Advanced Diploma of Industrial Data Communication, Networking and IT 52782WA (DIT)

Module 02 Practical Assignment Lab Instructions

Industrial Data Communications

V2

Question 3: Modbus Basics

Created By:	EIT	Date:	
Reviewed By:	MADDOX	Date:	15 MAR 2016
Reviewed By:	John Lawrence	Date:	04 June 2020

ENGINEERING INSTITUTE OF TECHNOLOGY

QUESTION 3 MODBUS BASICS

(15 marks)

Getting Started

- Logon to Electromeet (Follow the How to Connect to RemoteLabs_Electromeet_HTML5_Remote_Lab instructions document)
- The software is installed on **Remote Lab 1 & 2**

Hardware:

- Normally we would run the MODBUS simulation software (client and server a.k.a. master and slave) on two separate computers, via a null modem cable. However, in this case we are running them both on one machine, via a null modem simulator.

Software used:

- Modbus Poll v3.60
- Modbus Slave v3.10
- TAL Virtual Null Modem(taltech.com)
- All the above is installed on the Remote Lab computer

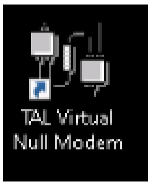
Modbus

- You can complete this practical assignment by logging into the Electromeet Remote Lab computer as per above.
- You do not need any specialised hardware of your own, as was the case in the past.

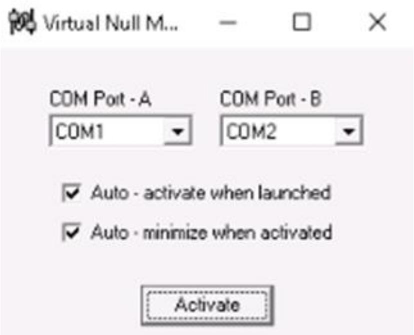
We will be using COM1 and COM2

- Other pairs may be set (COM5 and COM6 say) and the Software activated ... there is no need to change these numbers!
- Windows below (in Modbus Poll and Slave) show PORTS 7 and 8 – I prefer not to use these last 2 port numbers.

Open TAL Virtual Null Modem by clicking on the icon (on desktop or taskbar)



1. Use the default settings for COM1 on COM Port - A and COM2 on COM Port - B
   a. Then Tick the two Boxes:
   b. “Auto-activate when launched” and “Auto-minimize when activated”



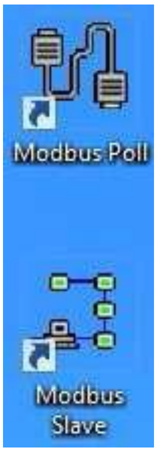
COM Port - A: COM1
COM Port - B: COM2

☑ Auto - activate when launched

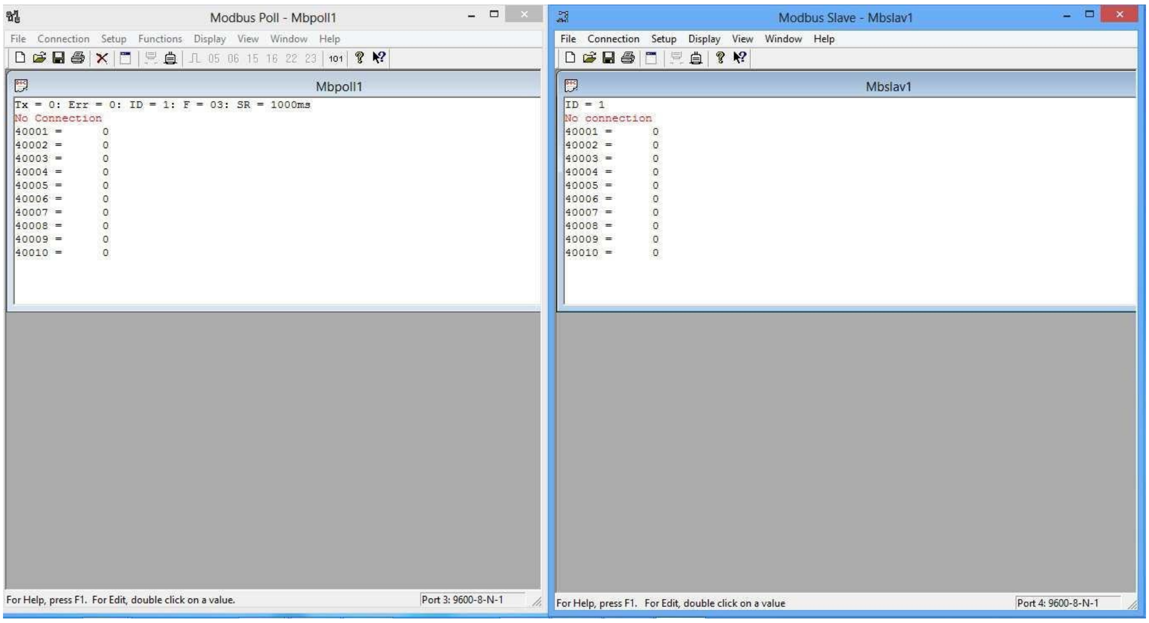
☑ Auto - minimize when activated

2. Now click the “Activate” button and the window will be minimized to taskbar. It may still be minimized in Task Bar from previous Users so check there please.

3. Now run both the MODBUS Master (MBPoll) and the MODBUS Slave (MBSlave) by clicking on the desktop icons.



The easiest approach is to run them side-by-side, adjusting them to fit in next to each other like this


Modbus Poll - Mbpoll1
File Connection Setup Functions Display View Window Help
05 06 15 16 22 23 101
Modbus Slave - Mbslav1
File Connection Setup Display View Window Help

Mbpoll1

```
Tx = 0: Err = 0: ID = 1: F = 03: SR = 1000ms
No Connection
40001 =       0
40002 =       0
40003 =       0
40004 =       0
40005 =       0
40006 =       0
40007 =       0
40008 =       0
40009 =       0
40010 =       0
```

For Help, press F1. For Edit, double click on a value.   Port 3: 9600-8-N-1

Mbslav1

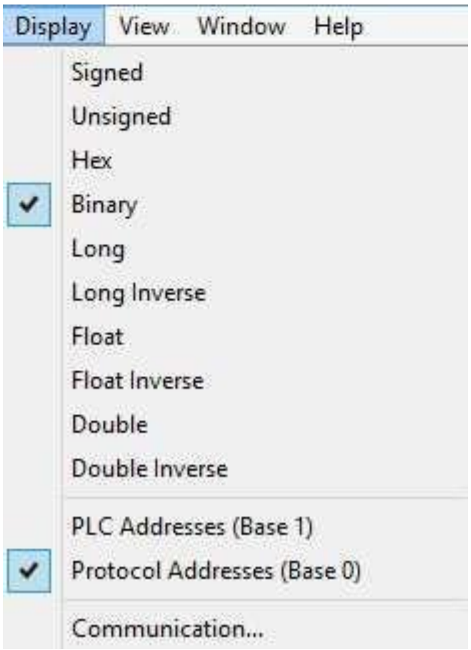
```
ID = 1
No connection
40001 =       0
40002 =       0
40003 =       0
40004 =       0
40005 =       0
40006 =       0
40007 =       0
40008 =       0
40009 =       0
40010 =       0
```

For Help, press F1. For Edit, double click on a value   Port 4: 9600-8-N-1

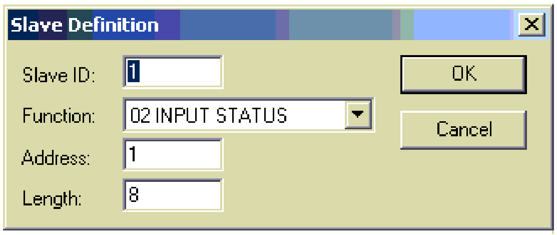
Let’s start with the Slave.

First, we are going to configure the way in which the information is displayed, viz. (a) binary (all 1’s and 0’s) and (b) Base 0 i.e. protocol address notation, starting from 0 (as opposed to PLC addresses, starting from 1).

1. Click Display and select Binary, then click Display again and select Protocol Addresses (Base 0). You will end up with something like this:



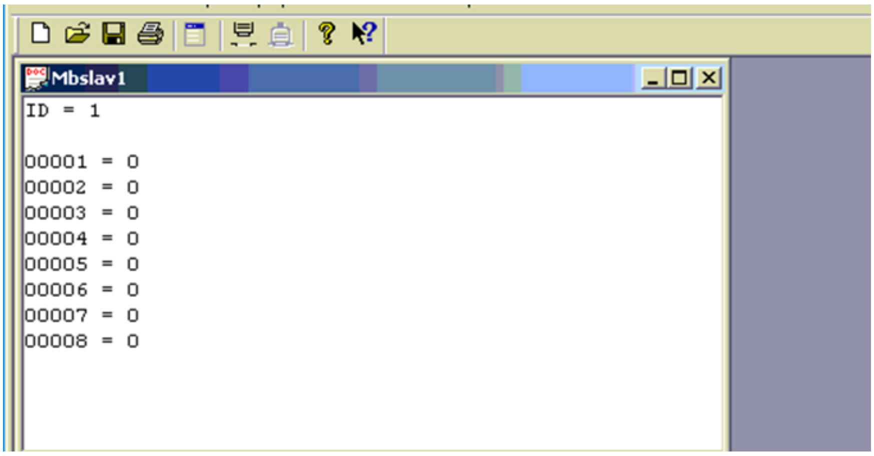
2. Now hit F2 or click Setup-> Slave definition. Set the slave up as follows.



