Modbus BIN Specification

@version@ (@date@)

by Dieter Wimberger

Table of contents

1. About

This document introduces a Modbus/BIN (serial) flavor which we have developed to overcome the problems of the RTU frames that are purely based on timings.

We succeeded with a relatively simple design and implementation and are convinced that a Modbus/BIN flavor is an interesting approach for reducing the traffic overhead, achieving higher throughput performance than with the ASCII flavor but without the drawbacks of the RTU flavor.

2. Protocol & Specification

Basically the protocol specification for the RTU flavor is valid, except for the following two major differences:

1. Message frames have start and end tokens.
2. Start and end token bytes are duplicated if they occur within a message.

Both of these differences will be discussed in more detail in the following.

3. Modbus/BIN Frames

Frames are transmitted binary to achieve a higher density. The error checksum is represented by a cyclic redundancy check (16 bit CRC; 2 byte) and appended to the message in the RTU order (low byte first). A message frame starts with a left/opening brace ({, ASCII Code: 123, hex 0x7B) and ends with a right/closing brace (}, ASCII Code: 125, 0x7D). Pauses between bytes can occur.

Wait, the message might contain bytes with these values!
Right. That's why we have additionally established a rule for writing message bytes. Please see the next section.

4. Modbus/BIN byte transport rule

Due to the fact that message bytes might have the value of the frame start and end tokens, we establish a very simple rule for transferring message bytes over a transport medium:
If a byte with the value 123 (0x7B) or with the value 125 (0x7D) occurs it has to be duplicated (i.e. written twice)..

This allows to distinguish between frame marker tokens and byte values when reading them from a transport medium, without much effort.

Wait, this means you might as well end up with the length of messages in ASCII flavor!

This is basically right, but very unlikely the case. We are convinced that we have probability on our side. If you consider that each byte value has the same probability you can calculate the density distribution for a specific message length and statistically speaking, the mean length will be below the length of the ASCII flavor message.

Wait, I don't believe the statistical crap!
Creationist, uh? Well, probably we convince you with these:

1. You don't need to worry about exact timing that much any longer. This is great at least if you don't have to cope with a strict hard real time world.
2. You won't loose time or resources to encode bytes, neither when writing them (you just write the same byte once more) nor when reading them (you just check for the duplicates and skip them).

I want to try this!
Right. That's why we have implemented it. All you need to do is to set the encoding parameter for the serial examples to `bin`. jamod will do the rest for you.

- Modbus Serial Slave: Serial Slave How-To (../development/serial_slave_howto.html)
- Modbus Serial Master: Serial Master How-To (../development/serial_master_howto.html)

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