RAPIDOSCAN® CLASSIC

Modbus TCP + Ethernet | EN

Version 0.14

© 2015 - 2018

dieEntwickler Elektronik GmbH

Linzer Straße 4, 4283 Bad Zell / AUSTRIA

Telefon: +43 7263 20900-0, Telefax: +43 7263 20900-4

office@dieentwickler.at, www.dieentwickler.at

Table of Content

1 Preface 4

2 Functional Overview 5

2.1 Problem: Real-Time Capability with Modbus Protocol 5
2.2 Approach: Expanding of Modbus TCP with additional functions 5
2.2.1 Additional AutoSend-Channel 5
2.2.2 Sending of Multiple Results as One Package 5
2.2.3 Optional Control of the AutoSend-Channel via a Hardware Line (gate) 6
2.2.3.1 Available Inputs for Gating 6
2.2.4 What data can be sent using the AutoSend Channel? 7

3 Parameterization 8

3.1 Modbus-Specification 8
3.1.1 Data Encoding 8
3.1.2 Unit Identifier 8
3.1.3 Function Codes 8
3.1.3.1 Read Holding Registers (0x03) 9
3.1.3.2 Read Input Registers (0x04) 9
3.1.3.3 Write Multiple Registers (0x10) 9
3.1.4 Register Addresses 9
3.1.4.1 Input Registers 9
3.1.4.1.1 Process data 9
3.1.4.1.2 Consistent pre calculated data 10
3.1.4.1.3 Consistent pre calculated holding data 10
3.1.4.1.4 Device Information 11
3.1.4.1.5 Manual Teach 11
3.1.4.2 Holding Registers 11
3.1.4.2.1 Basic setup 12
3.1.4.2.2 Common Teach settings 12

3.1.4.2.3 Manual Teach settings 12
3.1.4.2.4 Physical Scanarea 13
3.1.4.2.5 Digital I/O 13
3.1.4.2.6 WLAN Wifi Configuration 15
3.1.4.2.7 LAN Ethernet Configuration 16
3.1.4.2.8 ModbusTCP Configuration 16
3.1.4.2.9 AutoSend Channel Configuration 16
3.1.4.2.10 Mapping 18
3.1.4.2.11 Commands | Triggering of Actions 18
3.1.4.2.12 Command Status | Results of Actions 19
3.1.4.2.13 Triggering of Actions 20
3.1.5 Deviations from the Modbus specification 20

4 Testing the Modbus TCP Functionality 21

4.1 Connecting to the integrated Web-Server 21
4.2 Configuration of the LAN Ethernet Setting 21
4.3 Configuration of Modbus TCP Setting 22
4.4 Doing a Test-Run with QModMaster 23
4.5 Connect to SIMATIC S7-1200/S7-1500 25
4.5.1 Register Addresses 25

5 Testing the AutoSend Functionality 25

5.1 Connecting to the integrated Web-Server 25
5.2 Configuration of the LAN Ethernet Setting 26
5.3 Configuration of the AutoSend Channel 26
5.4 Doing a Test-Run with Docklight Scripting 29
5.5 Doing a Test-Run with Wireshark 30

1 Preface

The goal of this document is to describe the options available, which can be derived from using the slow but well standardized Modbus TCP protocol together with an additional data channel over which automatically pre-parameterized UDP or TCP packets can be sent.

The great advantage of Ethernet based solutions is to operate the Modbus link and the so called "Auto Send " channel parallel on the same line or on the same connector without leaving the Modbus specifications !

The Modbus protocol itself will not be discussed here further. The specification is available at www.modbus.org. Its underlying structure is briefly mentioned:

The Modbus protocol is a communication protocol based on a master / slave architecture. The device which is requesting the data is referred to as Modbus Master. The device which provides the data is called the Modbus Slave.

Additionally together with Modbus TCP the terms client / server also are used.

Modbus Master = Modbus TCP Client

Modbus Slave = Modbus TCP Server

2 Functional Overview

2.1 Problem: Real-Time Capability with Modbus Protocol

Using the Modbus protocol the RapidoScan can be fully parameterized. Of course the measurement results can be read via Modbus telegrams too. But because of the master - slave principle of Modbus a controlled polling by a master unit is necessary.

Therefore, to transmit a measurement result via Modbus an additional query telegram is required. This not only means more telegrams but it also means that some mechanism must be implemented in the Modbus master to do so cyclically. And this might become a timing and performance problem in a control system as the update rates are getting higher.

2.2 Approach: Expanding of Modbus TCP with additional functions

The idea is that the measured data will not only be made available on polling but it will also be supported to automatically send them from a slave to a master.

If this mechanism would be used together with RS485/UART interfaces it would be necessary to expand the Modbus protocol in an improper way. And if working with a normal Modbus stack one would be faced with several problems on implementing these non standardized features.

2.2.1 Additional AutoSend-Channel

Not so when using an Ethernet connection. By using the Ethernet Modbus TCP variant multiple connections/channels can be active which will not influence each other (with the exception of the bus load).

Therefore this approach is implemented in RapidoScan. On the one hand the RapidoScan communicates via standardized Modbus TCP telegrams. But moreover an additional port can be opened which allows communication independently from the Modbus link what we call the AutoSend channel. This second channel adds the opportunity for automatically sending of results independently of the Modbus connection. On the receiving side the only task is to fetch the incoming data from the receive buffer.

To meet different customer requirements the AutoSend channel can be parameterized within wide ranges. It is also possible to choose the communication setup behaviour.

After having connected the AutoSend channel the AutoSend functionality is enabled and the RapidoScan starts self-sending its preconfigured data.

As long as the connection is held active the measurement data will be transmitted cyclically with each single measurement or with each complete package of multiple measurements depending on its configuration.

The AutoSend channel works completely independent of the Modbus TCP functionality. If the configuration is done using the integrated web server of the RapidoScan the Implementation of Modbus TCP can be even completely eliminated.

2.2.2 Sending of Multiple Results as One Package

Particularly noteworthy is the ability of combining multiple measurement results into one package.

The cyclic transmission of results directly after a measurement can arise to a greater burden for the Programmable Logic Controller (PLC). RapidoScan variants having only a few rays will give small data packages with a very high update rate. And all that data must be fetched and copied within the PLC at short intervals.

For these situations the RapidoScan offers the helpful feature of combining the data of multiple measurements to a bigger data package. The package can then be sent automatically when it has reached a preset number of measurements for example or on triggering a dedicated input line.

© 2015 dieEntwickler Elektronik GmbH

In general each individual measurement in RapidoScan is provided with a time stamp. Using the timestamp it does not matter whether the results are instantaneously passed as individual results or summarized in the form of a packet. The timestamp ensures a precise reproduction of the measurement data as a line camera image.

2.2.3 Optional Control of the AutoSend-Channel via a Hardware Line (gate)

Another variant of controlling the AutoSend data stream is carried out via a hardware line. That functionality can be optionally enabled and if so this hardware line is then working as a gating signal.

Gating means that the data flow to the PLC can be enabled (= RUN) or stopped (= PAUSE).

On changing the gating signal from RUN to PAUSE the existing contents of the data buffer will be output even if there is not all data present at the package.

During PAUSE no further information will be generated and no data will be transferred to the buffer.

If any input is setup as gating line then an unconnected gating line acts as a PAUSE.

Additional option "Fill gating frame"

If e.g. your PLC uses a fix receive data length, this option fills up the last frame (RUN -> STOP).

With this option all frames will be extended with a 2 byte header

[Bit 0-14]: payload length

[Bit15]: 1, if this is the last frame (fill up with CCh), otherwise 0

2.2.3.1 Available Inputs for Gating

Input signal DIGITAL_IN

Fig.: 2-1

<u>Mode Trigger In:</u>

Normally used in cascaded applications

<u>Autosend Gating</u>

DIGITAL_IN 0V: PAUSE

DIGITAL_IN 24V: RUN

Autosend Reset

Similar to Autosend Gating, but with the rising edge the time stamp is set to = 0.

