Interfacing a Modbus/TCP system

Abstract:

Modbus is an old protocol introduced in 1979 by the PLC manufacturer Modicon. A PLC is a kind of first generation microcontroller.

The modbus protocol became a success because it was easy to use and could be applied to many applications. It continues to be standard in industrial controllers and is still used in industrial and home automation systems.

More information about Modbus protocol can be found at:
http://www.modbus.org/specs.php

Modbus is essentially a master slave protocol that can be used to read and write registers in some device. Initially it was made for systems that run on RS232 communication. Today many devices talk Modbus/TCP which is the Modbus protocol embedded into a TCP/IP connection.

We use the tuxgrahics avr ethernet board as a TCP/IP client talking to a Modbus/TCP server.

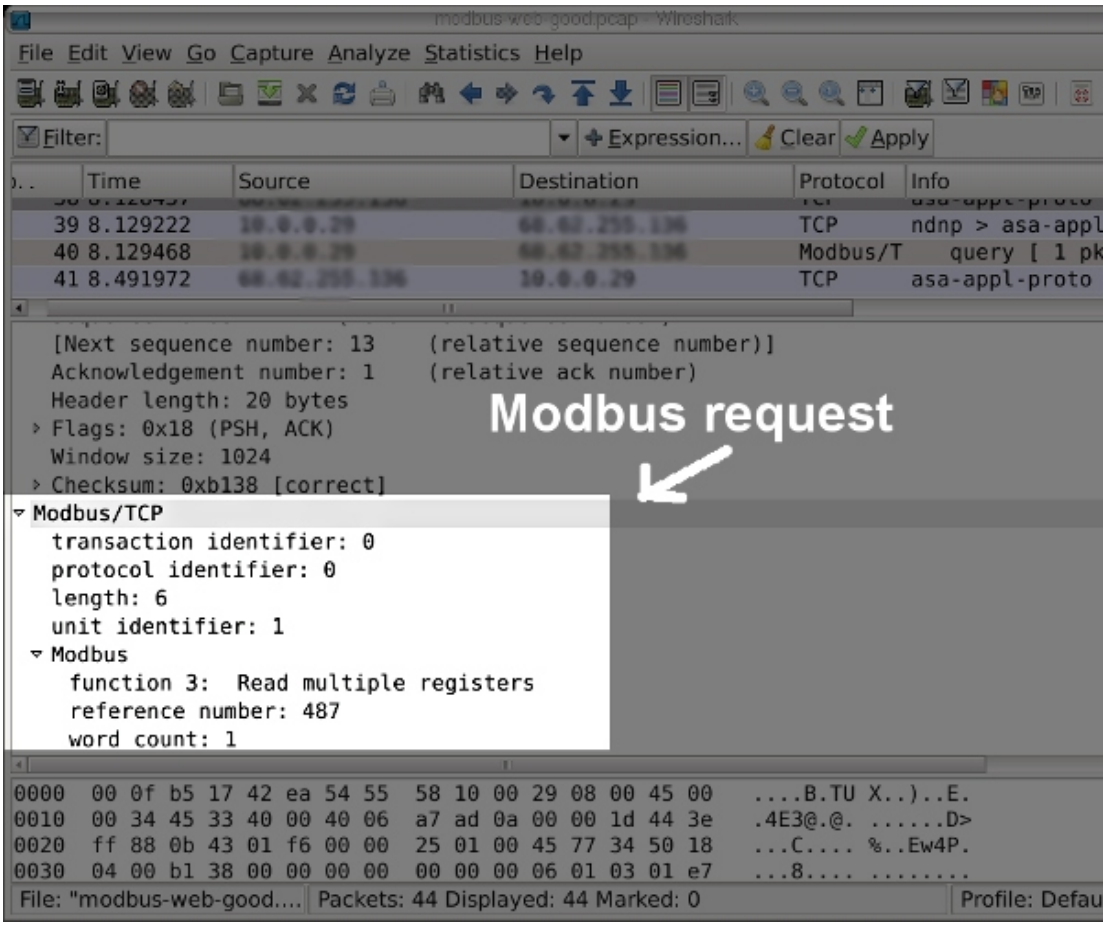
How Modbus/TCP works

Modbus/TCP places a special version of the Modbus protocol into a tcp/ip communication. Port 502 is used on the server side for this purpose. The Modbus protocol consists of a header part and a data part. The fields in the header part are:

- 2 byte transaction identifier
- 2 byte protocol identifier (always zero, 0=Modbus)
- 2 byte length, number of bytes following
- 1 byte unit identifier, ID of the remote slave connected on a serial line o on other buses reachable via the Modbus/TCP server.

The actual data part of a query consists then of commands to read/write a given single bit or a number of 16 bit registers.

