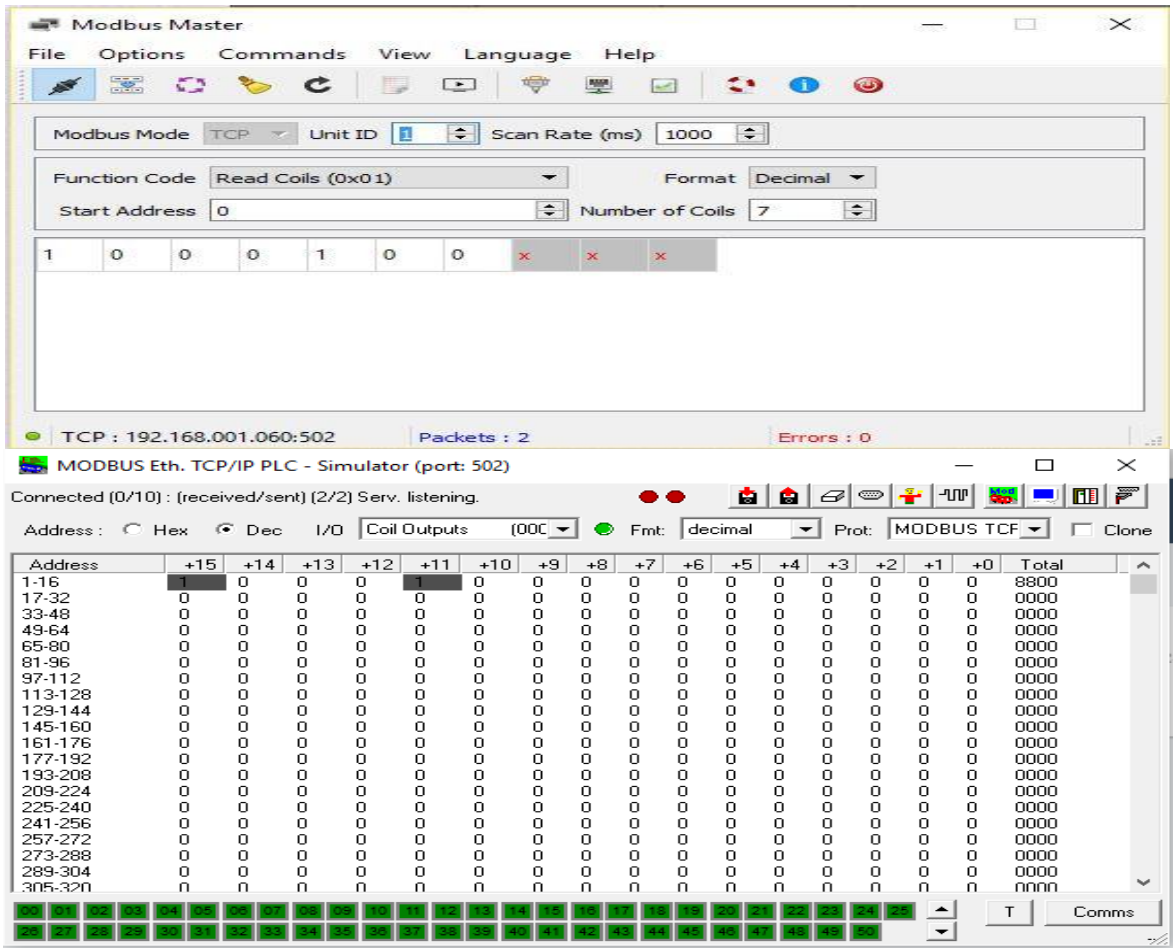
Salman Azim
Student No. 243850

Advance Course in Factory Communication

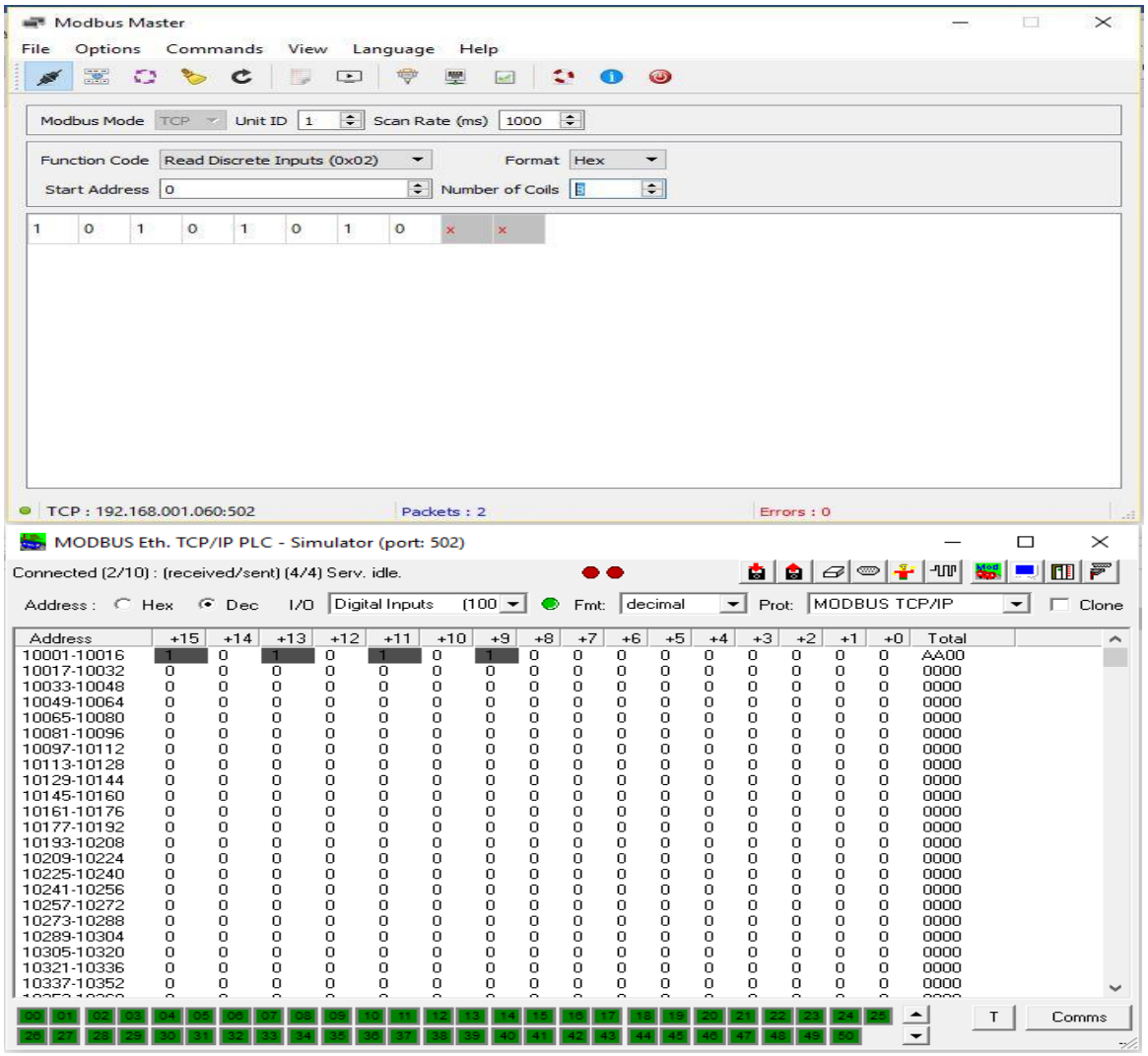
Assignment 1
MODBUS TCP

Environment Setup: For this Assignment two laptops were used. First of all, the provided softwares were downloaded and installed, in one the master application and another the slave simulator. Wireshark was installed in the pc which was selected as a master client. Then it was made sure that the two laptops are working on the same network, in this case same wireless network. Also the IP addresses of both the PC were known. It was also made sure that both the PC can communicate with each