2.2.4 What data can be sent using the AutoSend Channel?

The RapidoScan allows setting up the AutoSend output data by filling a list of wanted data objects (mapping area). On enabling AutoSend these data objects will then be cyclically copied into the output buffer.

Both the mapping area and the output buffer itself are data areas that lie within the Modbus address range. So all data can be transferred using the AutoSend functionality or using the Modbus protocol whatever you prefer.

Mostly the data objects are setup using Modbus and then the AutoSend function becomes enabled to transfer data using the standard Ethernet link. As the timestamp is also a data object you are free to include the timestamp to your output data.

© 2015 dieEntwickler Elektronik GmbH

3 Parameterization

3.1 Modbus-Specification

The latest information on the Modbus protocol and its specification can be accessed at http://www.modbus.org/.

3.1.1 Data Encoding

The Modbus specification defines that for data and addresses the "big-endian" representation is used. The MSB (most significant byte) is thus the first byte to be transmitted.

Example: 16-bit value: 0x1234 is transmitted as 0x12 first and 0x34 second.

The RapidoScan adheres to this requirement.

3.1.2 Unit Identifier

This field is used for routing purpose when addressing a device on a MODBUS+ or MODBUS serial line sub-network. In that case, the "Unit Identifier" carries the MODBUS slave address of the remote device:

If the MODBUS server is connected to a MODBUS+ or MODBUS Serial Line sub-network and addressed through a bridge or a gateway, the MODBUS Unit identifier is necessary to identify the slave device connected on the sub-network behind the bridge or the gateway. The destination IP address identifies the bridge itself and the bridge uses the MODBUS Unit identifier to forward the request to the right slave device.

The MODBUS slave device addresses on serial line are assigned from 1 to 247 (decimal). Address 0 is used as broadcast address.

On TCP/IP, the MODBUS server is addressed using its IP address; therefore, the MODBUS Unit Identifier is useless. The value 0xFF has to be used.

Remark: The value 0 is also accepted to communicate directly to a MODBUS/TCP device.

3.1.3 Function Codes

The RapidoScan supports the following function codes:

Object Type		***Funktion Code***		***Comment***
Access	***Area***	***[Decimal]***	***[Hex]***	
16 Bit	Holding Registers	03	03	Read Holding Registers
16 Bit	Input Registers	04	04	Read Input Register

16 Bit	Holding Registers	16	10	Write Multiple Registers

3.1.3.1 Read Holding Registers (0x03)

This function code is used to read the contents of a contiguous block of holding registers. The Request PDU specifies starting register address and the number of registers to be read.

3.1.3.2 Read Input Registers (0x04)

This function code is used to read (up to 125) input registers. The Request PDU specifies starting register address and the number of registers to be read. Here typically the process data (=measuring data) will be read.

3.1.3.3 Write Multiple Registers (0x10)

This function code is used to write a block of contiguous registers. The Request PDU specifies starting register address and the number of registers to be read. The data to be written must be specified in the request data field.

3.1.4 Register Addresses

In RapidoScan the data is arranged as a large data block. This means that data can be accessed using different function codes (see http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf - Chapter MODBUS Data Model).

In addition to numerical prefixes, some documentation will refer to protocol addresses (addresses start at 0), while other documentation will refer to data model addresses (addresses start at 1). That is, the first holding register may be 0 or 1 (or 40000 versus 40001 using prefixes). However, this has no bearing on what gets sent over the wire as a Modbus message. For a Modbus protocol message, the lowest address is always "0", not "1".

If you use a SIMATIC S7-1200/S7-1500, please note the remarks at 4.5.1.

This document uses standard Modbus terms and addresses throughout using Modbus protocol addressing without prefixes or offsets (direct addressing scheme).

The RapidoScan works with an object data model. So, read/write access is limited to the address range of the object. The objects are marked with

3.1.4.1 Input Registers

The input register table is used for read-only access to device information, system status and measurement data (process data).

3.1.4.1.1 Process data

Starting Address	***Mask***	***Description***

0x0000 – 0x04AF	-	Analog Signal Strength Values of received infrared signals – 8 Bit for every single beam
0x0A00 – 0x0A95	-	Binary Data Output – bitwise for every single beam

3.1.4.1.2 Consistent pre calculated data

Starting Address	*Mask*	*Description*
0x0B00 – 0x0B01	-	Timer value of measured data on starting the measuring cycle
0x0B02 – 0x0B03	-	Counter for number of measuring cycles incremented on every cycle
0x0B04	-	First Interrupted Beam
0x0B05	-	First NOT Interrupted Beam
0x0B06	-	Last Interrupted Beam
0x0B07	-	Last NOT Interrupted Beam
0x0B08	-	Total number of Interrupted Beams
0x0B09	-	Total number of NOT Interrupted Beams
0x0B0A	-	First Interrupted Beam of Major Data Block
0x0B0B	-	Total number of Interrupted Beams of Major Data Block
0x0B0F - 0x0B10	-	Rotary Encoder Value

3.1.4.1.3 Consistent pre calculated holding data

Starting Address	*Mask*	*Description*
0x0FF0	-	First Interrupted Beam
0x0FF1	-	First NOT Interrupted Beam
0x0FF2	-	Last Interrupted Beam
0x0FF3	-	Last NOT Interrupted Beam
0x0FF4	-	Total number of Interrupted Beams
0x0FF5	-	Total number of NOT Interrupted Beams

© 2015 dieEntwickler Elektronik GmbH

3.1.4.1.4 Device Information

Starting Address	*Mask*	*Description*
0x2000 – 0x2001	-	Version number of register table definition
0x2002 – 0x2011	-	Vendor Name
0x2012 – 0x2031	-	Vendor Text
0x2052 – 0x2053		Update Package version number

Starting Address	*Mask*	*Description*	
0x2400	Low Byte	Device Error	
		[Bit 0]	1: Generic error pending
		[Bit 1]	1: Startup active
		[Bit 2]	1: System error pending
		[Bit 3]	1: Receiver power supply error pending
		[Bit 4]	1: Transmitter power supply error pending
	High Byte	Device Status	
		[Bit 0]	1: Device configuration isn't persistent
		[Bit 1]	1: Firmware update active
		[Bit 2]	1: Teach Required
0x2401	-	Beam Distance	
0x2402	-	Diode Count	
0x2403	-	Beam Count	

3.1.4.1.5 Manual Teach

Starting Address	*Mask*	*Description*
0x2500	-	Receiver gain level count

3.1.4.2 Holding Registers

… used for read/write access to device configuration and for triggering actions.

3.1.4.2.1 Basic setup

Starting Address	*Mask*	*Description*
0x4000		Beam Sequence 0: Up 1: Down
0x4001		Beam Mode 0: Parallel 1: Diagonal 2: Crossed
0x4002		Mirrored Mode 0: Direct 1: Mirrored
0x4006		Trigger Mode 0: Deactivated - free running 1: Slave 2: Master

3.1.4.2.2 Common Teach settings

These settings are taken over by any teach command (see 3.1.4.2.11)

Starting Address	*Mask*	*Description*
0x4100	Low Byte	signal strength for binary 1 (threshold)
0x4101	Low Byte	signal strength for binary 0 (threshold)

3.1.4.2.3 Manual Teach settings

These settings are taken over by a manual teach command (see 3.1.4.2.11)

Starting Address	*Mask*	*Description*
0x4103		Receiver gain level index (see 3.1.4.1.5)
0x4104		common value for the transmit/emitter power

3.1.4.2.4 Physical Scanarea

Starting Address	*Mask*	*Description*
0x4200	Low Byte	Config 0: Full Area 1: Restricted Area
0x4201		Offset
0x4202		First Beam
0x4203		Last Beam
0x4204		Resolution

3.1.4.2.5 Digital I/O

Since Digital Outputs 1 and 2 share their two lines with the encoder inputs the functionality of the Digital Outputs 1 and 2 become disabled if encoder is enabled independent of their prior settings. The encoder setup overrules the Digital Output setup for these two outputs!

For more details please refer to the manual "RapidoScan, manual.pdf" chapter "Discrete I/O".