Slave ID: 1
Function: 02 INPUT STATUS
Address: 1
Length: 8

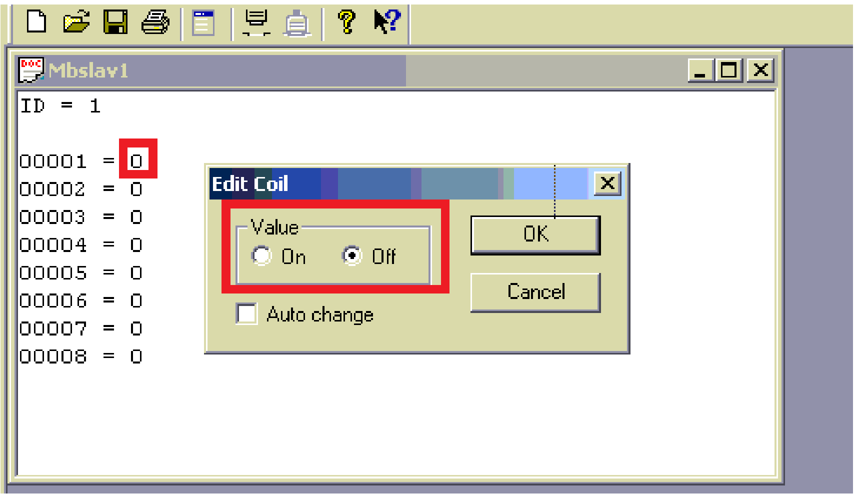
The slave address in this case is 1. Function Code (FC) = 02 (Input status). Address = 1 refers to the logical address of the first coil, and length = 8 means that there are 8 consecutive coils (numbered 1 thru 8). This represents an 8 bit number in our case.

3. Check / Replace the address with the value 1 and the Length with the value 8 These will now show up as follows



```
ID = 1

00001 = 0
00002 = 0
00003 = 0
00004 = 0
00005 = 0
00006 = 0
00007 = 0
00008 = 0
```

To edit any coil, just double-click on it and toggle the radio buttons on Edit Coil between on and off. You can do this once the simulation is running.

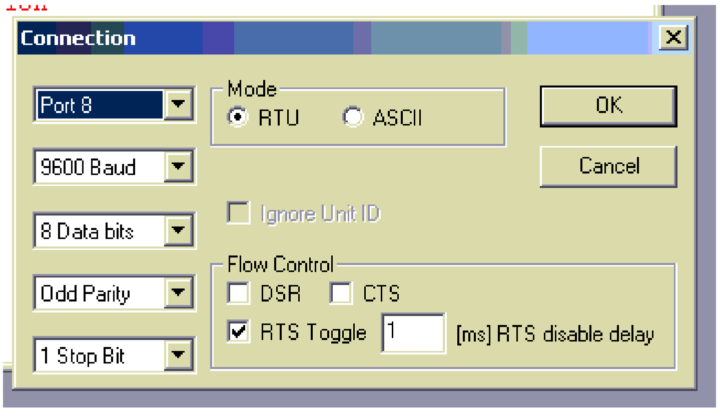


```
ID = 1

00001 = 0
00002 = 0
00003 = 0
00004 = 0
00005 = 0
00006 = 0
00007 = 0
00008 = 0
```

Value ○ On ● Off

□ Auto change

4. Click *Connection->connect* and set up the serial communications parameters as shown. Ensure that RTU mode is selected for now. Then click OK. Alternatively you might want to refrain from connecting until the Master is also ready.



Port 8

9600 Baud

8 Data bits

Odd Parity

1 Stop Bit

Mode ● RTU ○ ASCII

□ Ignore Unit ID

Flow Control

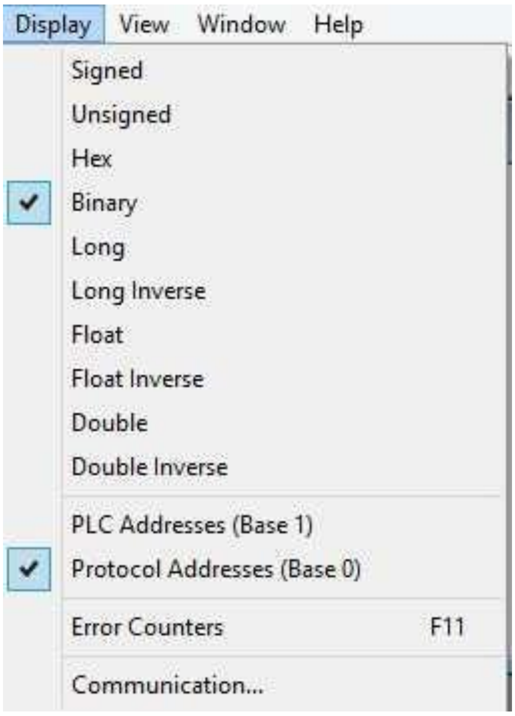
□ DSR □ CTS

☑ RTS Toggle 1 [ms] RTS disable delay

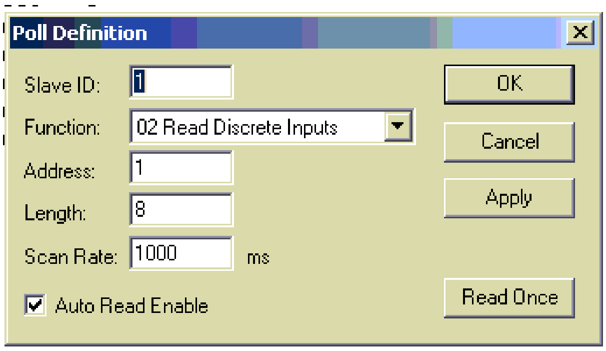
Check the port settings carefully, as the Master side settings have to match. The configuration here is 9600,8,O,1.

Let us Focus on the Master side.

5. Click *Display* and set it up as follows (the same as for the Slave).



6. hit F2 or click Setup-> Poll definition.


Poll Definition

Slave ID: 1

Function: 02 Read Discrete Inputs

Address: 1

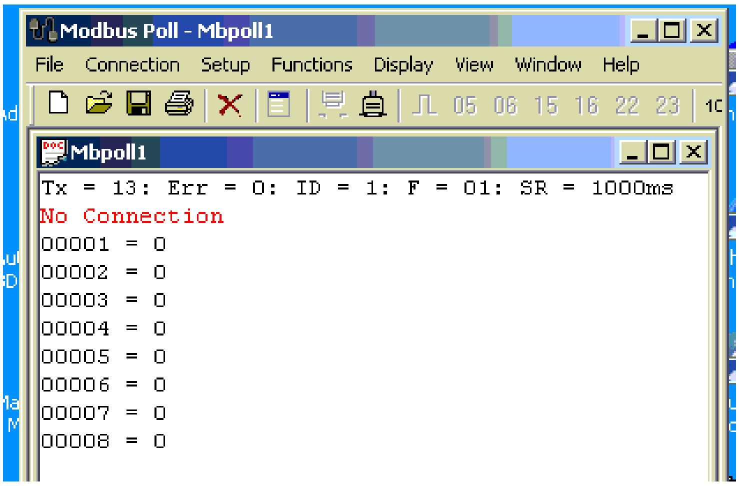
Length: 8

Scan Rate: 1000 ms

☑ Auto Read Enable

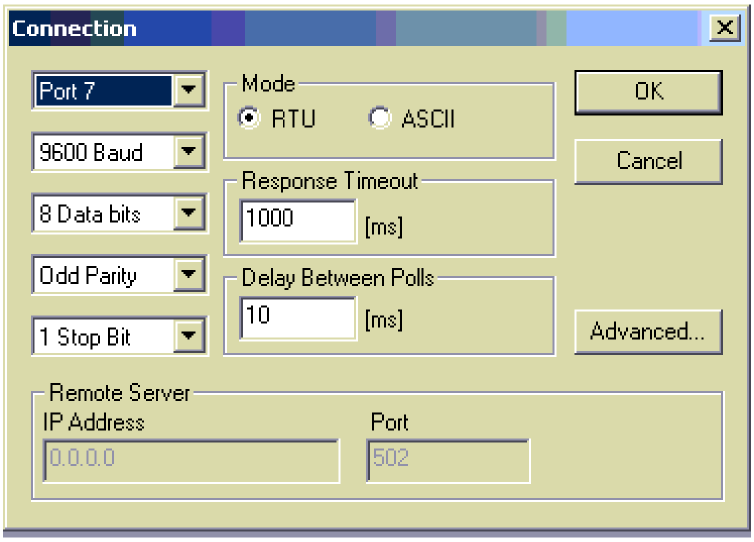
In the poll definition above, coils 1 to 8 (starting with 1, total =8) on slave 1 will be read once every second.

7. Make sure the correct function is selected. Click *OK*.



```
Tx = 13: Err = 0: ID = 1: F = 01: SR = 1000ms
No Connection
00001 = 0
00002 = 0
00003 = 0
00004 = 0
00005 = 0
00006 = 0
00007 = 0
00008 = 0
```

8. Hit F3 or click *Connection->connect*. Select COM7 and ensure that the settings are the same as for the Slave. Also ensure that RTU mode is selected for now.



Port 7
9600 Baud
8 Data bits
Odd Parity
1 Stop Bit

Mode: ● RTU ○ ASCII

Response Timeout: 1000 [ms]

Delay Between Polls: 10 [ms]

Remote Server
IP Address: 0.0.0.0
Port: 502

If all goes according to plan, connection will be established.