other. Then the IP address of the slave PC was assigned in Modbus Master application. After the setup both the master and slave simulator was connected and Wireshark was kept on run mode to capture the packets.

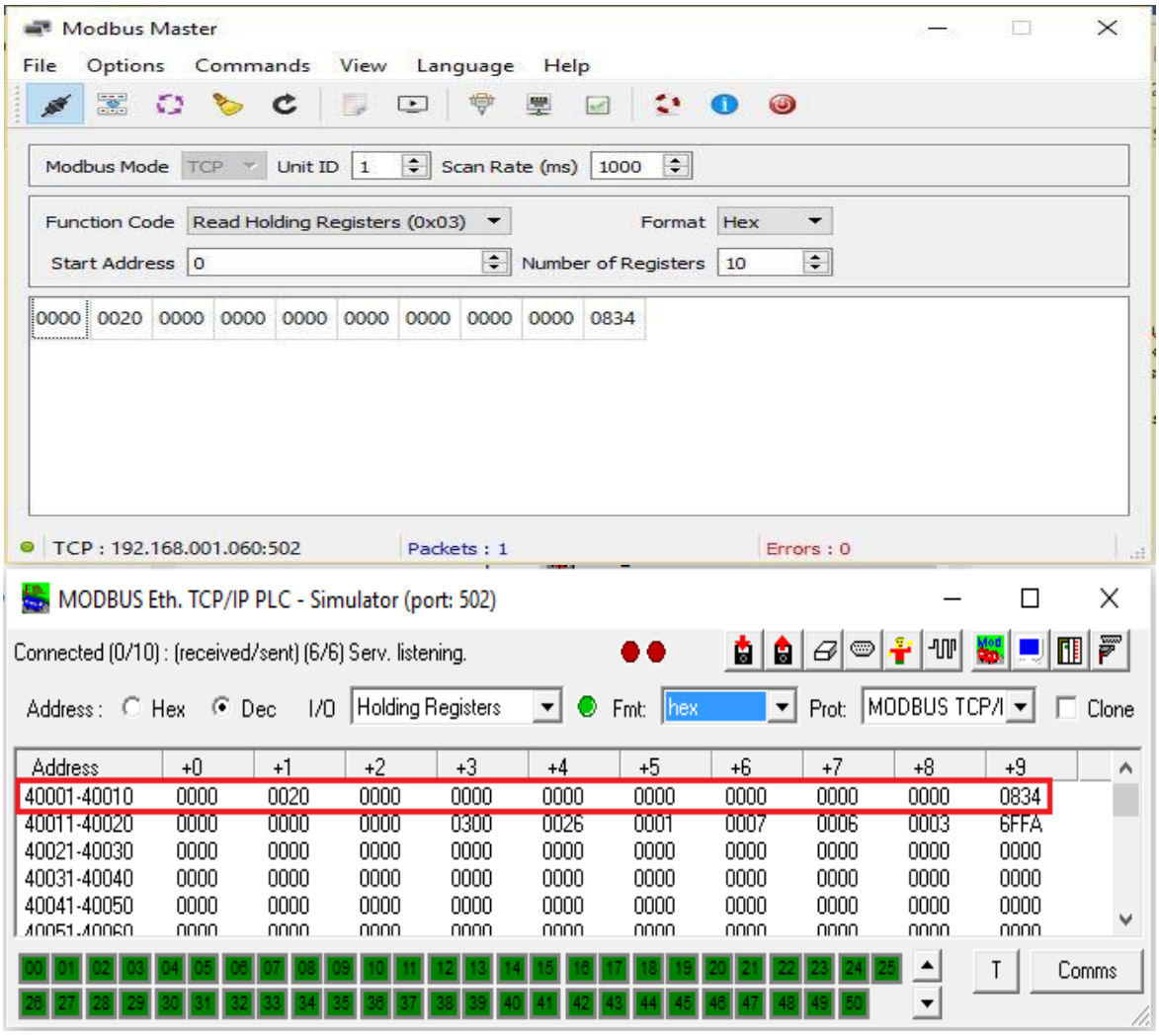
1. Some arbitrary changes were made in the slave simulator, like changing the values of some certain coils, discrete inputs and holding registers. Then the result was read from the master application and all the results were matched according to the address of the certain coils, discrete inputs and holding registers. The screenshots from both the master application and the slave simulator are provided below.