3.1.4.2.5.1 Digital Output 1

Only available if Encoder mode (3.1.4.2.5.5) is disabled! Otherwise these settings are ignored.

Starting Address	*Mask*	*Description*
0x4040	Low Byte	Mode 0: Disabled 1: Push-Pull 2: PNP open collector 3: NPN open collector
0x4041	Low Byte	Function 0: First beam 1: Last beam 2: Any interruption 3: Special beam number 4: Toggle
0x4042	-	Beam number, if function 3 (Special beam number) is selected

3.1.4.2.5.2 Digital Output 2

Only available if Encoder mode (3.1.4.2.5.5) is disabled! Otherwise these settings are ignored.

Starting Address	***Mask***	***Description***
0x4060	Low Byte	Mode 0: Disabled 1: Push-Pull 2: PNP open collector 3: NPN open collector
0x4061	Low Byte	Function 0: First beam 1: Last beam 2: Any interruption 3: Special beam number 4: Toggle
0x4062	-	Beam number, if function 3 (Special beam number) is selected

3.1.4.2.5.3 Digital Output 3

Starting Address	***Mask***	***Description***
0x4080	Low Byte	Mode 0: Disabled 1: Push-Pull
0x4081	Low Byte	Function 0: First beam 1: Last beam 2: Any interruption 3: Special beam number 4: Toggle 240: Customized
0x4082	-	Beam number, if function 3 (Special beam number) is selected

3.1.4.2.5.4 Digital Input 0

Starting Address	***Mask***	***Description***

0x40A0	Low Byte	Mode 0: Disabled 1: Trigger In - normally used in cascaded applications 2: Autosend Gating 3: Autosend Reset

3.1.4.2.5.5 Encoder

Starting Address	***Mask***	***Description***
0x40C0	Low Byte	Configuration
		[Bit 0] Encoder Feature 0: Disabled 1: Enabled This setting changes the physical direction of X2.2/ X2.4! This change requires a restart. Check physical connection to X2.2/ X2.4 first!
		[Bit 1] Encoder pulses counting direction 0: normal 1: inverted

3.1.4.2.6 WLAN Wifi Configuration

Starting Address	***Mask***	***Description***
0x4500 – 0x4501	-	IP-Address (host byte order). If DCHP is enabled, the DCHP server uses the range from .20 to .254.
0x4502 – 0x4503	-	Standard gateway (host byte order). (not supported yet)
0x4504 – 0x4505	-	Subnet mask (host byte order). (not supported yet)
0x4506	Low Byte	Flags
		[Bit 0] 0: Disabled 1: Enabled

		[Bit 1] 0: static (not supported yet 1: DHCP
0x4507 – 0x4522	-	SSID

3.1.4.2.7 LAN Ethernet Configuration

Starting Address	Mask	Description
0xC000 – 0xC001	-	Static IP-Address (host byte order). This change requires a restart for ModbusTCP connections.
0xC002 – 0xC003	-	Standard gateway (host byte order).
0xC004 – 0xC005	-	Subnet mask (host byte order).
0xC006	Low Byte	Flags
		[Bit 0] 0: static 1: DHCP (not supported yet)

3.1.4.2.8 ModbusTCP Configuration

Starting Address	Mask	Description
0xC040	-	Port number to be checked for incoming connections This change requires a restart.
0xC041	Low Byte	Flags (not supported yet – fix Ethernet)
		[Bit 1-0] Network interface binding 10: Ethernet 01: WLAN 11: any address

3.1.4.2.9 AutoSend Channel Configuration

Beside of the general enabling and disabling of this functionality it has to be defined:

- Who is setting up the communication?
  - Client Mode at the RapidoScan means that it is set up by the RapidoScan
  - Server Mode means that the communication must be set up by the PLC or PC.
- What protocol is to be used?
  - TCP

- UDP
- IP-Address of the communication partner if the RapidoScan has been defined to set up the communication (client mode)
- Port Numbers

Starting Address	***Mask***	***Description***	
0xC080	Low Byte	Basic Settings	
		[Bit 0]	AutoSend Function 0: Disabled 1: Enabled
		[Bit 1]	On connection setup the RapidoScan behaves like a … 0: Client 1: Server
		[Bit 2]	Transport Layer 0: TCP (requires Server Mode) 1: UDP (requires Client Mode)
		[Bit 7]	Fill Gating Frame 0: Disabled 1: Enabled
0xC081 – 0xC0082	-	Client-Mode	IP-Address of the Server (i.e. PLC) which needs to be connected to.
		Server-Mode	Not relevant – the connection will be set up by the PLC.
0xC083	-	Client-Mode	Port Number which the Server (PLC) checks for incoming connections and datagrams.
		Server-Mode	Port number to be checked by RapidoScan for incoming connections.
0xC084	Low Byte	Client-Mode	Not relevant.
		Server-Mode	Maximum count of connections to be supported by RapidoScan in parallel. So more than one PLC can receive the same data automatically. Not supported yet – must be set to 1
0xC085	-	Number of measurement results that make up a data block before sending out the block.	

After having changed one of these registers please SAVE these registers and re-power the RapidoScan to make them effective!

3.1.4.2.10 Mapping

The chosen approach is based on known solutions from other Field-Bus protocols. In practice it is very similar to the one used with CANopen.

The RapidoScan provides a variety of process data (see 3.1.4.1.1, 3.1.4.1.2a).

Using mapping you can select the needed process data which then will be provided in a specific address space.

Start Address	***Mask***	***Description***
0xA000 + Offset 0	-	Number of data objects to be mapped
0xA000 + Offset 1	-	Object 1: Starting Register Address
0xA000 + Offset 2	-	Object 1: Object Length
0xA000 + Offset 3	-	Object 2: Starting Register Address
0xA000 + Offset 4	-	Object 2: Object Length
…		…

For setting up the mapping configuration please use following order:

1. AutoSend->Config: Disable AutoSend feature first
2. Mapping->Offset 0: Clear the number of mapped objects (set to 0)
3. Fill up the mapping list starting with offset 1
4. Mapping->Offset 0: Set the correct number of mapped data objects
5. AutoSend->Block: Set the number of measurements that should make up one output data block
6. AutoSend->Config: Enable AutoSend feature

3.1.4.2.11 Commands | Triggering of Actions

Starting Address	***Mask***	***Description***
0xBF00	-	Automatic Teach Command will be asynchronously triggered on any write request
0xBF01	-	Manual Teach Command will be synchronously executed on any write request

0xBF10	-	Restart Device will be asynchronously triggered on any write request
0xBF11	-	Save Settings Command will be asynchronously triggered on any write request

3.1.4.2.12 Command Status | Results of Actions

Starting Address	***Mask***	***Description***
0xBF80	Low Byte	Teach Status 0: Idle 1: Busy -1: Failed
	High Byte-	Teach Error Code 0: OK 1…255: Error code

0xBF91	Low Byte	Save Settings Status 0: Idle 1: Busy -1: Failed
	High Byte-	Save Settings Error Code 0: OK 1…255: Error code

3.1.4.2.13 Triggering of Actions

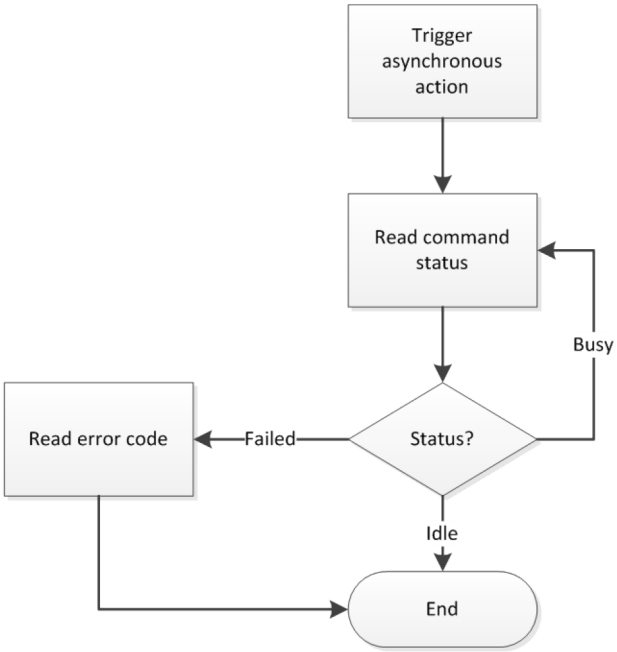

Fig.: 3-1 Trigger asynchronous action state machine

3.1.5 Deviations from the Modbus specification

In difference to the Modbus Specification the RapidoScan works with an object view. In a Modbus frame an access is only allowed in the address space of the object. Reading/writing on multiple objects by a single request is not supported!