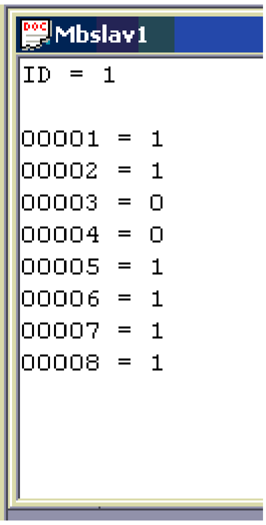
If a red ‘timeout’ message appears on either side, do the following.

- Click *Disconnect* on both sides
- Check that the communications parameters (baud, etc.) are the same for both sides
- Reconnect on both sides

If a red ‘illegal’ message appears on either side, do the following.

- Click *Disconnect* on both sides
- Check if the poll definitions match (slave addresses and function codes, modes)
- Check that the inputs read by the Poll program are a SUBSET of the coils defined by the Slave program, and not the other way around
- Reconnect on both sides

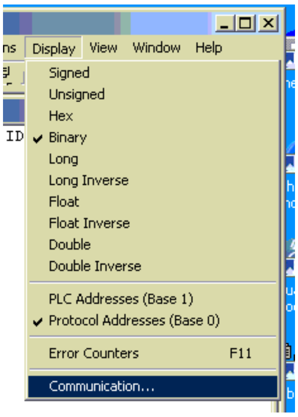
9. 14.Now setup you slave input values to represent $1111\ 0011_2$ = $F3_{16}$. (You will setup the equivalent binary input as below)

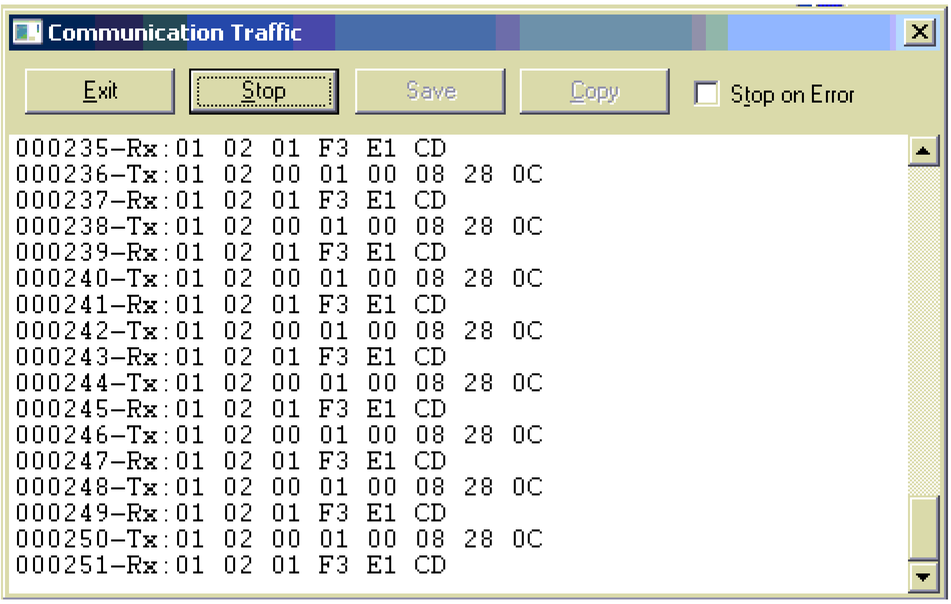


Mbslav1

```
ID = 1

00001 = 1
00002 = 1
00003 = 0
00004 = 0
00005 = 1
00006 = 1
00007 = 1
00008 = 1
```

10. 15. Click *Display->communications* and observe the traffic between master and slave. Remember that it is as seen from the master's perspective, the display on the slave will be the other way around i.e. Tx on the master will be Rx on the slave.





```
000235-Rx:01 02 01 F3 E1 CD
000236-Tx:01 02 00 01 00 08 28 0C
000237-Rx:01 02 01 F3 E1 CD
000238-Tx:01 02 00 01 00 08 28 0C
000239-Rx:01 02 01 F3 E1 CD
000240-Tx:01 02 00 01 00 08 28 0C
000241-Rx:01 02 01 F3 E1 CD
000242-Tx:01 02 00 01 00 08 28 0C
000243-Rx:01 02 01 F3 E1 CD
000244-Tx:01 02 00 01 00 08 28 0C
000245-Rx:01 02 01 F3 E1 CD
000246-Tx:01 02 00 01 00 08 28 0C
000247-Rx:01 02 01 F3 E1 CD
000248-Tx:01 02 00 01 00 08 28 0C
000249-Rx:01 02 01 F3 E1 CD
000250-Tx:01 02 00 01 00 08 28 0C
000251-Rx:01 02 01 F3 E1 CD
```

Tx refers to the Modbus request, because we are looking at the Master here. 0x means Hex.

- Slave = 0x01 (i.e. 1 decimal)
- Function code = 0x02 (2 = Read input status)
- Initial coil address = 0x0001 (i.e. decimal 1 protocol)
- Number of coils = 0x0008
- CRC = 0x280C

Rx refers to the Modbus Rx response:

- Slave = 0x01
- Function = 0x02
- Byte count = 0x01
- Coil status = 0xF3 = 11110011
- CRC = 0xE1CD

11. Without much further assistance, retry all the steps but reconfigure both applications to ASCII mode, to read an 8-bit Digital Input (Function 2) register that reads the value F3$_{16}$.

Question 3 (a): Capture and paste the ASCII communications (command & response) frames.

12. Without much further assistance, retry all the steps but reconfigure both applications to RTU mode again, to read an 8-bit Digital Input (Function register that reads the value F3$_{16}$.

Question 3 (b): Capture and paste the RTU communications (command & response) frames.

Question 3 (c): Discuss differences and similarities seen between **ASCII** & **RTU** communications frames as captured in previous two questions.

***Tip**: to clearly see the differences/similarities, it is advantageous to map the two encoding systems against each other in tabular format.*

Question 3 (d): If compared to the RTU frame as described in the Modbus RTU standard, does the captured data conform to the standard RTU frame – Yes or No?
Clearly explain your answer in one sentence.

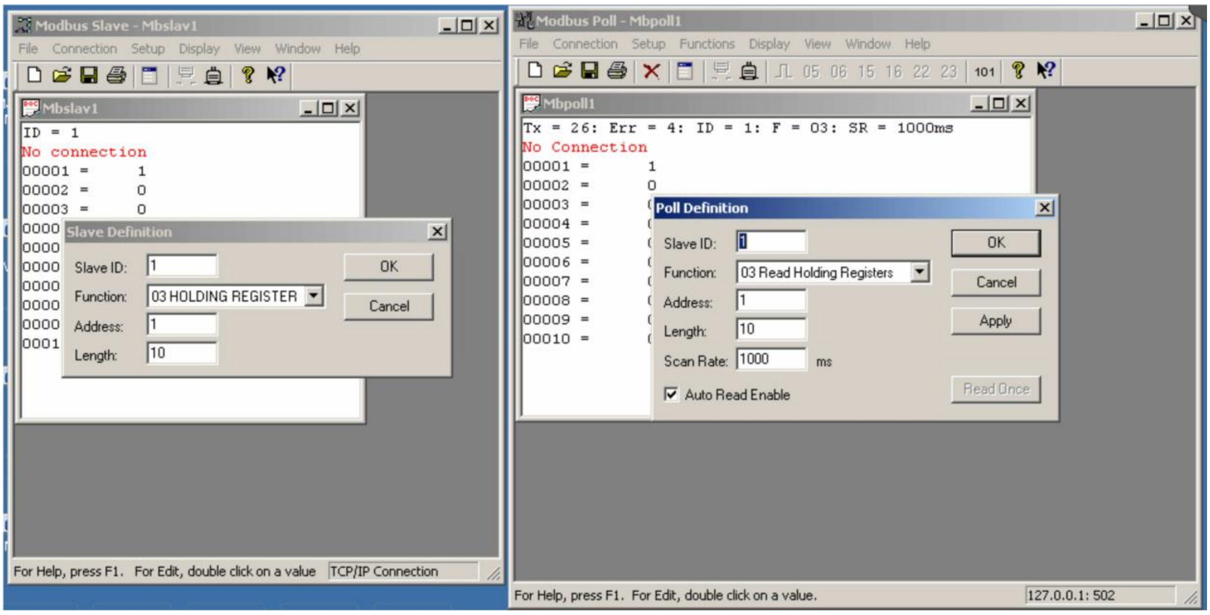
Question 3 (e): If compared to the ASCII frame as described in the Modbus ASCII standard, does the captured data conform to the standard ASCII frame – Yes or No?
Clearly explain your answer in one sentence.

End of Practical Question 3.

Modbus TCP (OPTIONAL, for the adventurous)

Although not required for this Lab, if you do wish to play with MODBUS over TCP/IP, you can do a loopback by writing and reading from **IP address 127.0.0.1 Port 502** (which is a PC's default local loopback IP address) with both MODPOLL and MODSLAVE.

The purpose of this document is to introduce you to the Modbus/TCP concept. You may, however, use other registers, etc. Also take care to use the IP address in bold (above) and not the one in the screenshots below.