Capture 1.1 Modbus Master reading data of 7 coils from Slave simulator

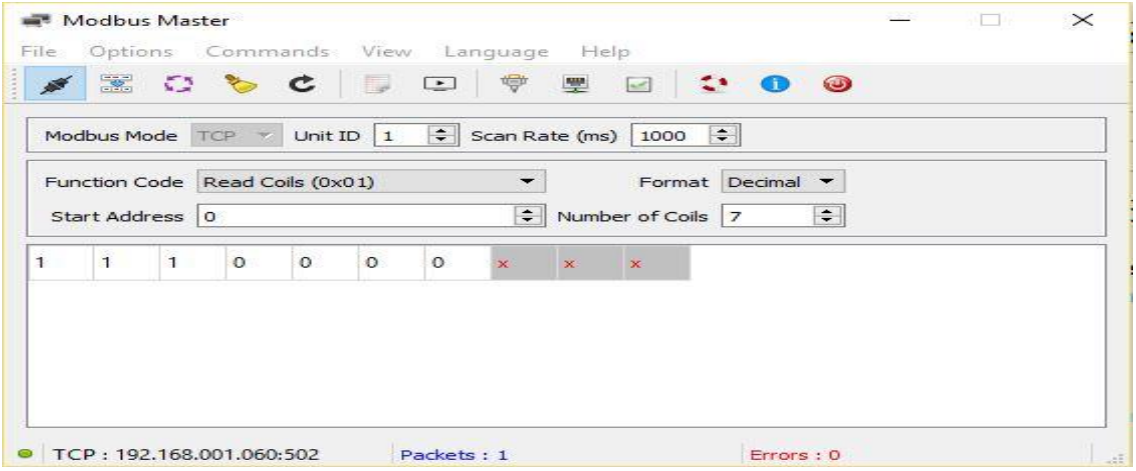


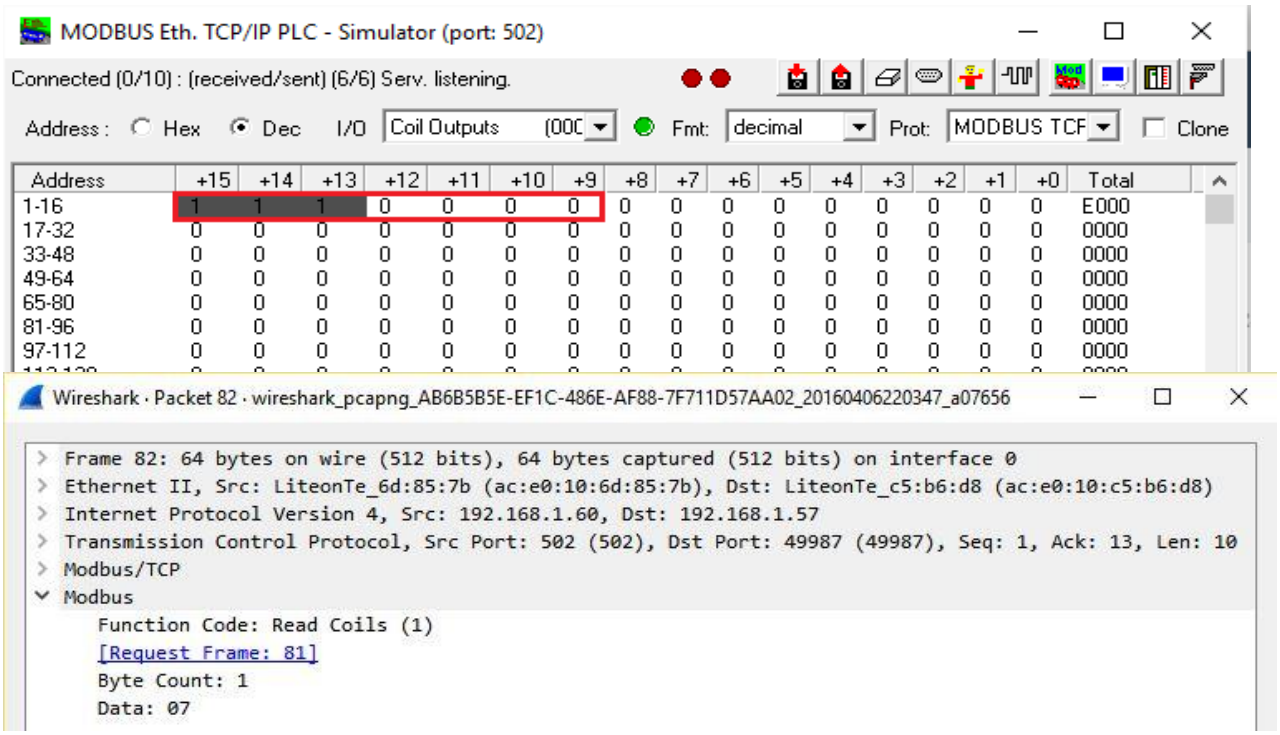
Capture 1.2 Modbus Master reading 8 Digital inputs from server simulator



Capture 1.3 Modbus Master reading status of 10 Holding Registers from server simulator

