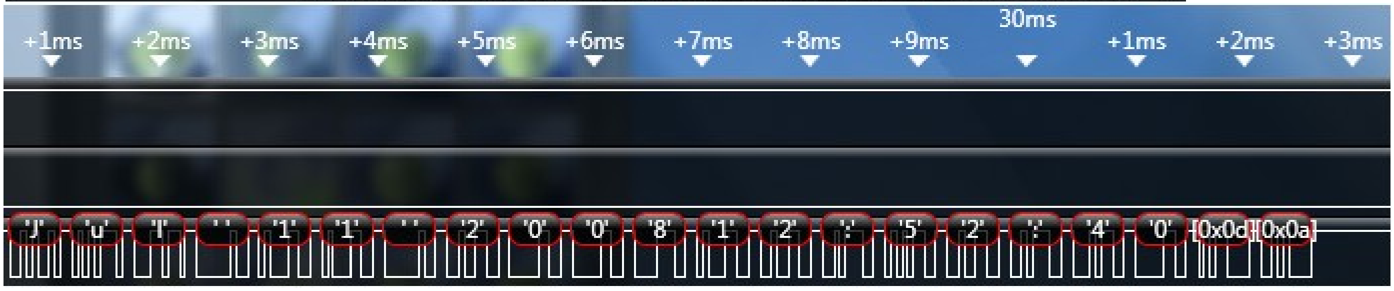


Now we zoom into the different regions. The transmitter sets Rf EN to low, and queries the rf module by a 'v', (19200 baud, 1 start bit, no parity, 1 stop bit).

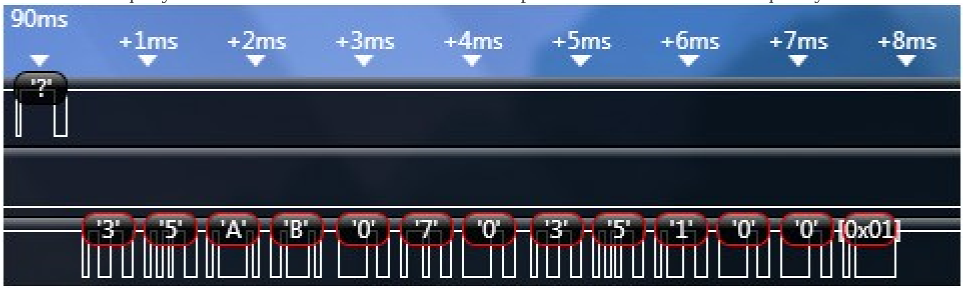


The module responds in clear text:




30ms
'J' 'u' 'l' ' ' '1' '1' ' ' '2' '0' '0' '8' '1' '2' ':' '5' '2' ':' '4' '0' [0x0d] [0x0a]

Next comes a query for the scanner with a '?'. The scanner reports the channel and the frequency...

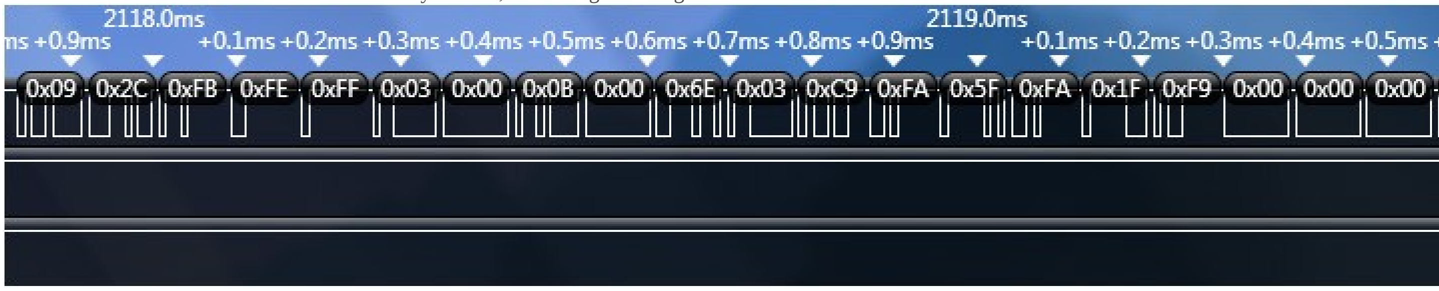


... followed by the measured intensity.