Modbus Slave - Mbslav1
File Connection Setup Display View Window Help
Modbus Poll - Mbpoll1
File Connection Setup Functions Display View Window Help
05 06 15 16 22 23 101

Mbslav1

```
ID = 1
No connection
00001 =      1
00002 =      0
00003 =      0
```

Slave Definition

Slave ID: 1
Function: 03 HOLDING REGISTER
Address: 1
Length: 10
OK
Cancel

For Help, press F1. For Edit, double click on a value TCP/IP Connection

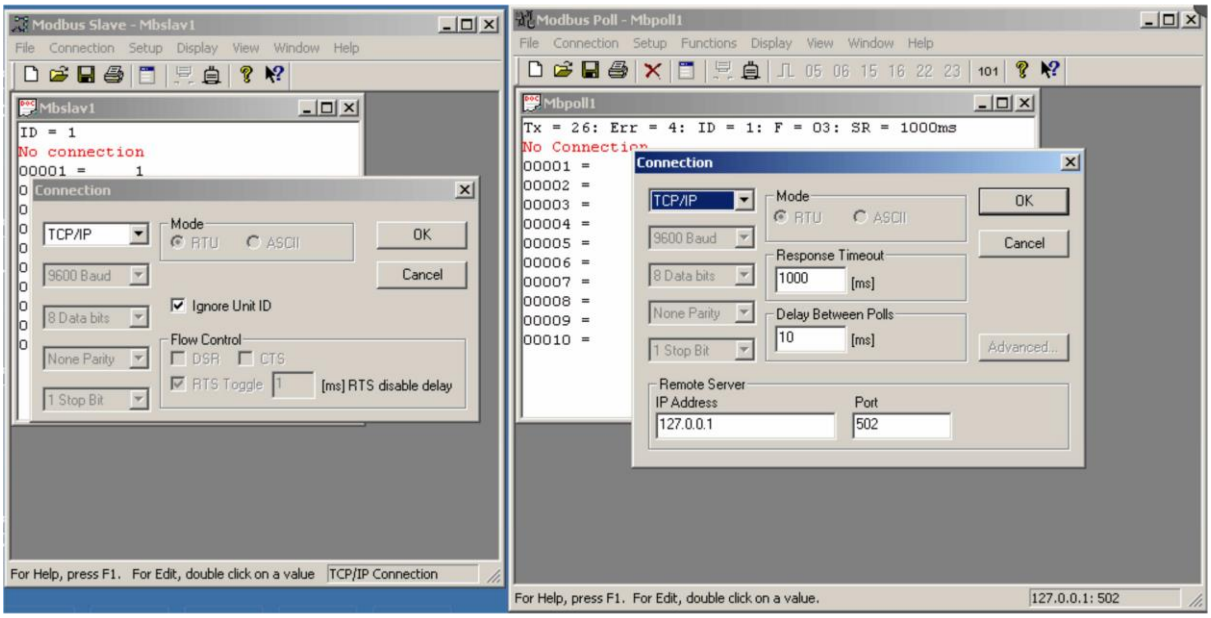
Mbpoll1

```
Tx = 26: Err = 4: ID = 1: F = 03: SR = 1000ms
No Connection
00001 =      1
00002 =      0
00003 =
00004 =
00005 =
00006 =
00007 =
00008 =
00009 =
00010 =
```

Poll Definition

Slave ID: 1
Function: 03 Read Holding Registers
Address: 1
Length: 10
Scan Rate: 1000 ms
☑ Auto Read Enable
OK
Cancel
Apply
Read Once

For Help, press F1. For Edit, double click on a value. 127.0.0.1: 502



Modbus Slave - Mbslav1
File Connection Setup Display View Window Help
Modbus Poll - Mbpoll1
File Connection Setup Functions Display View Window Help
05 06 15 16 22 23 101

Mbslav1

```
ID = 1
No connection
00001 =      1
```

Connection

TCP/IP
9600 Baud
8 Data bits
None Parity
1 Stop Bit
Mode ● RTU ○ ASCII
☑ Ignore Unit ID
Flow Control ☐ DSR ☐ CTS
☑ RTS Toggle 1 [ms] RTS disable delay
OK
Cancel

For Help, press F1. For Edit, double click on a value TCP/IP Connection

Mbpoll1

```
Tx = 26: Err = 4: ID = 1: F = 03: SR = 1000ms
No Connection
00001 =
00002 =
00003 =
00004 =
00005 =
00006 =
00007 =
00008 =
00009 =
00010 =
```

Connection

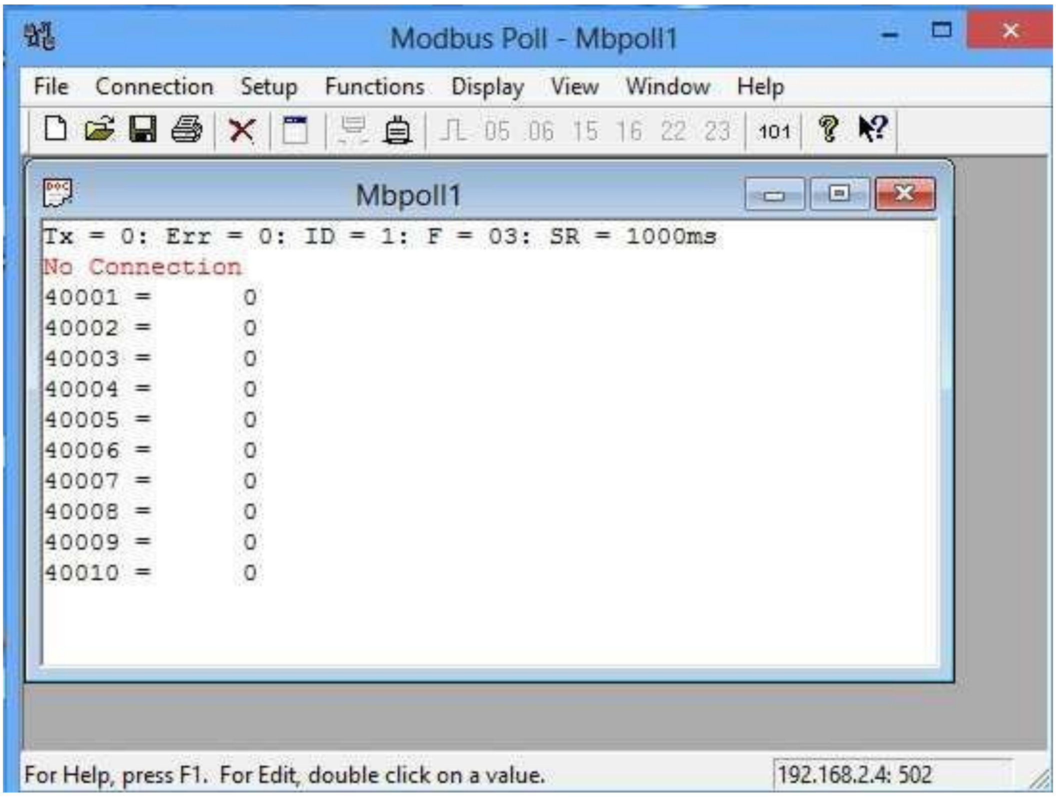
TCP/IP
9600 Baud
8 Data bits
None Parity
1 Stop Bit
Mode ● RTU ○ ASCII
Response Timeout 1000 [ms]
Delay Between Polls 10 [ms]
Remote Server
IP Address 127.0.0.1
Port 502
OK
Cancel
Advanced...

For Help, press F1. For Edit, double click on a value. 127.0.0.1: 502

The steps to get to the above results, are broken down below.

1. Run the Slave first and set it up EXACTLY as in the pictures above. Start the connection.
2. Now run the Modbus Master (MBPoll) by clicking on the desktop icons.

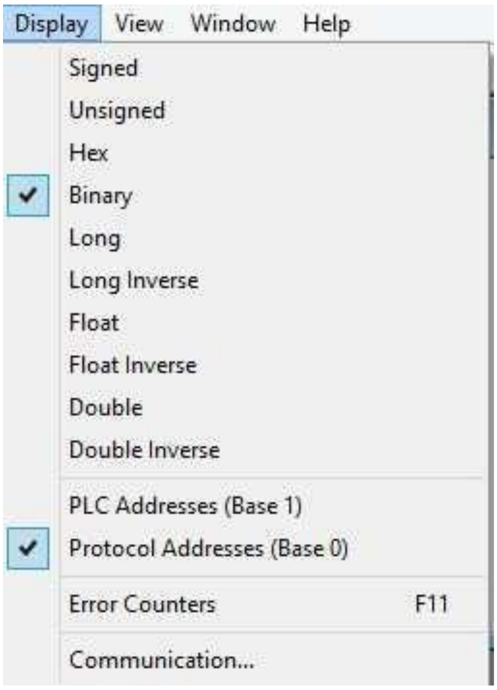
3. MBPoll opens.



```
Tx = 0: Err = 0: ID = 1: F = 03: SR = 1000ms
No Connection
40001 =       0
40002 =       0
40003 =       0
40004 =       0
40005 =       0
40006 =       0
40007 =       0
40008 =       0
40009 =       0
40010 =       0
```

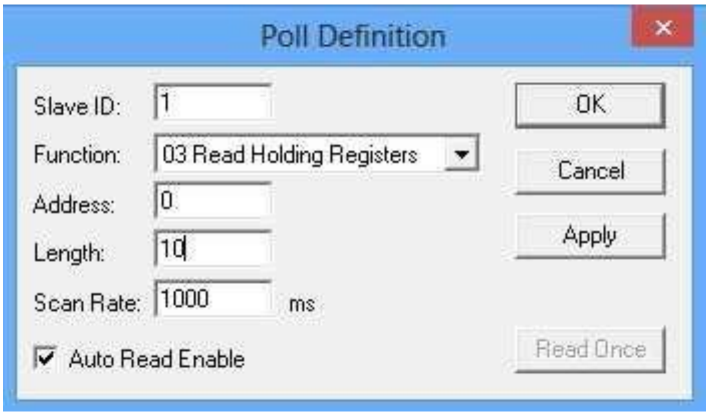
For Help, press F1. For Edit, double click on a value.   192.168.2.4: 502

Click *Display* and set it up as follows.