- **4.a Reading 7 coils from bit 0 and reading 4 coils from bit 1:**

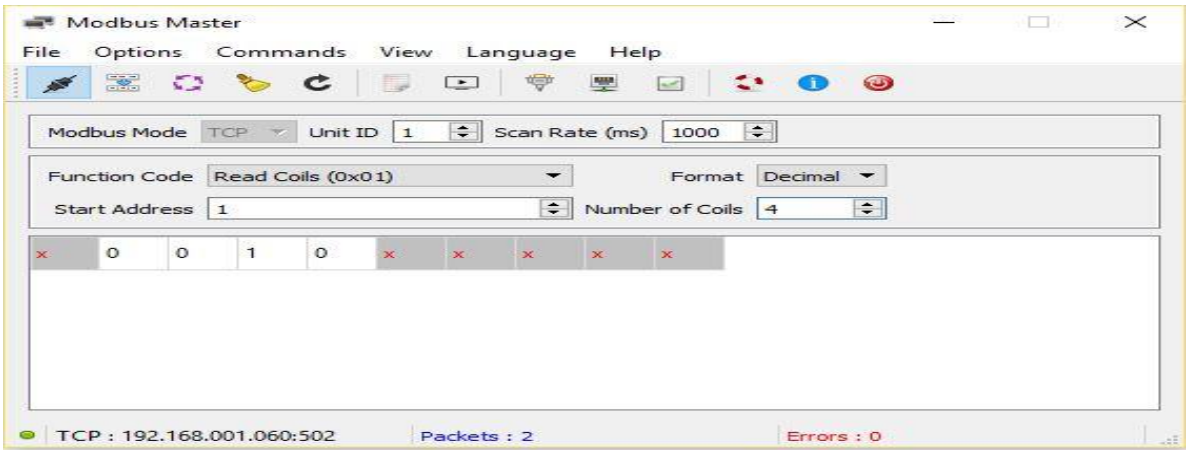


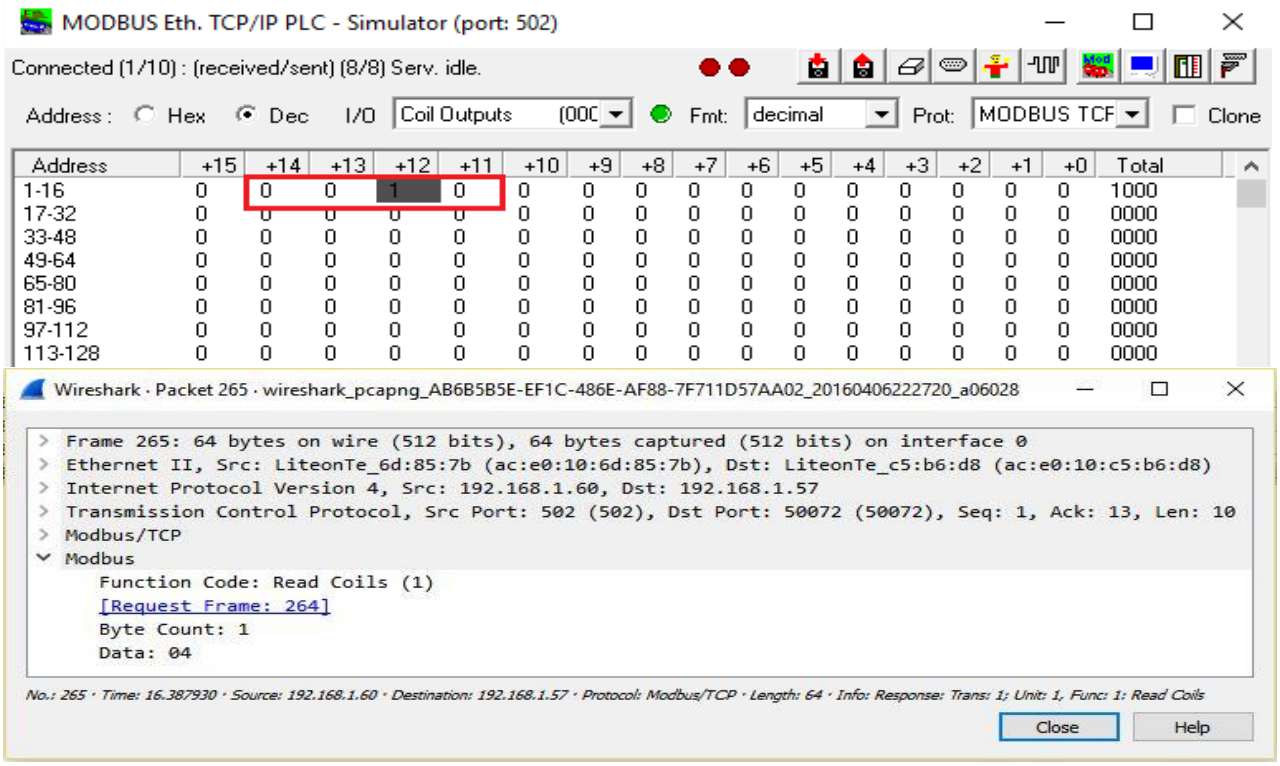


Capture 4. a1: Modbus master request for 7 coils from 0 bit, slave and wireshark response packet

Message Analysis:

- Function code matches the request by Modbus Master
- Byte count is 1 which is correct for 7 coils.
- Data is 7 bit for 7 coils.
- Source and destination of message is also ok
- Total frame size is 64 bytes



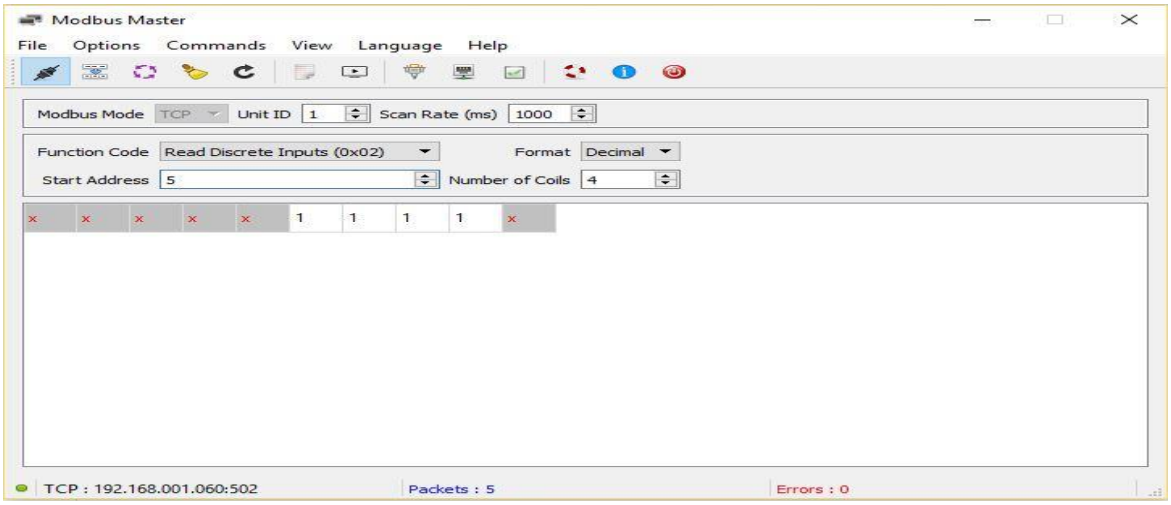


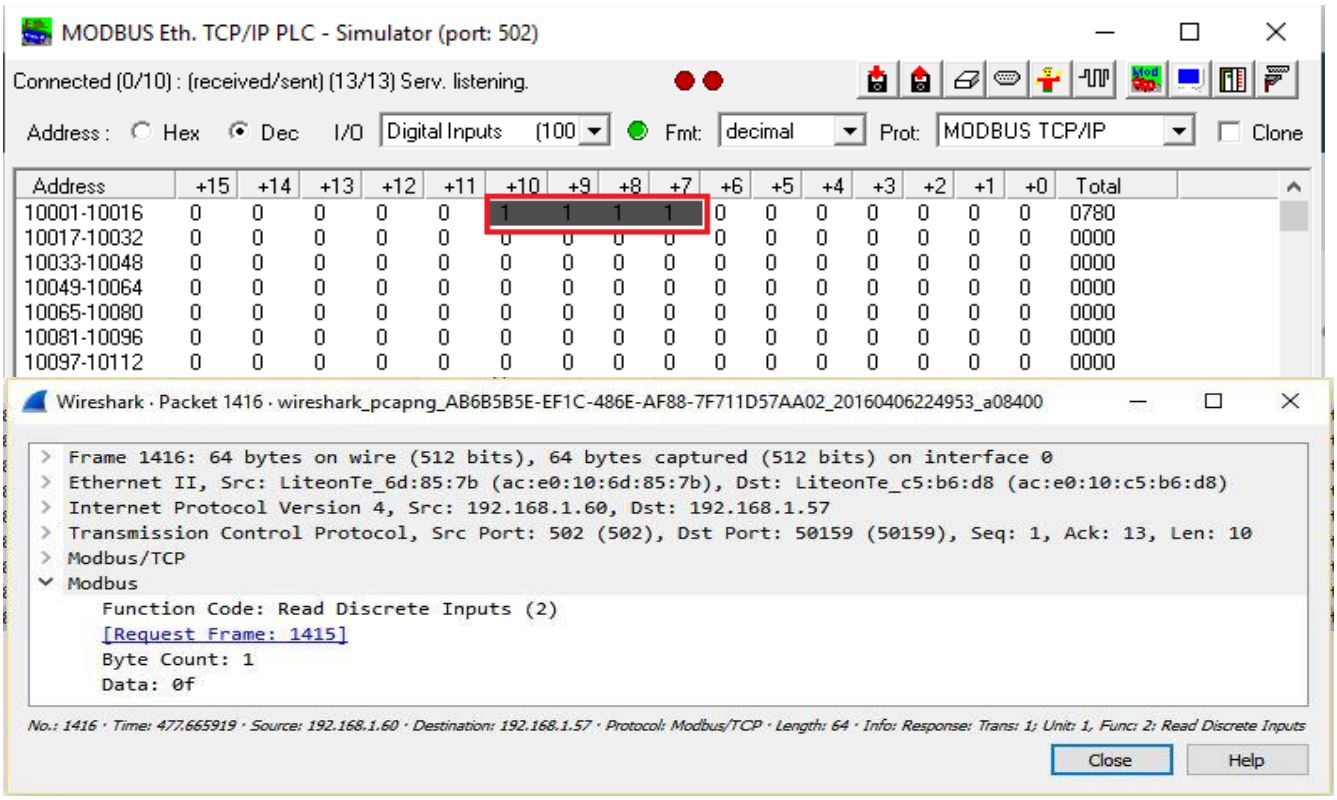
Capture 4. a2: Modbus Master request to read 4 coils from bit 1 & Wireshark response packet for reading 4 coils starting from bit 1.

