Modbus UDP Specification

@version@ (@date@)

by Dieter Wimberger

1. About

This document introduces a Modbus/UDP flavor which we have developed to verify if the Modbus protocol is suited for the use of UDP/IP as lower level communication stack as the protocol:

1. is stateless,
2. transaction oriented
3. and the package size is limited to 256 bytes, which should be easily transferable over any IP capable link (including IP over Serial) without the necessity to split the package.

We succeeded with a relatively simple design and implementation and are convinced that a Modbus/UDP flavor is an interesting approach for reducing the traffic overhead, achieving high throughput performance.

2. Protocol & Specification

That's the beauty of the thing, basically we have **not changed anything** of the Modbus application level (and TCP) specification but how messages are transported.

This means that the IP specific header (called MBAP in the specification) is exactly the same as for Modbus/TCP. It's 7 bytes long and composed of the following fields:

1. the *invocation identification* (2 bytes) used for transaction pairing; formerly called *transaction identifier*
2. the *protocol identifier* (2 bytes), is 0 for Modbus by default; reserved for future extensions
3. the *length* (2 bytes), a byte count of all following bytes
4. the *unit identifier* (1 byte) used to identify a remote unit located on a non-TCP/IP network

Also nothing has changed about possible network setups. They are not really by the specification; it is possible to setup multi-master systems or realize bidirectional communication (i.e. have nodes that are master and slave at the same time). However, the

user should be well aware that there are implications from deviations of the Master/Slave schema.

3. This is a joke right?

Definately not. It is and has been as easy as this and it works great.

So what is new?
Well, only the implementation. Messages are unicast between nodes on the network, with two packages (a *Request* and a *Response* package/message) per transaction.

And what about UDP being an **unreliable** transport?
Well, the protocol is transaction oriented. If a package get's lost, the transaction will not be successfully executed. Thus, the master will know and can decide whether to retry it (a given number of times for example) or give up.

I still don't believe it?!
Right. That's why we have implemented it and added some examples to the *jamod* library. Please see yourself:

- Modbus/UDP Slave: UDP Slave How-To
- Modbus/UDP Master: UDP Master How-To

You are also more then welcome to check out the source code.