The project presented here is a TCP/IP client. In other words we implement a Modbus/TCP client that talks to a Modbus/TCP server. The Modbus/TCP server has the actual devices attached where we want to read/write data.



A Modbus/TCP request decoded in wireshark

The example: A web to Modbus/TCP gateway

There is no general purpose application for such a Modbus/TCP client. It is useful if you have some automation equipment that appears as a Modbus/TCP server and you want to interface to it. Maybe you want to build your own web-interface to it or you have some other application in mind that you want to run on the tuxgraphics ethernet board and it should talk to this automation equipment.

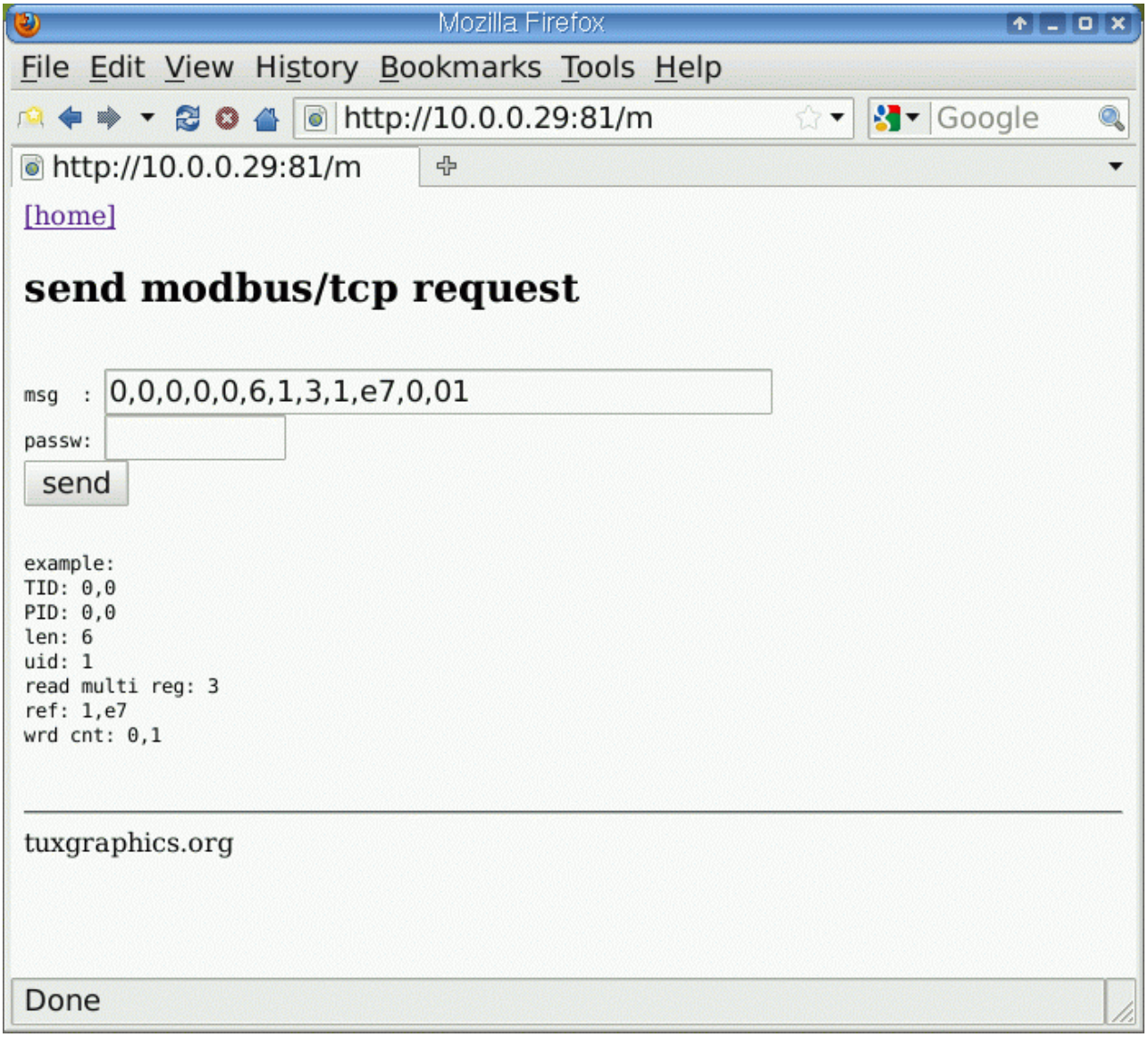
In any case the code in the download section is an example and you can integrate it into your own ideas. Sending a query out of the tuxgraphics ethernet board works in 3 simple steps:

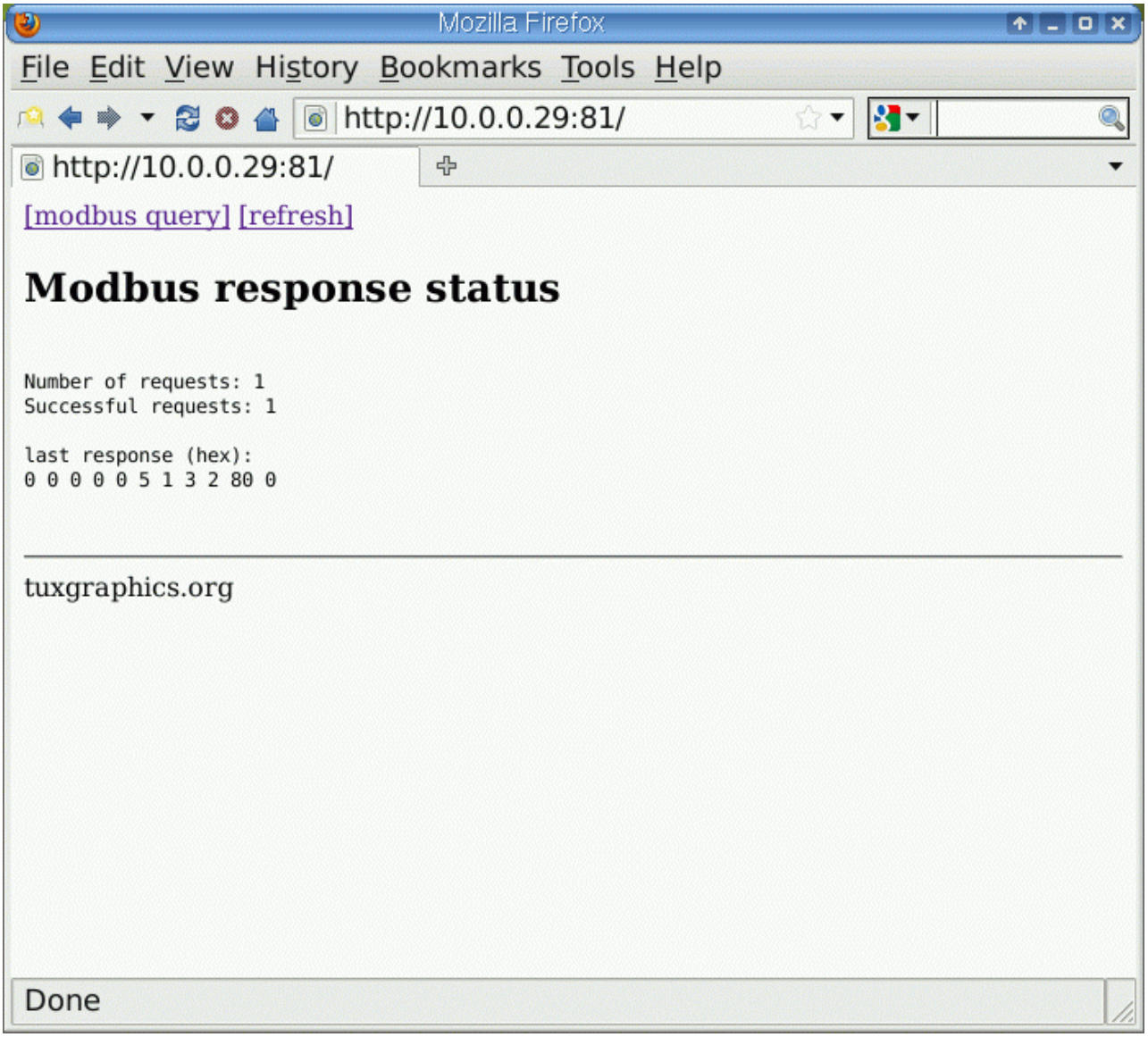
1. You call function client_tcp_req like this:
   client_tcp_req(&modbusresult_callback,&modbus_datafill_callback,502);
   This starts a TCP transaction.
2. The function modbus_datafill_callback will then be called when it is time to fill in the modbus/tcp query data. This happens normally a few milliseconds after you called client_tcp_req.
3. The function modbusresult_callback will be called when the answer from the server is back and you get the result passed to this function.

The callback functions are a good way to integrate any user code into the state-machine of a communication protocol. They save storage memory and allow a direct interaction with the protocol flow. Think of them like interrupt routines which can be triggered after you called client_tcp_req.

Open file main.c to see how this works.

This example application from the download section is a web to modbus/tcp gateway. It allows you to enter an arbitrary hex string which will be sent out as modbus/tcp query and it displays the answer (in hex).





References/Download

- Download section. The software archive contains a README.htm file with instructions on how to adapt the software to your network.
- The tuxgraphics ethernet board: http://shop.tuxgraphics.org/electronic/index-eth.html

2010-02-20, generated by tuxgrparser version 2.57