Message Analysis:

- Function code matches the request by Modbus Master
- Byte count is 1 which is correct for 4 coils.
- Data is 4 bit for 4 coils.
- Source and destination of message is also ok.
- Total frame size is 64 bytes.

- **4.b Reading 4 discrete inputs starting from bit 5 & reading 11 bits starting from bit 2**

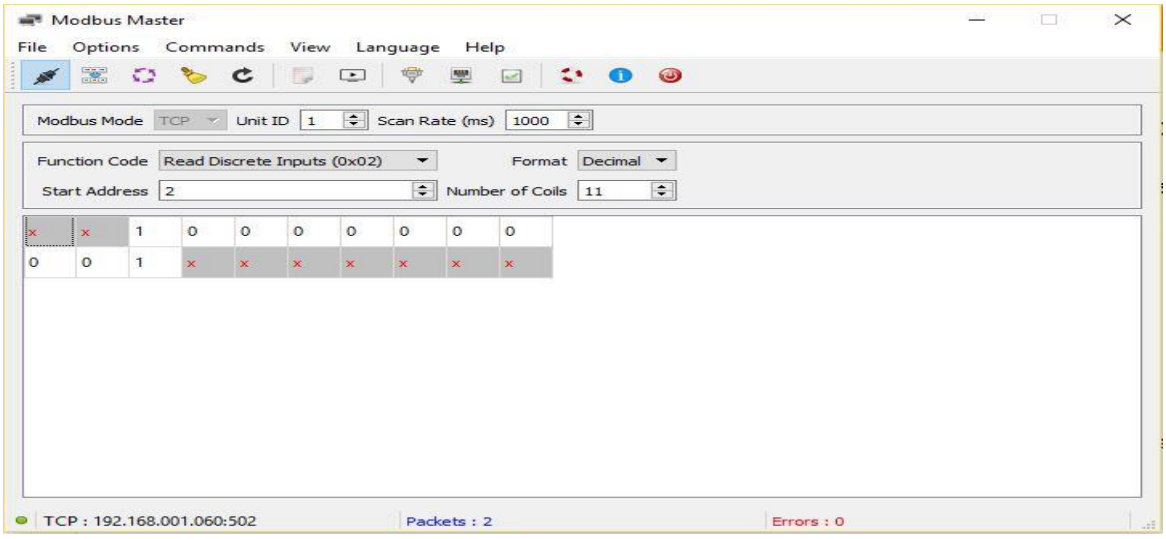


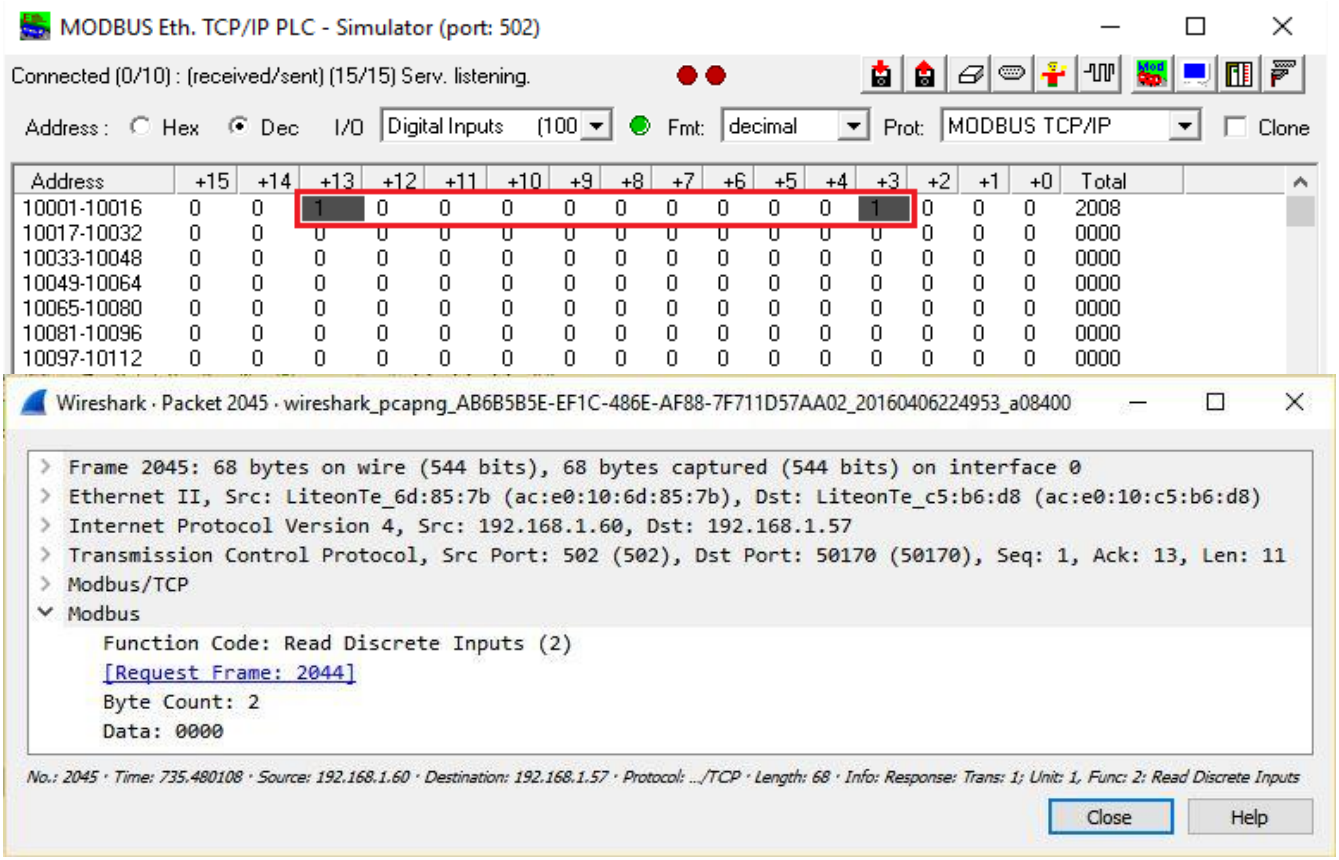


Capture 4. B1: Wireshark response packet for reading 4 digital input starting from bit 5.