Now, the Tx requests rf by sending an 'a':

Which is promptly done:

The transmitter switches to 116 kBaud and sends the channel info package every 24.2 ms. The first byte contains the number of channels (e.g. 0x09), then the channel information is encoded in two bytes each, containing 12 bit signed values:


2118.0ms
ms +0.9ms
+0.1ms +0.2ms +0.3ms +0.4ms +0.5ms +0.6ms +0.7ms +0.8ms +0.9ms
2119.0ms
+0.1ms +0.2ms +0.3ms +0.4ms +0.5ms

```
0x09 0x2C 0xFB 0xFE 0xFF 0x03 0x00 0x0B 0x00 0x6E 0x03 0xC9 0xFA 0x5F 0xFA 0x1F 0xF9 0x00 0x00 0x00
```

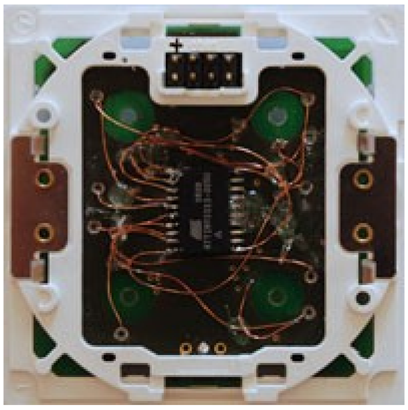
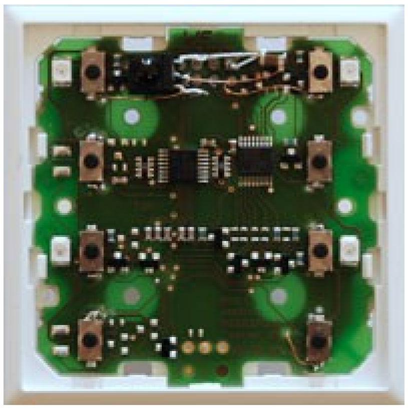
Simple as that...
If you want to analyze the data yourself, you may download this startup sequence recorded from a RoyalPro with Firmware 3.30 set to 9 channels, with ch1 set to -100, ch2 to +100 and the rest to 0. The software for the analysis you'll find here. And here you'll find the same data recorede with firmware 2.62.
Posted at 22:12 by Markus Stoeckli

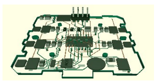
9/28/2008

Zeptrion IR Fernbedienung

OK, diese Projekt geht schon eine Weile zurück. Da es auch nur in der Schweiz wohnende betrifft, erfolgt die Beschreibung auch in Deutsch. Die Idee kam zustande, als ich einen Teil der Hauses auf Feller® Zeptrion™ umrüstete. Diese System erlaubt das einfache Einrichten von sogenannten "Szenen" für Beleuchtung und Storen. Mehr zum Hintergrund: Ich habe einen Beamer um TV zu schauen. Dieser leuchtet quer über den Raum, mit einer Lampe dazwischen. All meine A/V Geräte werden von einen Logitech® Harmony™Fernbedienung angesteuert, ausser eben die Lampen. Laut Herstellerangaben unterstützt die Harmony auch Zeptrion und Zeptrion unterstützt Szenen - also ideale Kombination. Ein Zeptrion Dimmer für jede Lampe ist schnell eingebaut, ein Zeptrion Storenschalter für den Motor der die eine Lampe nach oben bewegen soll und eine Zentralstelle mit IR Empfänger ergänzen das System. So weit so gut, ich kann die Hauptstelle so programmieren, dass ich eine Fernseh-Szene per Knopfdruck aktivieren kann (Hängelampe geht nach oben und schaltet aus, Stehlampe geht auf 10% Helligkeit). Aber dann kam die Ernüchterung: Der Zeptrion IR Empfänger unterstützt keine Szenen! Das System sieht viel mehr vor, einen IR-Empfänger auf jede Hauptstelle zu setzen und dann die Szenen auf dem IR-Sender zu programmieren. Aber dies kann die Harmony nicht...

Nach dem grossen Frust kam dann die Idee, die Frontplatte der Zentralstelle mit einem eigenen IF-Empfänger zu versehen. Von der Idee bis zur Lösung lagen dann zwei Tage und nun läuft mein Zeptrion problemlos mit der Harmony. Interessiert? Schreibt einen Comment und ich publiziere die Details... bin bis jetzt einfach noch nicht dazu gekommen... Also hier schon mal ein paar Bilder vom Prototypen und dem neuen PCB:


Posted at 19:43 by Markus Stoeckli