Therefore, for example, the reading of (1) the Ethernet configuration and (2) the reading of the Modbus TCP configuration needs two separate Modbus frames!

4 Testing the Modbus TCP Functionality

4.1 Connecting to the integrated Web-Server

First connect to the Web-Server of the RapidoScan using your standard browser. If you haven't changed the settings of the RapidoScan then its IP address will be ***192.168.0.223*** when using ***LAN***.

If your RapidoScan provides WLAN too then you can use the built in Soft-AP functionality of the RapidoScan. Here the standard IP address is ***192.168.1.1*** and the network key of the ***WLAN*** is ***87654321***.

Log in as ***service*** user. The default password is "***service***":

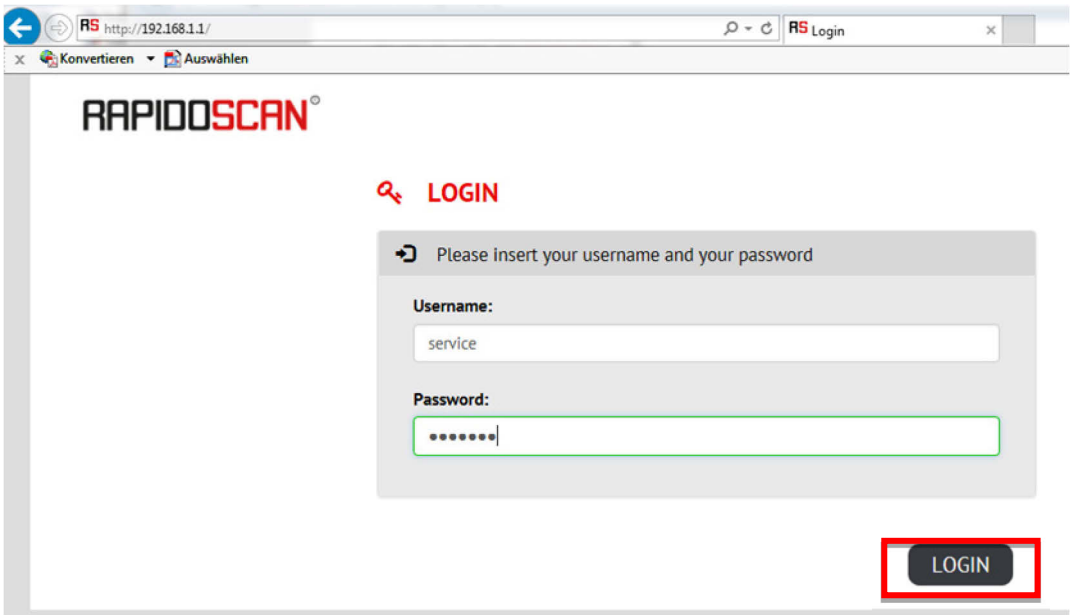

Fig.: 4-1

Next press ***LOGIN*** ***(***Fig.: 4-1***)***.

4.2 Configuration of the LAN Ethernet Setting

First navigate to the ***Connectivity*** page (Fig.: 4-2***)*** for configuration.

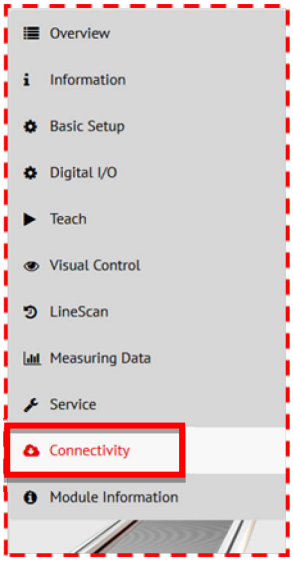

Fig.: 4-2

Open the ***LAN Ethernet*** folder to setup the general network settings.

The pre-set standard value is shown below:

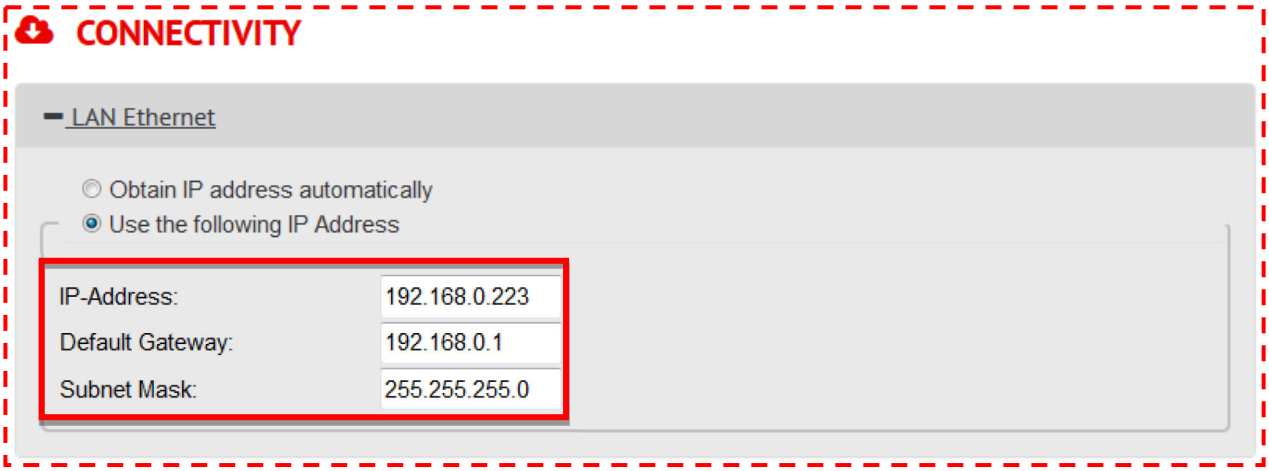

Fig.: 4-3

4.3 Configuration of Modbus TCP Setting

Open the ***ModbusTCP Settings*** folder to setup the ***TCP Port***.

Fig.: 4-4

The standard ***TCP port*** (Fig.: 4-4) for Modbus TCP is: ***502***
Using these settings, the RapidoScan will do the Modbus Communication at: 192.168.0.223:502

4.4 Doing a Test-Run with QModMaster

The tool is available at https://sourceforge.net/projects/qmodmaster/

At first we have to configure the connection settings "Options" -> "Modbus TCP …"