Message Analysis:

- Function code matches the request by Modbus Master.
- Byte count is 1 which is correct for 4 coils.
- Data is 4 bit for 4 coils.
- Source and destination of message is also ok.
- Total frame size is 64 bytes.



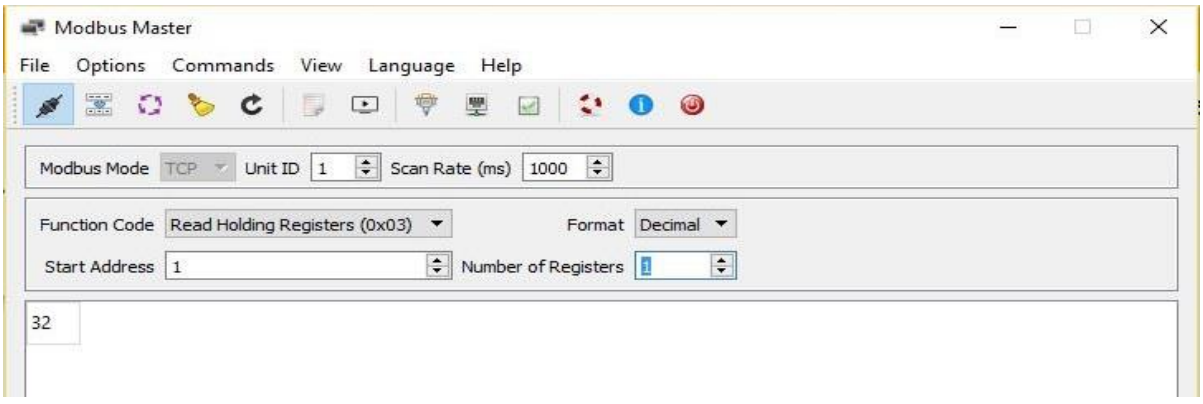


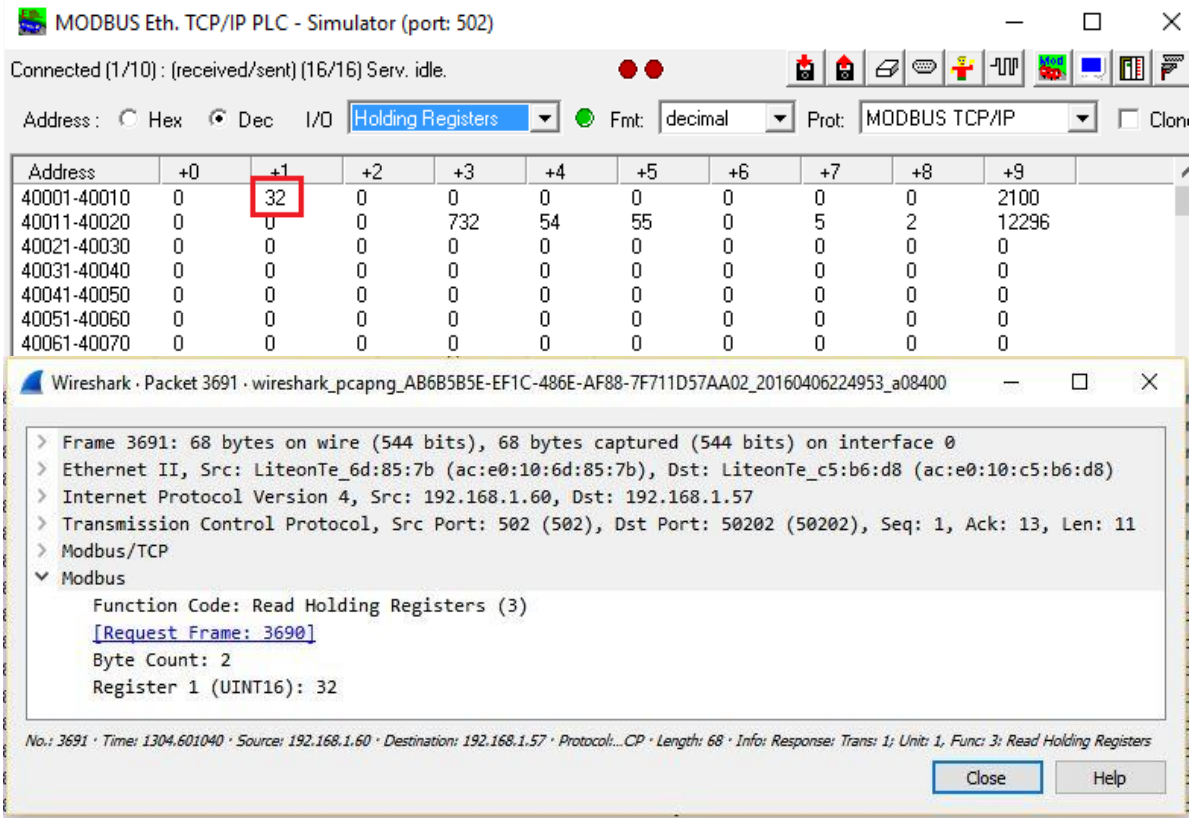
Capture 4. B2: Wireshark response packet for reading 11 digital inputs starting from bit 2.

Message Analysis:

- Function code matches the request by Modbus Master.
- Byte count is 2 which is correct for 11 coils.
- Data is 0 as all the states of the inputs are 0.
- Source and destination of message is also ok.
- Total frame size is 68 bytes.