Signed
Unsigned
Hex
✓ Binary
Long
Long Inverse
Float
Float Inverse
Double
Double Inverse
PLC Addresses (Base 1)
✓ Protocol Addresses (Base 0)
Error Counters F11
Communication...

Hit F2 or click *Setup-> Poll definition.* This time we will use FC03.



Poll Definition

Slave ID: 1

Function: 03 Read Holding Registers

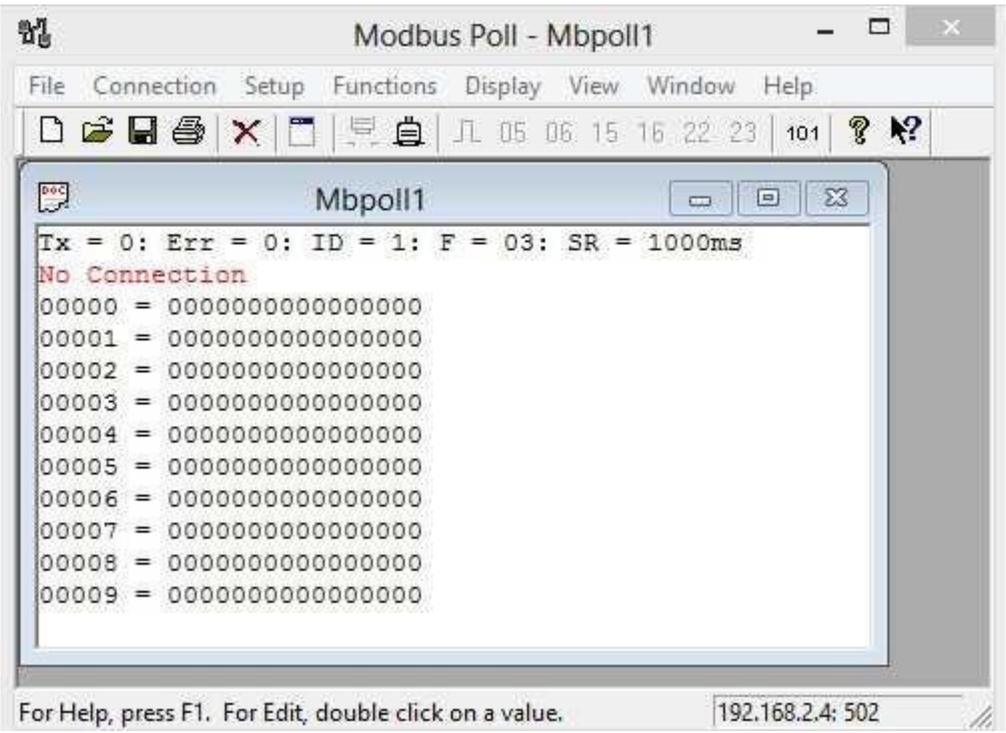
Address: 0

Length: 10

Scan Rate: 1000 ms

☑ Auto Read Enable

In the poll definition above, holding registers 0 thru 9 inclusive (protocol) i.e. 40001 thru 40010 will be read once every second. Click *OK*.



Modbus Poll - Mbpoll1

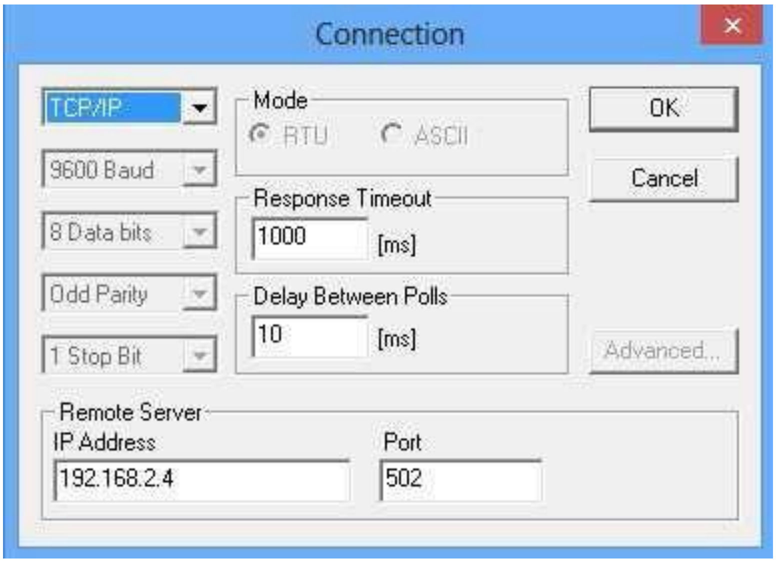
Mbpoll1

```
Tx = 0: Err = 0: ID = 1: F = 03: SR = 1000ms
No Connection
00000 = 0000000000000000
00001 = 0000000000000000
00002 = 0000000000000000
00003 = 0000000000000000
00004 = 0000000000000000
00005 = 0000000000000000
00006 = 0000000000000000
00007 = 0000000000000000
00008 = 0000000000000000
00009 = 0000000000000000
```

For Help, press F1. For Edit, double click on a value.

192.168.2.4: 502

Hit F3 or click *Connection->connect*. However, instead of specifying a COM port, we will specify TCP/IP and the IP address of the Slave (192.168.2.4 in this case, but different for you). Note the Well-Known Modbus port number of 502. Do not change this.


Connection

TCP/IP
9600 Baud
8 Data bits
Odd Parity
1 Stop Bit

Mode: ● RTU ○ ASCII

Response Timeout: 1000 [ms]

Delay Between Polls: 10 [ms]

Remote Server
IP Address: 192.168.2.4
Port: 502

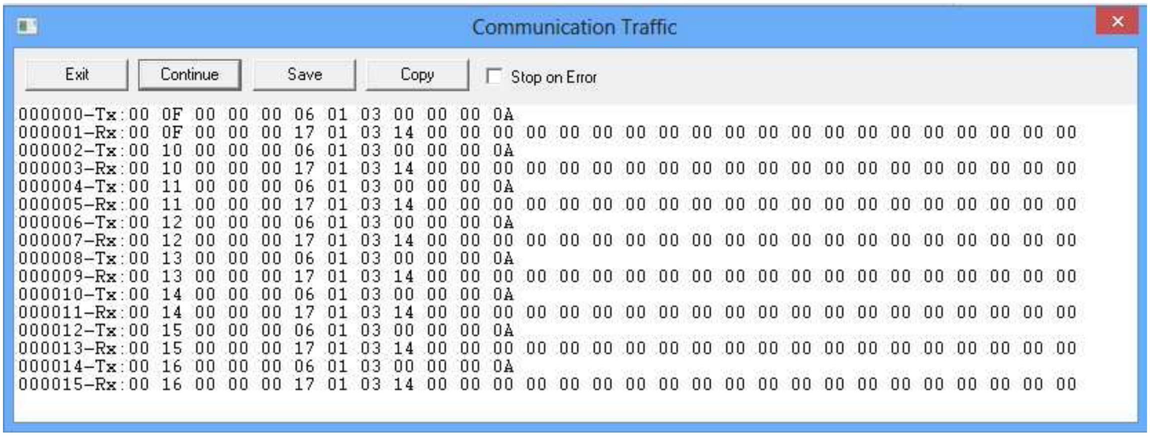
If all goes according to plan, connection will be established. If a red 'timeout' message appears on either side, do the following.

- Click *Disconnect*
- Check the IP address and port number
- Reconnect

If it still does not work, click *Display->Communication* and check if you are at least getting messages sent (Tx). If this is the case, the Slave is not responding. Check Modbus Slave settings.
Let's proceed.

Click *Display->communications* and observe the traffic between master and slave. Remember that it is as seen from the master's perspective.

Let's now compare the messages with the ones you obtained with Modbus RTU earlier.