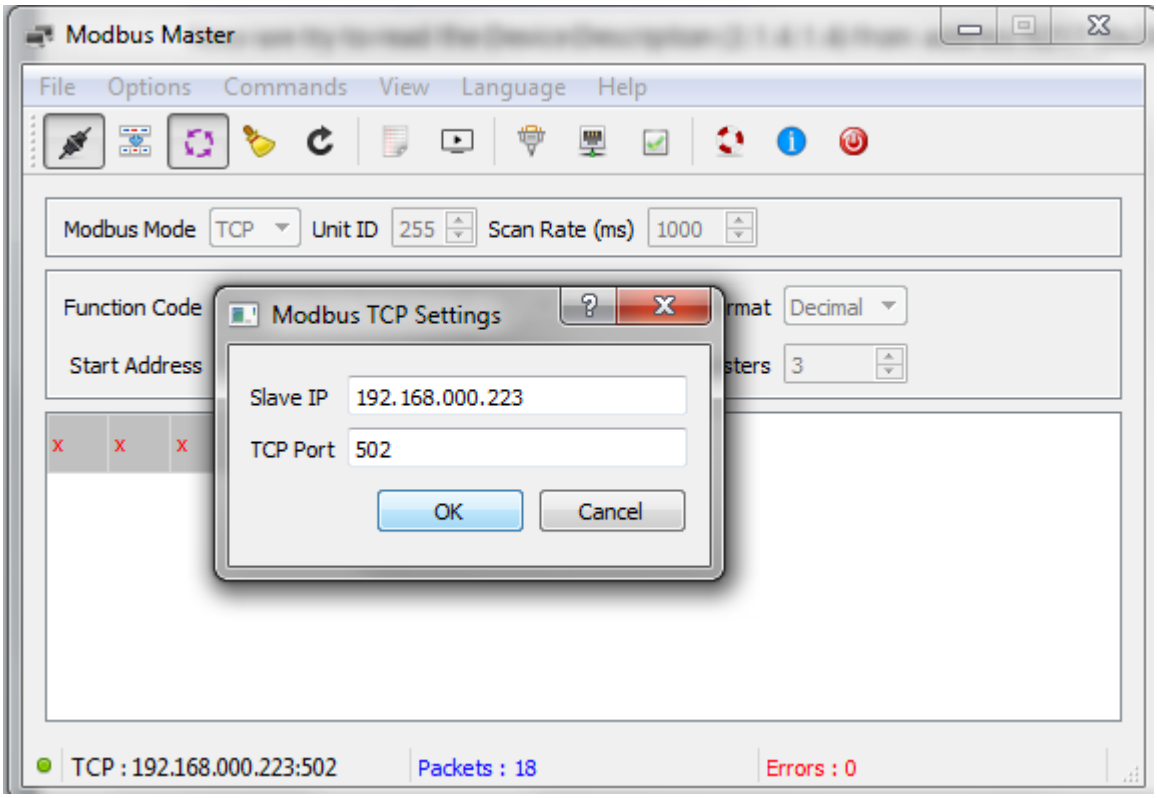

Fig.: 4-5

Now we try to read the Device Description (3.1.4.1.4) from Modbus address 9217 (0x2401)

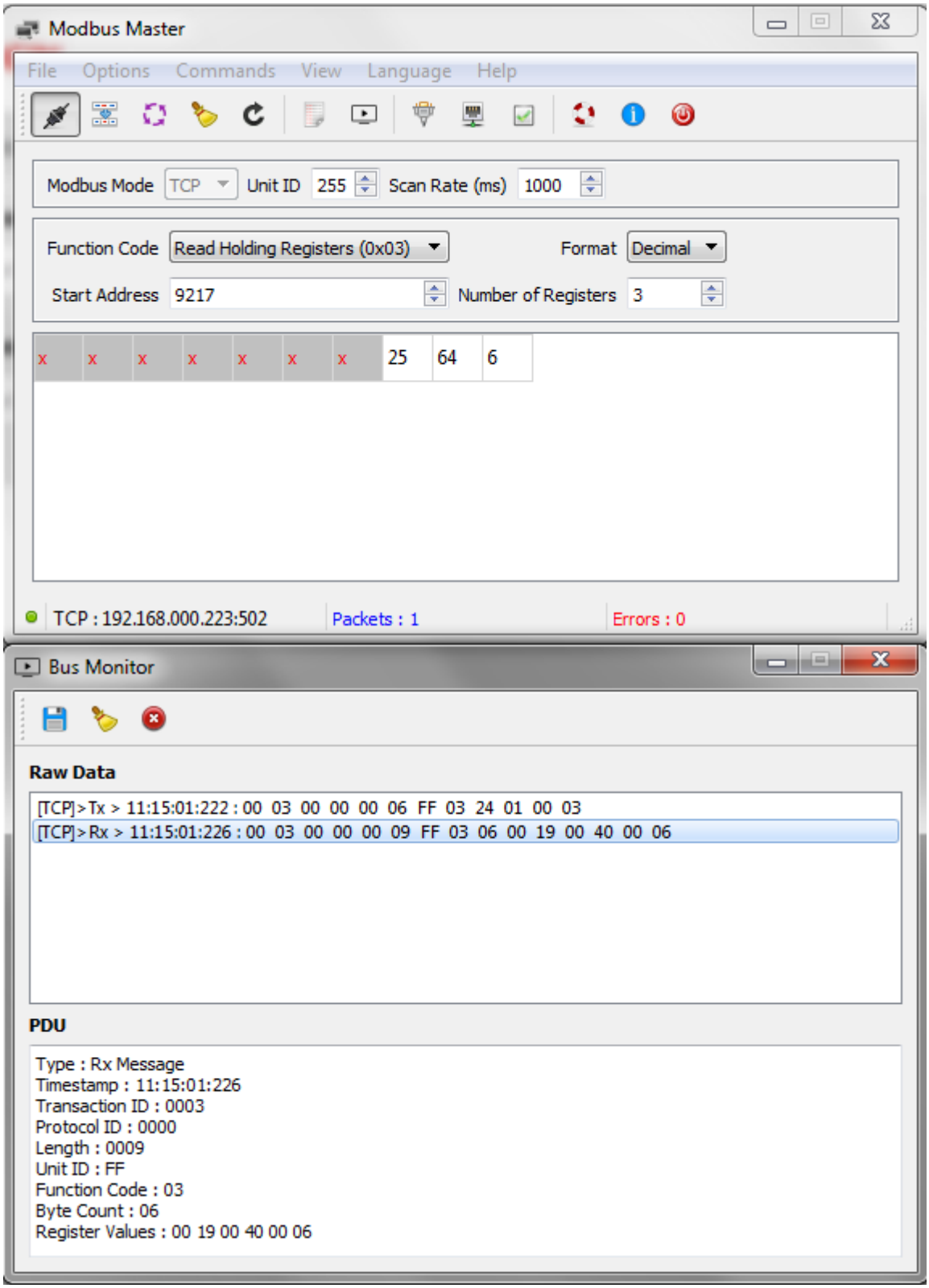

Fig.: 4-6

Beam Distance: 25mm

Diode Count: 64

Beam Count: 6

4.5 Connect to SIMATIC S7-1200/S7-1500

You use the "MB_CLIENT" instruction to establish a connection between the PLC (Modbus TCP client) and the RapidoScan (Modbus TCP server) over the PROFINET connection. To use the instruction, you do not require any additional hardware module or license.

4.5.1 Register Addresses

Siemens uses a simple data structure with 4 separated memory areas. The function code derives from the parameters MB_MODE, MB_DATA_ADDR and MB_DATA_LEN. So normally you have to add an address offset to the register addresses which are described in this document.

Further Information:

Detailed information about the addressing of the MB_CLIENT instruction is available in the manual "SIMATIC STEP 7 Professional …", in the section entitled "Parameters MB_MODE, MB_DATA_ADDR and MB_DATA_LEN (S7-1200, S7-1500)".

5 Testing the AutoSend Functionality

5.1 Connecting to the integrated Web-Server

First connect to the Web-Server of the RapidoScan using your standard browser. If you haven't changed the settings of the RapidoScan then its IP address will be ***192.168.0.223*** when using ***LAN***.

If your RapidoScan provides WLAN too then you can use the built in Soft-AP functionality of the RapidoScan. Here the standard IP address is ***192.168.1.1*** and the network key of the ***WLAN*** is ***87654321***.