- **4.c Reading 2nd Holding Register**



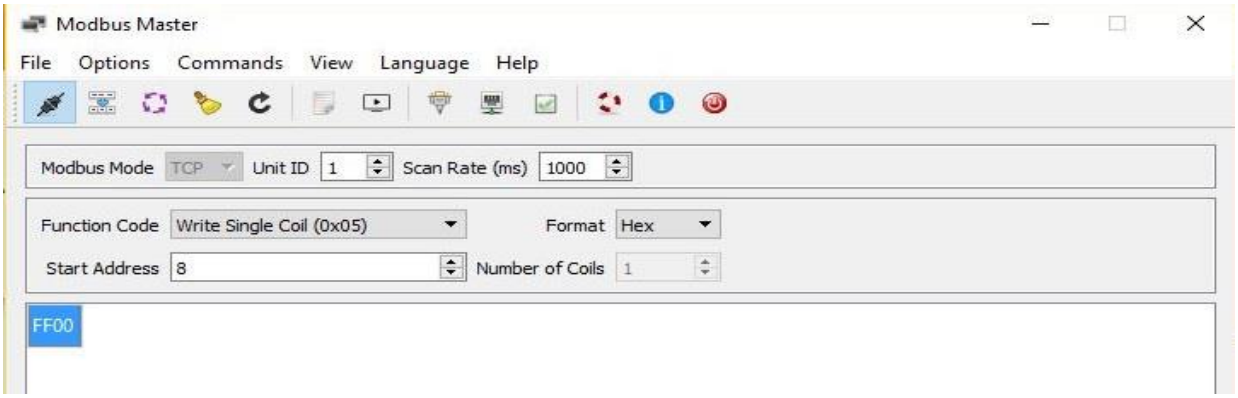


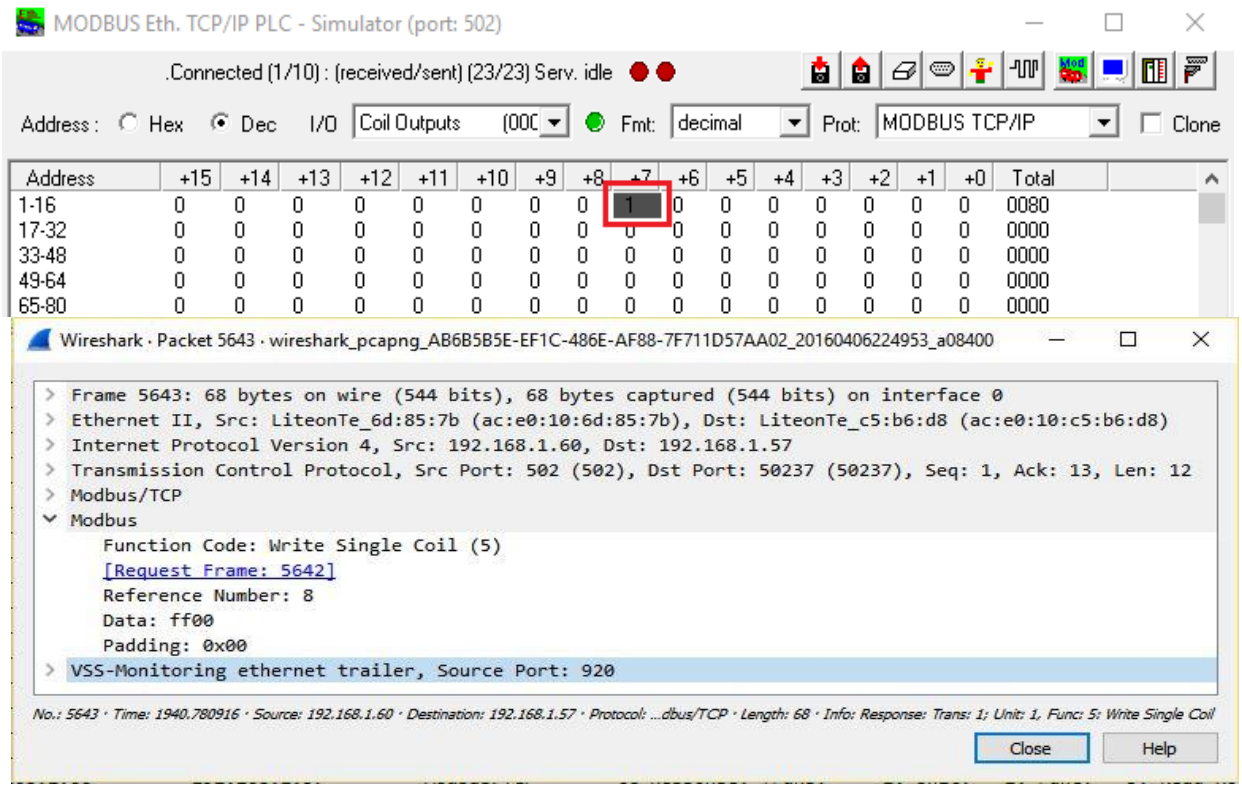
Capture 4. c: Modbus master request for reading 2nd Holding Register and wireshark response packet

Message Analysis:

- Function code matches the request by Modbus Master.
- Byte count is 2 which is correct for a holding register which consumes 16 bit.
- Data is 32.
- Source and destination of message is also ok.
- Total frame size is 68 bytes.

4.d Writing the 9th Coil value to True





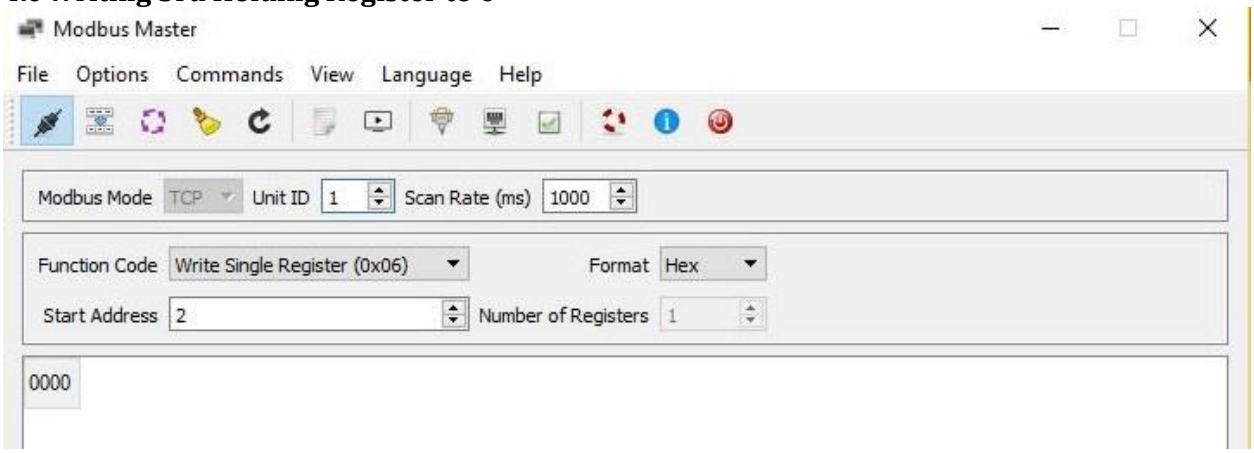
Capture 4. d: Modbus master request for setting the 9th coil to true and wireshark response packet