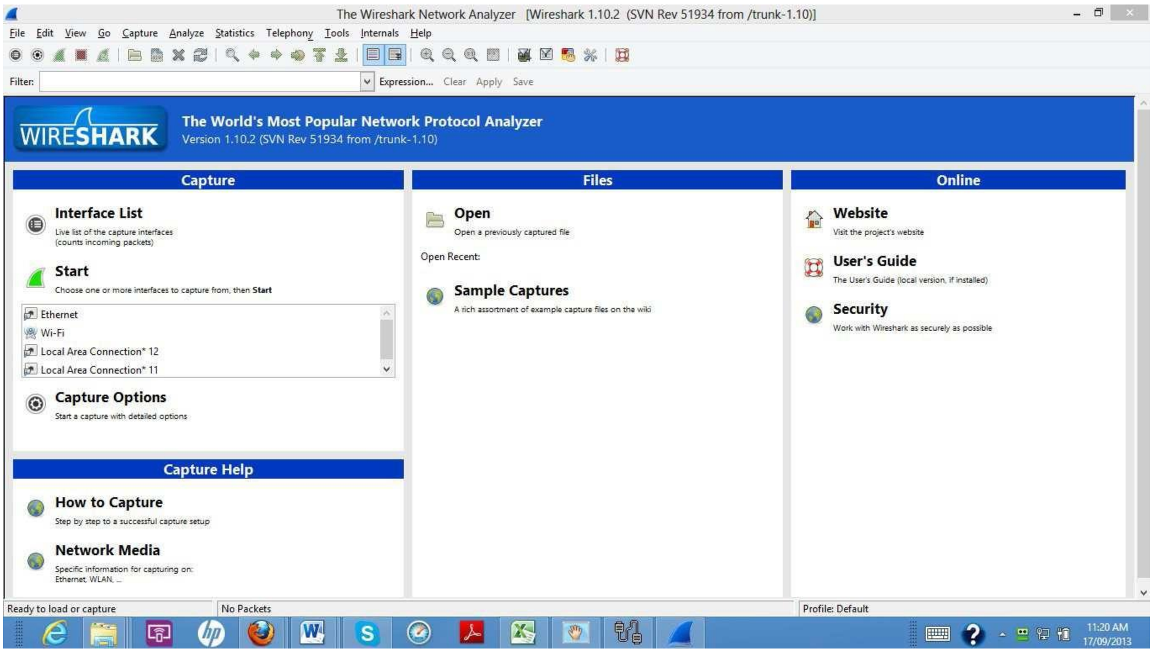

Communication Traffic

```
000000-Tx:00 0F 00 00 00 06 01 03 00 00 00 0A
000001-Rx:00 0F 00 00 00 17 01 03 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002-Tx:00 10 00 00 00 06 01 03 00 00 00 0A
000003-Rx:00 10 00 00 00 17 01 03 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000004-Tx:00 11 00 00 00 06 01 03 00 00 00 0A
000005-Rx:00 11 00 00 00 17 01 03 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006-Tx:00 12 00 00 00 06 01 03 00 00 00 0A
000007-Rx:00 12 00 00 00 17 01 03 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000008-Tx:00 13 00 00 00 06 01 03 00 00 00 0A
000009-Rx:00 13 00 00 00 17 01 03 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010-Tx:00 14 00 00 00 06 01 03 00 00 00 0A
000011-Rx:00 14 00 00 00 17 01 03 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000012-Tx:00 15 00 00 00 06 01 03 00 00 00 0A
000013-Rx:00 15 00 00 00 17 01 03 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000014-Tx:00 16 00 00 00 06 01 03 00 00 00 0A
000015-Rx:00 16 00 00 00 17 01 03 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

When we look at the Modbus/TCP messages, as opposed to the Modbus Serial messages, we notice the original PDU in there (03 00 00 00 0A), plus several extra bytes in the beginning (e.g. 0F 00 00 00 06, as well as the absence of a two-byte checksum at the end.

Notice how the Transaction Identifier increments after every Request/Response pair.
Run Wireshark by clicking on the shark fin icon.

Wireshark will open up.



The Wireshark Network Analyzer [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

WIRESHARK

The World's Most Popular Network Protocol Analyzer
Version 1.10.2 (SVN Rev 51934 from /trunk-1.10)

Capture

Interface List
Live list of the capture interfaces (counts incoming packets)

Start
Choose one or more interfaces to capture from, then **Start**

- Ethernet
- Wi-Fi
- Local Area Connection* 12
- Local Area Connection* 11

Capture Options
Start a capture with detailed options

Capture Help

How to Capture
Step by step to a successful capture setup

Network Media
Specific information for capturing on: Ethernet, WLAN, ...

Files

Open
Open a previously captured file

Open Recent:

Sample Captures
A rich assortment of example capture files on the wiki

Online

Website
Visit the project's website

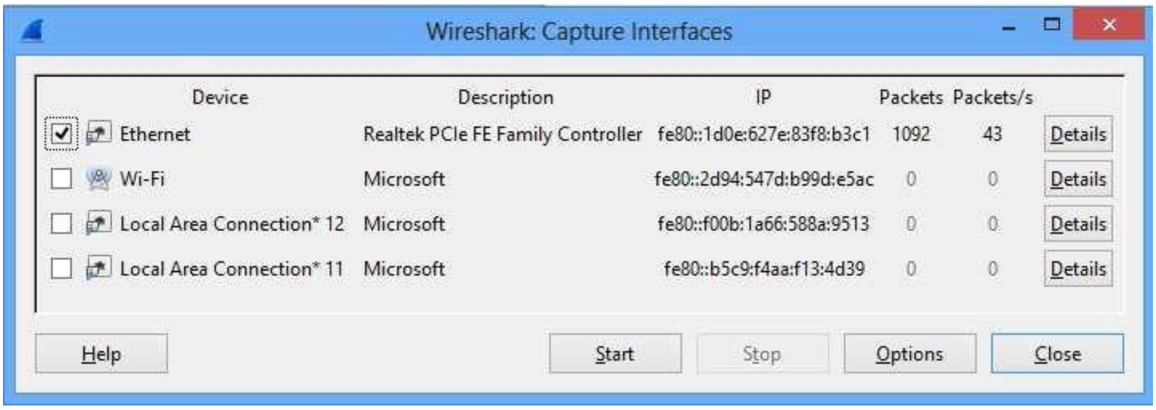
User's Guide
The User's Guide (local version, if installed)

Security
Work with Wireshark as securely as possible

Ready to load or capture | No Packets | Profile: Default

11:20 AM
17/09/2013

Click on *Capture->Interfaces*, tick the box that corresponds with the Ethernet interface (the one showing traffic) and click *Start.*

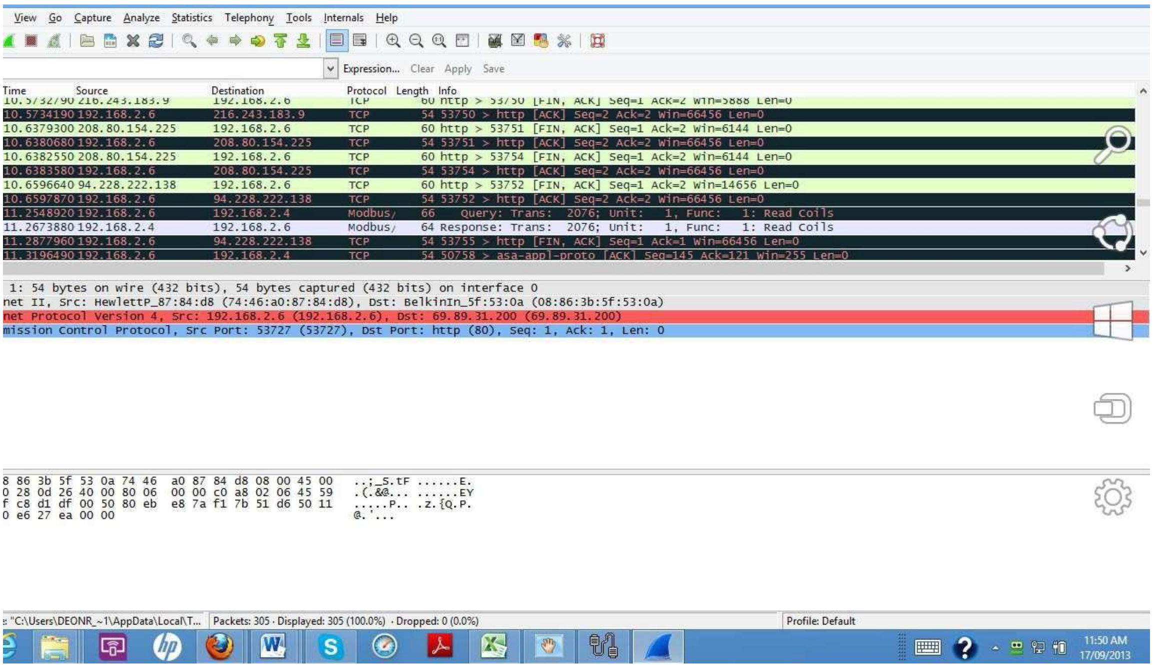


Wireshark: Capture Interfaces

	Device	Description	IP	Packets	Packets/s	
☑	Ethernet	Realtek PCIe FE Family Controller	fe80::1d0e:627e:83f8:b3c1	1092	43	Details
□	Wi-Fi	Microsoft	fe80::2d94:547d:b99d:e5ac	0	0	Details
□	Local Area Connection* 12	Microsoft	fe80::f00b:1a66:588a:9513	0	0	Details
□	Local Area Connection* 11	Microsoft	fe80::b5c9:f4aa:f13:4d39	0	0	Details

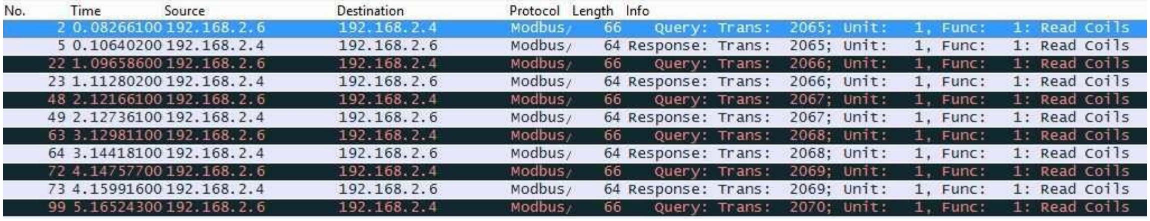
You will see packets being captured. Capture for a few seconds, then hit the square red Stop button in the top left-hand side of the screen (just below 'View').

Divide the screen into three equally-sized partitions by dragging the horizontal dividing lines up or down.


View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

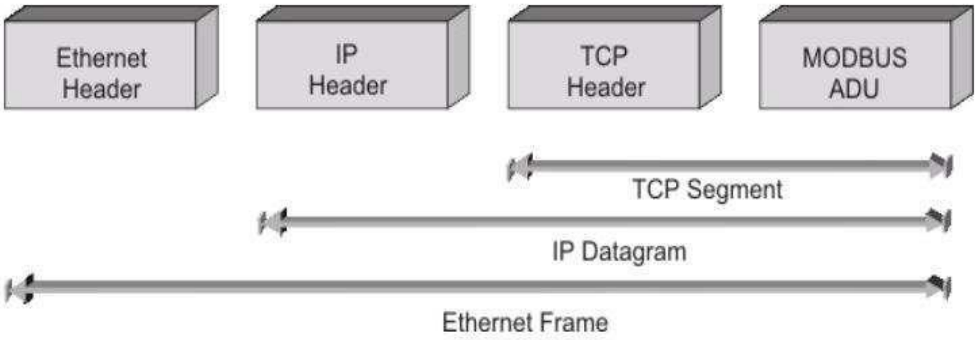
```
Time        Source           Destination      Protocol Length Info
10.5732790 216.243.183.9     192.168.2.6      TCP      60 http > 53750 [FIN, ACK] Seq=1 Ack=2 Win=5888 Len=0
10.5734190 192.168.2.6       216.243.183.9    TCP      54 53750 > http [ACK] Seq=2 Ack=2 Win=66456 Len=0
10.6379300 208.80.154.225    192.168.2.6      TCP      60 http > 53751 [FIN, ACK] Seq=1 Ack=2 Win=6144 Len=0
10.6380680 192.168.2.6       208.80.154.225   TCP      54 53751 > http [ACK] Seq=2 Ack=2 Win=66456 Len=0
10.6382550 208.80.154.225    192.168.2.6      TCP      60 http > 53754 [FIN, ACK] Seq=1 Ack=2 Win=6144 Len=0
10.6383580 192.168.2.6       208.80.154.225   TCP      54 53754 > http [ACK] Seq=2 Ack=2 Win=66456 Len=0
10.6596640 94.228.222.138    192.168.2.6      TCP      60 http > 53752 [FIN, ACK] Seq=1 Ack=2 Win=14656 Len=0
10.6597870 192.168.2.6       94.228.222.138   TCP      54 53752 > http [ACK] Seq=2 Ack=2 Win=66456 Len=0
11.2548920 192.168.2.6       192.168.2.4      Modbus/  66    Query: Trans:  2076; Unit:   1, Func:   1: Read Coils
11.2673880 192.168.2.4       192.168.2.6      Modbus/  64 Response: Trans:  2076; Unit:   1, Func:   1: Read Coils
11.2877960 192.168.2.6       94.228.222.138   TCP      54 53755 > http [FIN, ACK] Seq=1 Ack=1 Win=66456 Len=0
11.3196490 192.168.2.6       192.168.2.4      TCP      54 50758 > asa-appl-proto [ACK] Seq=145 Ack=121 Win=255 Len=0