Log in as ***service*** user. The default password is "***service***":

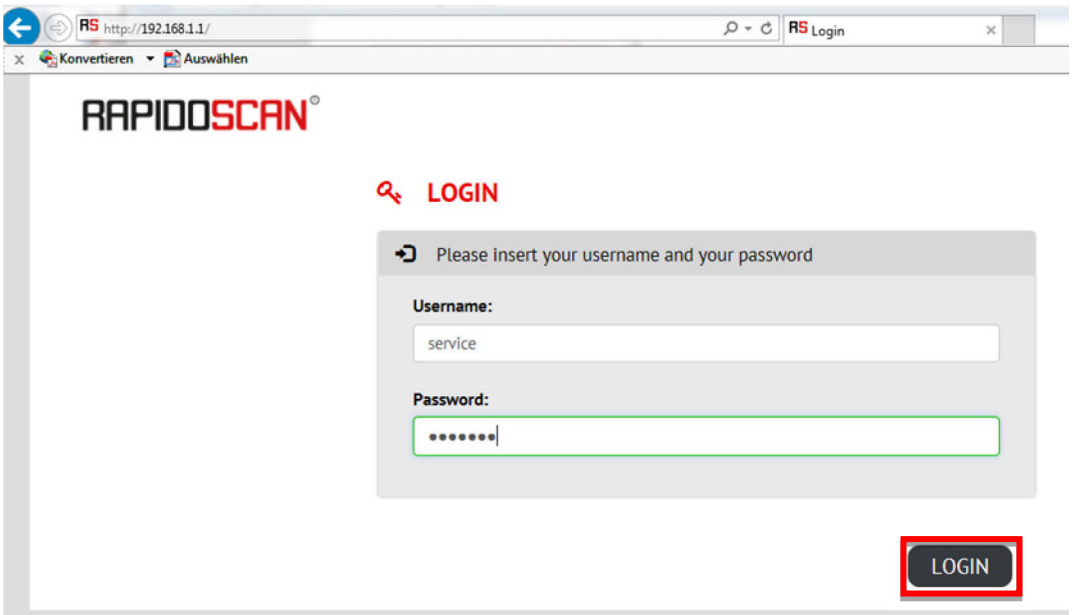

Fig.: 5-1

Next press ***LOGIN*** (Fig.: 5-1).

5.2 Configuration of the LAN Ethernet Setting

First navigate to the ***Connectivity*** page (Fig.: 5-2) for configuration.

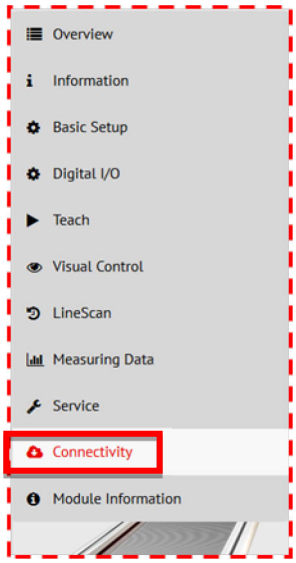

Fig.: 5-2

Open the ***LAN Ethernet*** folder to setup the general network settings.

The pre-set standard value is shown below:

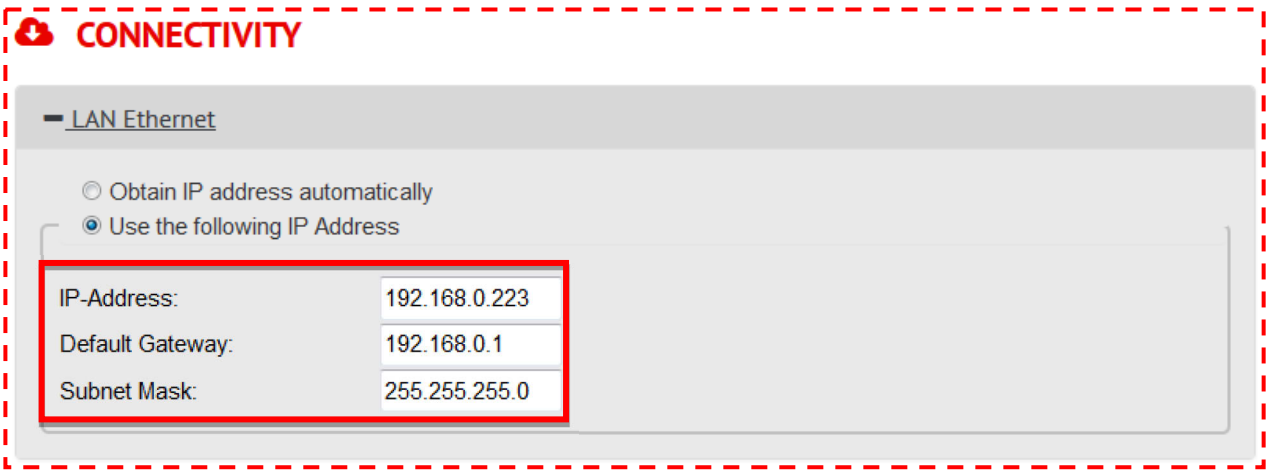

Fig.: 5-3

5.3 Configuration of the AutoSend Channel

At first open the ***AutoSend Channel*** folder (Fig.: 5-4).

Fig.: 5-4

Now enable ***AutoSend*** and choose the ***Client/Server*** & ***Network Protocol*** adjustment for your application (Fig.: 5-5).

Fig.: 5-5

UDP | Client setup

In case you are using the UDP protocol the RapidoScan typically acts as a Client seen from the connection buildup. So the RapidoScan must be setup to what address it has to buildup connection and which port the remote station is using.

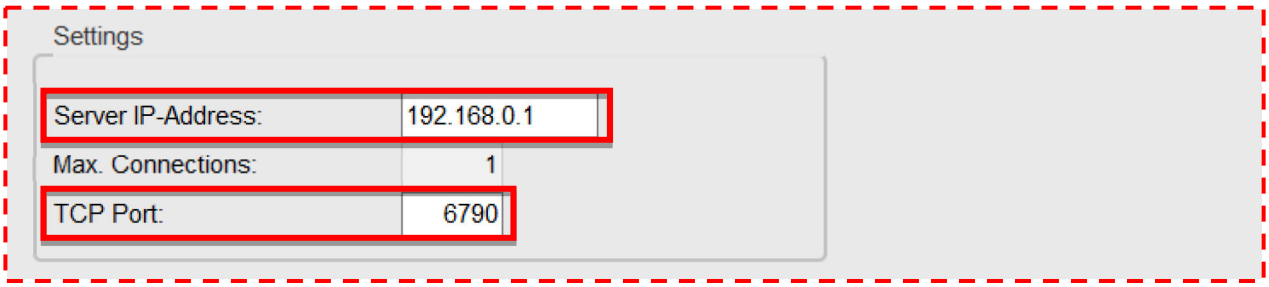

Fig.: 5-6

TCP | Server setup

In case you are using the TCP protocol the RapidoScan typically acts as a Server seen from the connection buildup. Here the RapidoScan awaits the connection buildup through a client. The number of parallel allowed connections is actually limited to 1.

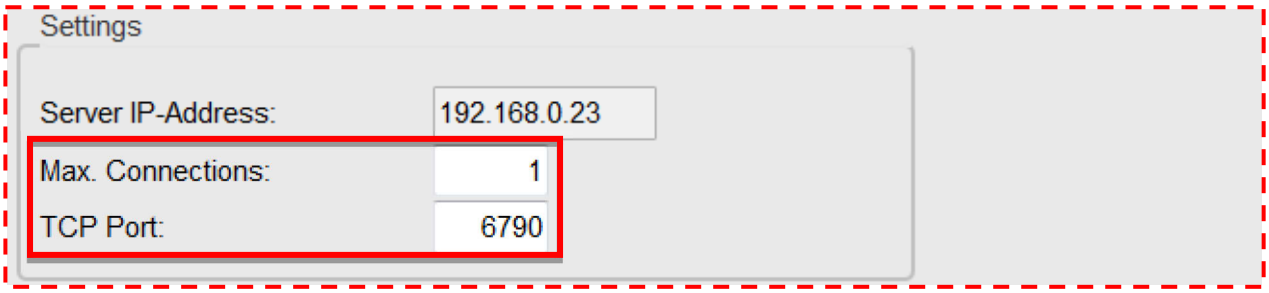

Fig.: 5-7

This test setup uses following settings:

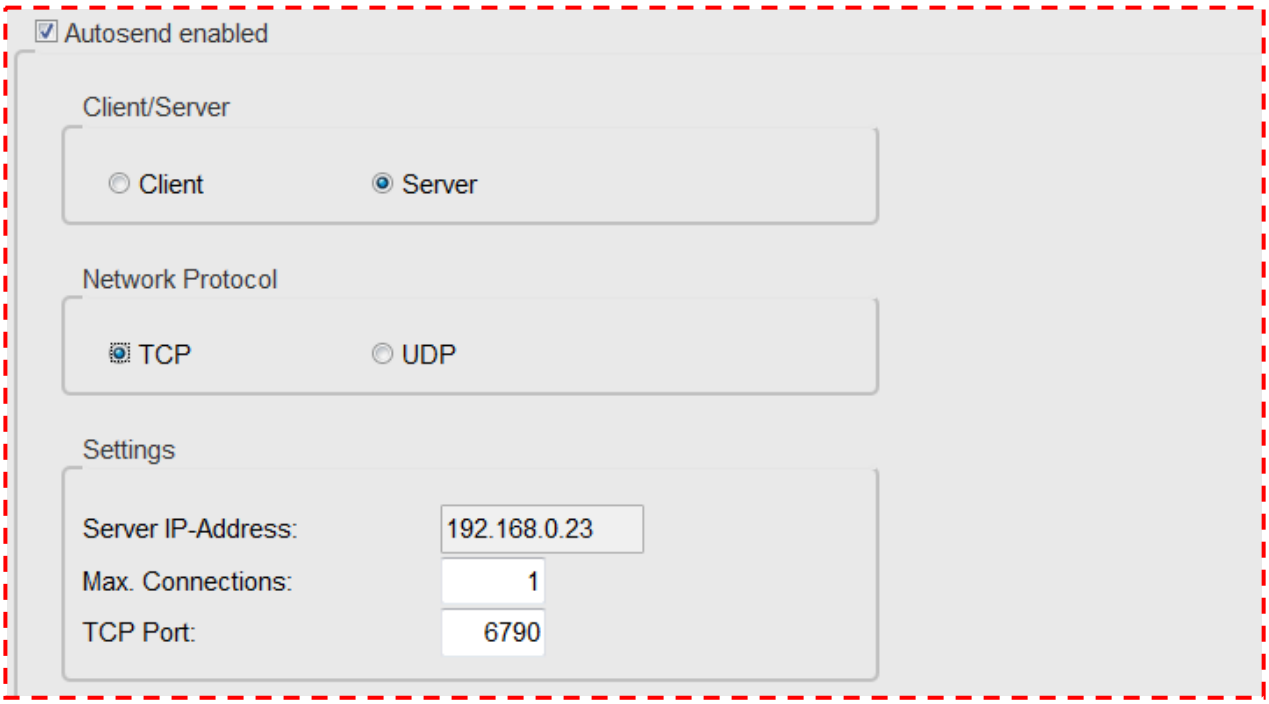

Fig.: 5-8

For checking purposes it will be best working with a simple setup: CYCLE_COUNTER.

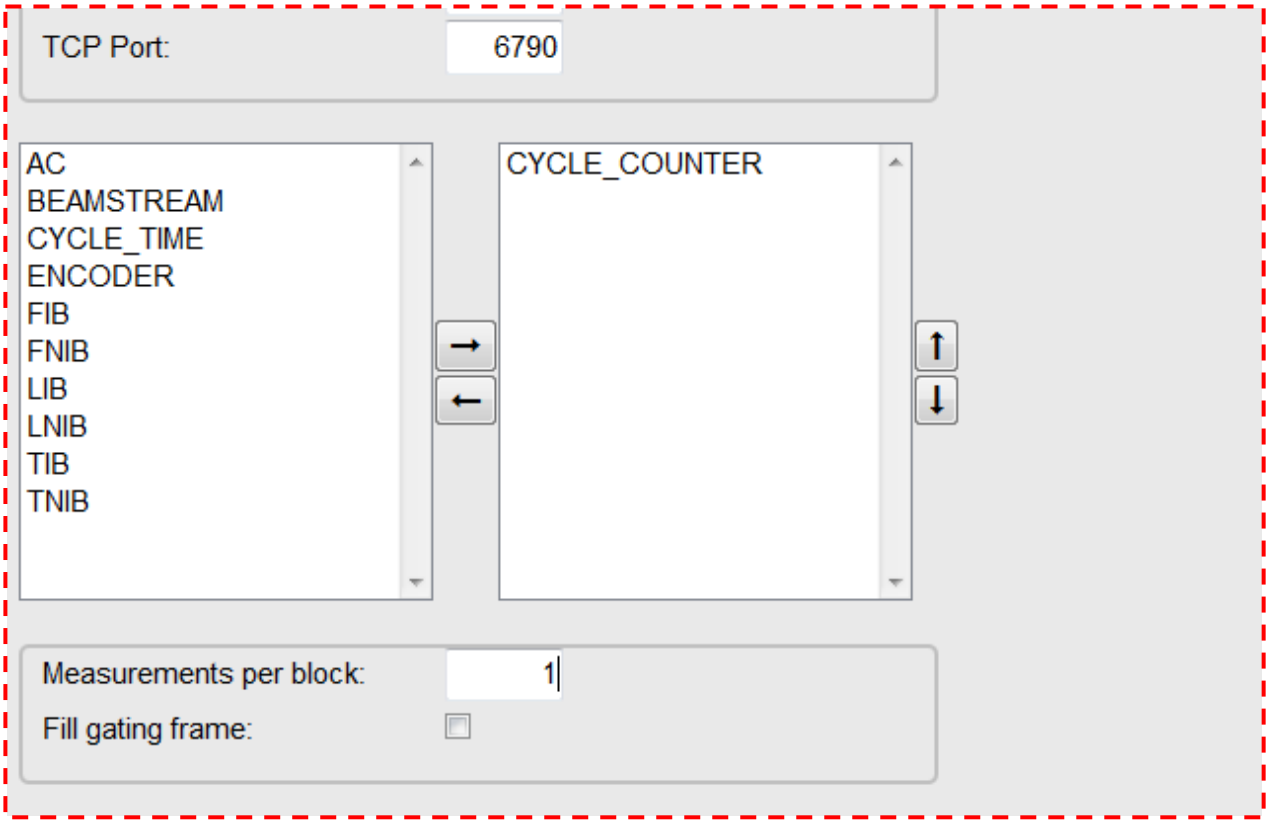

Fig.: 5-9

Very important: Do not forget to press the ***Write to Device*** button for transferring the configuration to the scanner. For permanent storing/making persistent the setup, press the ***SAVE*** button at the upper left corner

5.4 Doing a Test-Run with Docklight Scripting

Now you can setup the remote side.

As example we are using "Docklight Scripting" which is available at https://docklight.de

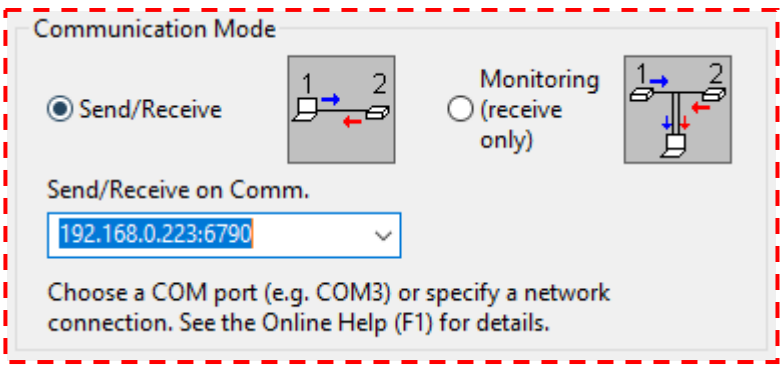

Fig.: 5-10

Once you have setup the correct address, you should be able to receive data from the RapidoScan.

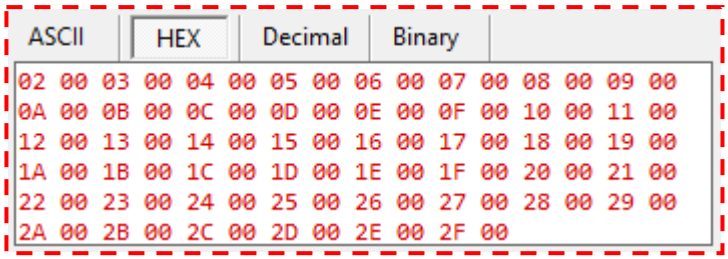

Fig.: 5-11

The cycle counter has to be interpreted as 0x0002, 0x0003, 0x0004, …

5.5 Doing a Test-Run with Wireshark

This test setup maps different objects and add up multiple measurements to a bigger UDP package

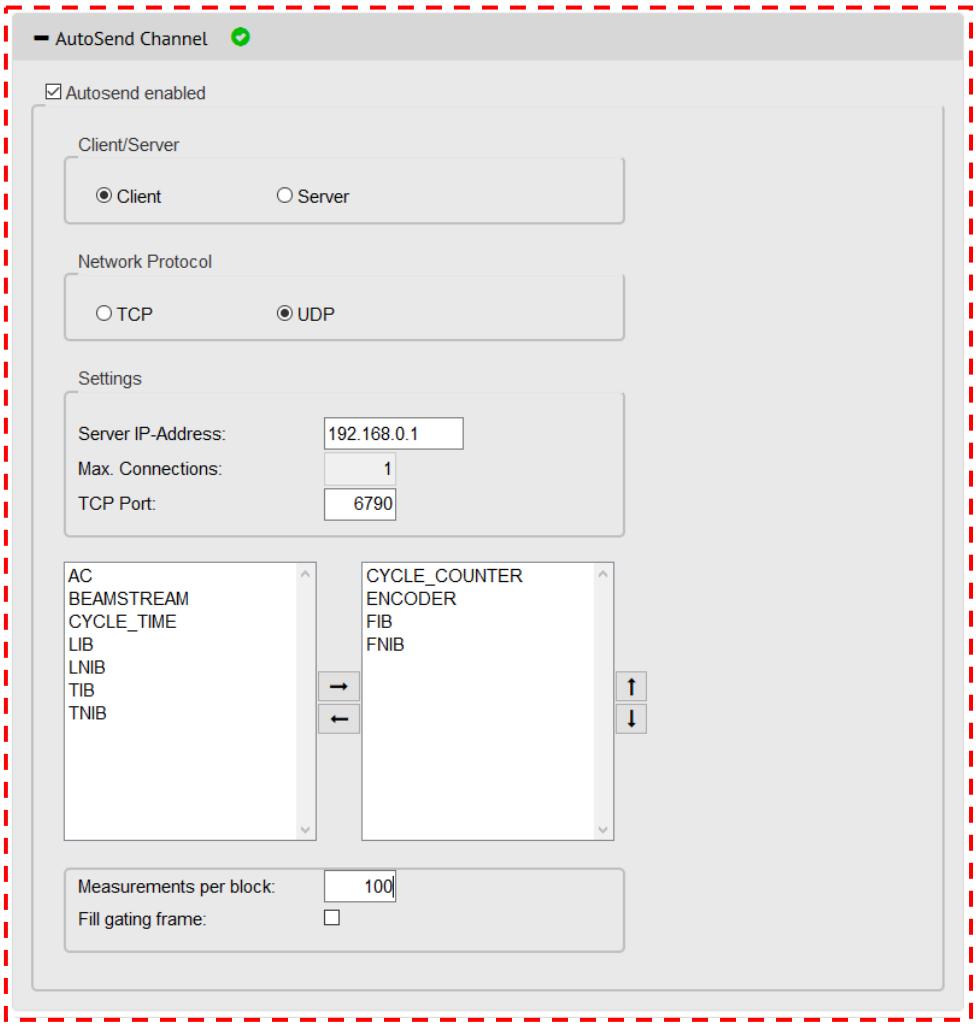

Fig.: 5-12

Captured packets by Wireshark:

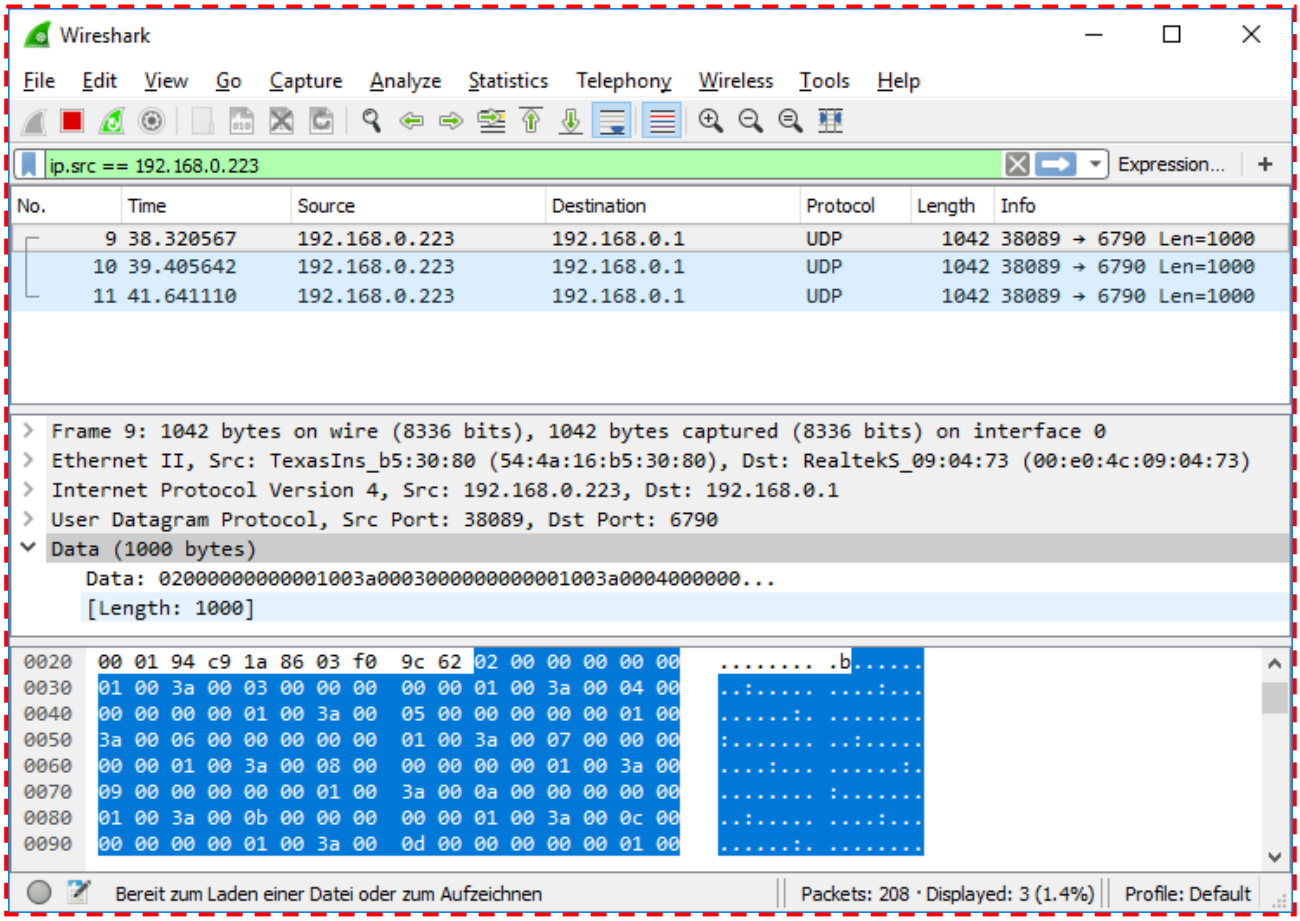

Fig.: 5-13

During the test only beam 58 (hex 3A) was free!

Broken down to single measurements (2 Byte CYCLE_COUNER, 4 Byte Encoder, 2 Byte FIB, 2 Byte FNIB):

02 00 | 00 00 00 00 | 01 00 | 3A 00

03 00 | 00 00 00 00 | 01 00 | 3A 00

04 00 | 00 00 00 00 | 01 00 | 3A 00