 1: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
net II, Src: HewlettP_87:84:d8 (74:46:a0:87:84:d8), Dst: BelkinIn_5f:53:0a (08:86:3b:5f:53:0a)
net Protocol Version 4, Src: 192.168.2.6 (192.168.2.6), Dst: 69.89.31.200 (69.89.31.200)
mission Control Protocol, Src Port: 53727 (53727), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0

8 86 3b 5f 53 0a 74 46  a0 87 84 d8 08 00 45 00   ..;_S.tF ......E.
0 28 0d 26 40 00 80 06  00 00 c0 a8 02 06 45 59   .(.&@... ......EY
f c8 d1 df 00 50 80 eb  e8 7a f1 7b 51 d6 50 11   .....P.. .z.{Q.P.
0 e6 27 ea 00 00                                  @.'...

: "C:\Users\DEONR_~1\AppData\Local\T...  Packets: 305 · Displayed: 305 (100.0%) · Dropped: 0 (0.0%)  Profile: Default
```

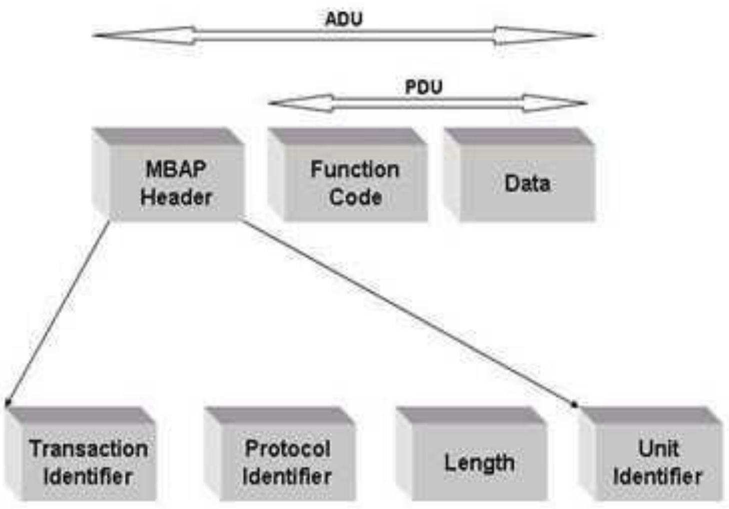
The next step is to hide the clutter. Type *mbtcp* in the filter box, and hit *Apply*. Now you will only see MODBUS/TCP Queries and Responses.



```
No.  Time       Source        Destination   Protocol Length Info
  2 0.08266100 192.168.2.6    192.168.2.4   Modbus/  66    Query: Trans:  2065; Unit:   1, Func:   1: Read Coils
  5 0.10640200 192.168.2.4    192.168.2.6   Modbus/  64 Response: Trans:  2065; Unit:   1, Func:   1: Read Coils
 22 1.09658600 192.168.2.6    192.168.2.4   Modbus/  66    Query: Trans:  2066; Unit:   1, Func:   1: Read Coils
 23 1.11280200 192.168.2.4    192.168.2.6   Modbus/  64 Response: Trans:  2066; Unit:   1, Func:   1: Read Coils
 48 2.12166100 192.168.2.6    192.168.2.4   Modbus/  66    Query: Trans:  2067; Unit:   1, Func:   1: Read Coils
 49 2.12736100 192.168.2.4    192.168.2.6   Modbus/  64 Response: Trans:  2067; Unit:   1, Func:   1: Read Coils
 63 3.12981100 192.168.2.6    192.168.2.4   Modbus/  66    Query: Trans:  2068; Unit:   1, Func:   1: Read Coils
 64 3.14418100 192.168.2.4    192.168.2.6   Modbus/  64 Response: Trans:  2068; Unit:   1, Func:   1: Read Coils
 72 4.14757700 192.168.2.6    192.168.2.4   Modbus/  66    Query: Trans:  2069; Unit:   1, Func:   1: Read Coils
 73 4.15991600 192.168.2.4    192.168.2.6   Modbus/  64 Response: Trans:  2069; Unit:   1, Func:   1: Read Coils
 99 5.16524300 192.168.2.6    192.168.2.4   Modbus/  66    Query: Trans:  2070; Unit:   1, Func:   1: Read Coils
```

First, observe how the Modbus ADU carried by TCP, IP and Ethernet.



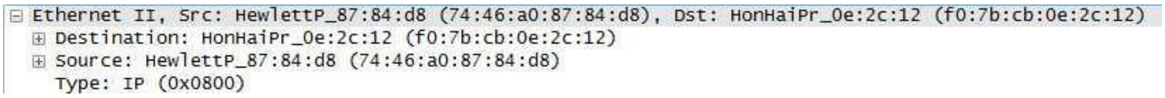


The Modbus/TCP ADU is created by omitting the checksum, and adding the Transaction Identifier, Protocol Identifier and Length fields to the original Slave Address (a.k.a Unit Identifier).

Let's have a look at what we have captured.

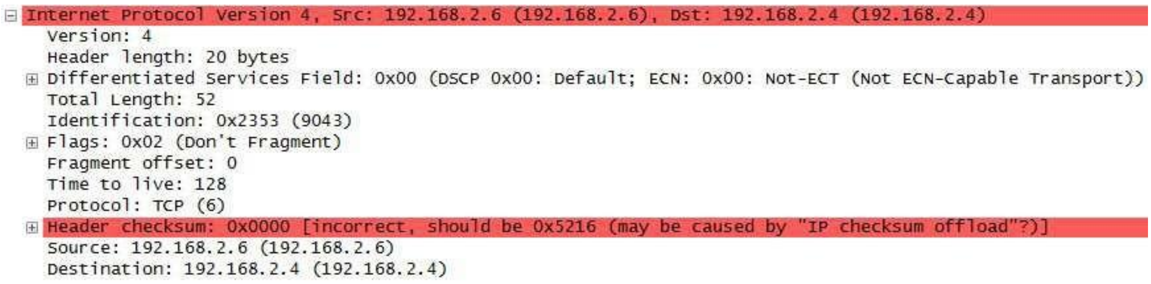
Select any Query message in the top section of the display. The middle section will show the makeup of that particular packet (a.k.a. frame, or message) while the bottom portion of the frame will show the actual hex (left) and ASCII (if a byte represents a valid ASCII character) on the right.

Click on the [+] next to the Ethernet header, observe the MAC addresses, then collapse it again.



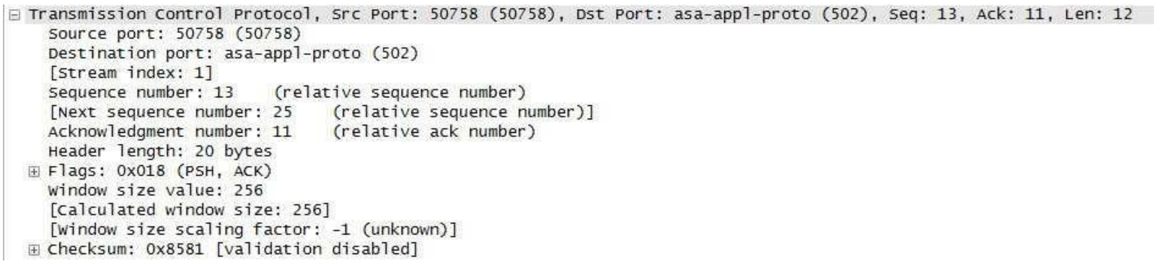
```
⊟ Ethernet II, Src: HewlettP_87:84:d8 (74:46:a0:87:84:d8), Dst: HonHaiPr_0e:2c:12 (f0:7b:cb:0e:2c:12)
  ⊞ Destination: HonHaiPr_0e:2c:12 (f0:7b:cb:0e:2c:12)
  ⊞ Source: HewlettP_87:84:d8 (74:46:a0:87:84:d8)
    Type: IP (0x0800)
```

Click on the [+] next to the IP header, observe the IP addresses, then collapse it again.



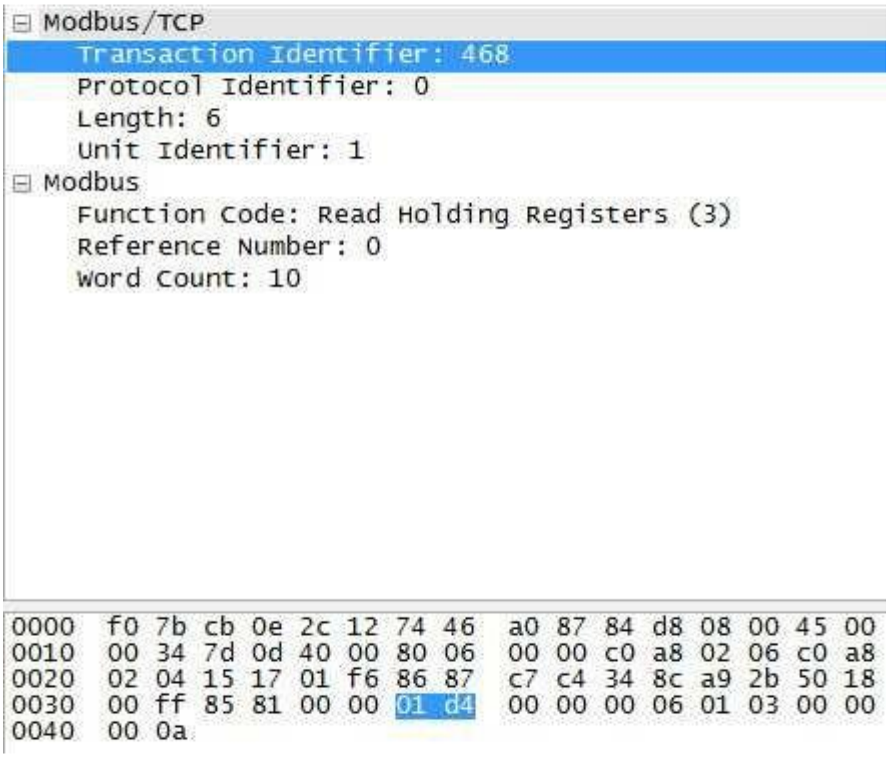
```
⊟ Internet Protocol Version 4, Src: 192.168.2.6 (192.168.2.6), Dst: 192.168.2.4 (192.168.2.4)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 52
    Identification: 0x2353 (9043)
  ⊞ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
  ⊞ Header checksum: 0x0000 [incorrect, should be 0x5216 (may be caused by "IP checksum offload"?)]
    Source: 192.168.2.6 (192.168.2.6)
    Destination: 192.168.2.4 (192.168.2.4)
```

Click on the [+] next to the TCP header, observe the port numbers, then collapse it again. Note the Well-Known port number (502) on the server (Slave) side vs. the Registered port number (>1023) on the client side.



```
⊟ Transmission Control Protocol, Src Port: 50758 (50758), Dst Port: asa-appl-proto (502), Seq: 13, Ack: 11, Len: 12
    Source port: 50758 (50758)
    Destination port: asa-appl-proto (502)
    [Stream index: 1]
    Sequence number: 13    (relative sequence number)
    [Next sequence number: 25    (relative sequence number)]
    Acknowledgment number: 11    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x018 (PSH, ACK)
    Window size value: 256
    [Calculated window size: 256]
    [Window size scaling factor: -1 (unknown)]
  ⊞ Checksum: 0x8581 [validation disabled]
```

Now we get to the Modbus ADU. Open up both [+] Modbus/TCP and [+] Modbus. Select the Modbus/TCP headline (see below) and notice how the relevant bytes at the bottom of the screen are highlighted.



```
⊟ Modbus/TCP
    Transaction Identifier: 468
    Protocol Identifier: 0
    Length: 6
    Unit Identifier: 1
⊟ Modbus
    Function Code: Read Holding Registers (3)
    Reference Number: 0
    Word Count: 10

0000  f0 7b cb 0e 2c 12 74 46  a0 87 84 d8 08 00 45 00
0010  00 34 7d 0d 40 00 80 06  00 00 c0 a8 02 06 c0 a8
0020  02 04 15 17 01 f6 86 87  c7 c4 34 8c a9 2b 50 18
0030  00 ff 85 81 00 00 01 d4  00 00 00 06 01 03 00 00
0040  00 0a
```

The Modbus ADU is made up as follows. The first four fields constitute the MBAP, the next 3 (in this example) constitute the PDU. Note the absence of a checksum as error checking is taken care of by the supporting protocols viz. TCP, IP and Ethernet.

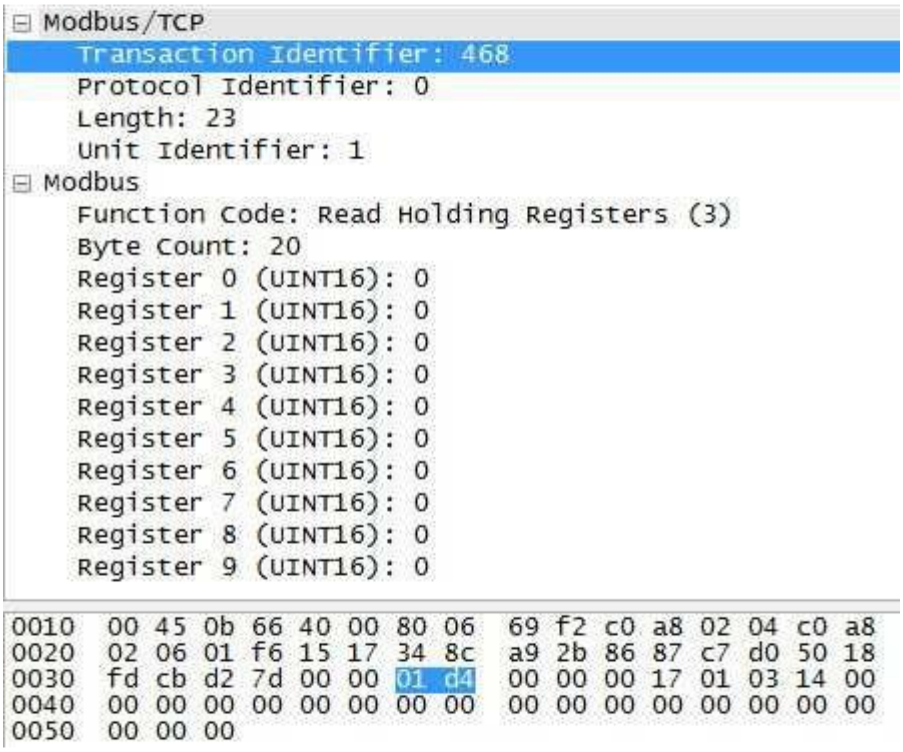
MBAP fields:

- Transaction identifier (2 bytes). This number changes for every request, but is returned with each corresponding response. Verify this by looking at the associated Response message. In the screenshot above, <01><d4> = 0x0812 = 468 decimal.
- Protocol identifier (2 bytes). In the screenshot above, <00><00> = 0x0000, since this number is 0 for Modbus.
- Length (2 bytes). This indicates the number of bytes to follow (6 in this case) and will obviously differ in the case of a response. In this case it is <00><06> = 0x0006 = 6 decimal.
- Unit identifier (1 byte). This is the slave address (1 in our case). <01> = 0x01 = 1 decimal.

PDU fields:

- Function (1 byte). This is the standard Modbus function (FC03 Read Holding Registers in our case).
- Reference number (2 bytes). This is the protocol address (NOT the physical PLC address) of the first (lowest) register. In our case it was set up to be 0. Address 0 (protocol) = address 40001 (PLC) in this case.
- Word count (2 bytes). This is the number of registers to be recovered from the slave. In this case it is 10 since <00><0a> = 0x000a = 10 decimal.

Select the associated reply in the top window to see the reply.



```
⊟ Modbus/TCP
    Transaction Identifier: 468
    Protocol Identifier: 0
    Length: 23
    Unit Identifier: 1
⊟ Modbus
    Function Code: Read Holding Registers (3)
    Byte Count: 20
    Register 0 (UINT16): 0
    Register 1 (UINT16): 0
    Register 2 (UINT16): 0
    Register 3 (UINT16): 0
    Register 4 (UINT16): 0
    Register 5 (UINT16): 0
    Register 6 (UINT16): 0
    Register 7 (UINT16): 0
    Register 8 (UINT16): 0
    Register 9 (UINT16): 0
```

```
0010  00 45 0b 66 40 00 80 06  69 f2 c0 a8 02 04 c0 a8
0020  02 06 01 f6 15 17 34 8c  a9 2b 86 87 c7 d0 50 18
0030  fd cb d2 7d 00 00 01 d4  00 00 00 17 01 03 14 00
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0050  00 00 00
```

End of Optional MODBUS TCP