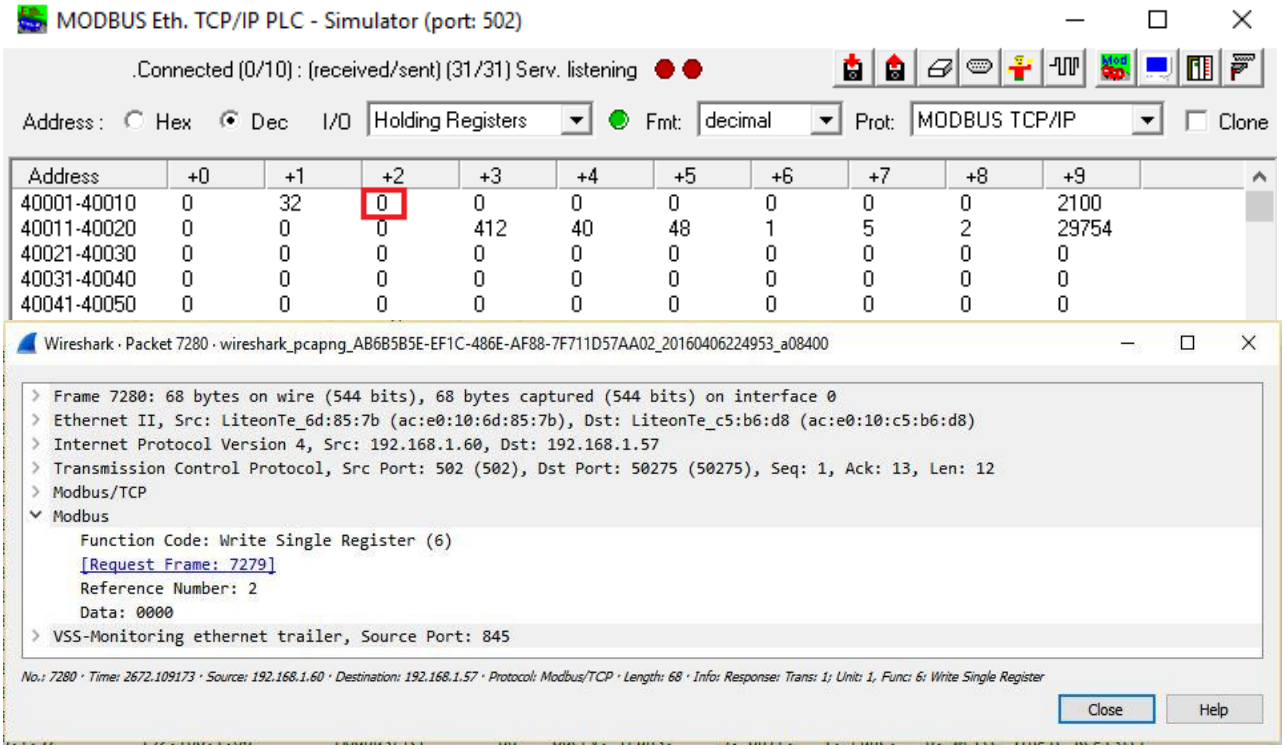
Message Analysis:

- Function code matches the request by Modbus Master.
- Reference Number is 8 as the 9th coil is set to true.
- Data is FF00 which is used for setting true value of coils.
- Source and destination of message is also ok.
- Total frame size is 68 bytes.

All the experiments tested for the 5 functions codes were observed and they matched with the Master application according to the changes made in the Slave simulator.

4.c Writing 3rd Holding Register to 0



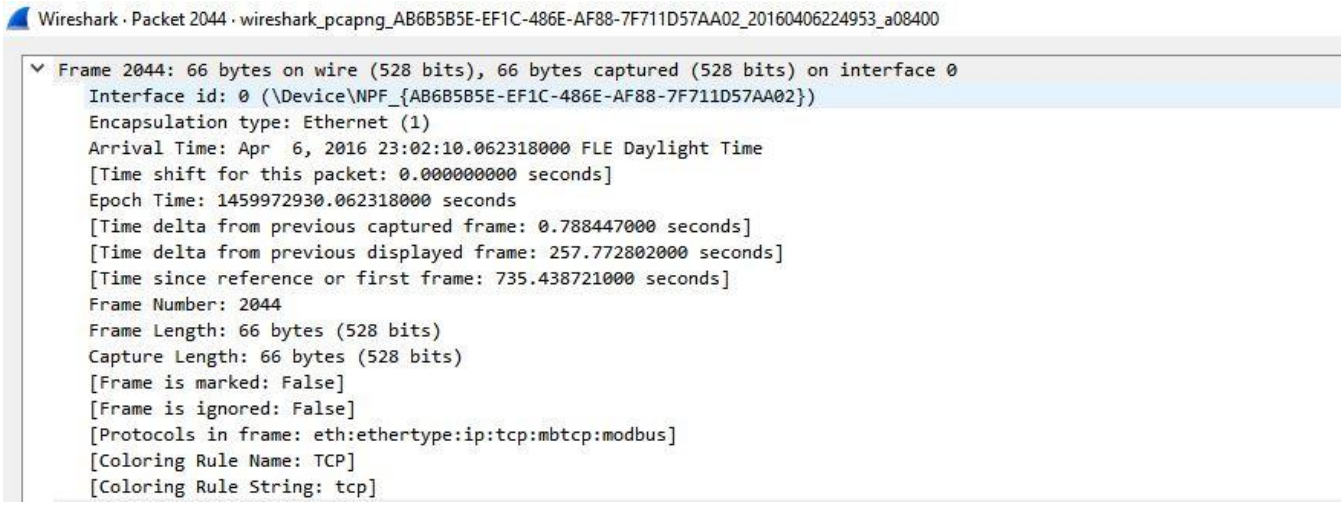


Capture 4. e: Modbus master request for setting the 3rd Holding Register to 0 and wireshark response packet

Message Analysis:

- Function code matches the request by Modbus Master.
- Reference Number is 2 as the 3rd coil is set to 0.
- Data is 0000 which is used for setting the holding register to 0.
- Source and destination of message is also ok.
- Total frame size is 68 bytes.

- **Analyzing protocol level from Wireshark packet:**



In this level the data that are encoded are:

- Transmission time for receiving this packet from slave to master.
- Frame size
- Frame Number
- Frame protocol: Ethernet

```
Ethernet II, Src: LiteonTe_c5:b6:d8 (ac:e0:10:c5:b6:d8), Dst: LiteonTe_6d:85:7b (ac:e0:10:6d:85:7b)
  Destination: LiteonTe_6d:85:7b (ac:e0:10:6d:85:7b)
      Address: LiteonTe_6d:85:7b (ac:e0:10:6d:85:7b)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: LiteonTe_c5:b6:d8 (ac:e0:10:c5:b6:d8)
      Address: LiteonTe_c5:b6:d8 (ac:e0:10:c5:b6:d8)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

- Destination and source address

```
Internet Protocol Version 4, Src: 192.168.1.57, Dst: 192.168.1.60
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0x2416 (9238)
  > Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
  > Header checksum: 0x52e8 [validation disabled]
    Source: 192.168.1.57
    Destination: 192.168.1.60
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
```

- IP Addresses of source and destination
- CRC
- Flags

```
Transmission Control Protocol, Src Port: 50170 (50170), Dst Port: 502 (502), Seq: 1, Ack: 1, Len: 12
    Source Port: 50170
    Destination Port: 502
    [Stream index: 44]
    [TCP Segment Len: 12]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 13    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 20 bytes
  > Flags: 0x018 (PSH, ACK)
    Window size value: 256
    [Calculated window size: 65536]
    [Window size scaling factor: 256]
  > Checksum: 0x4f54 [validation disabled]
    Urgent pointer: 0
  [SEQ/ACK analysis]
      [iRTT: 0.032925000 seconds]
      [Bytes in flight: 12]
    [PDU Size: 12]
```

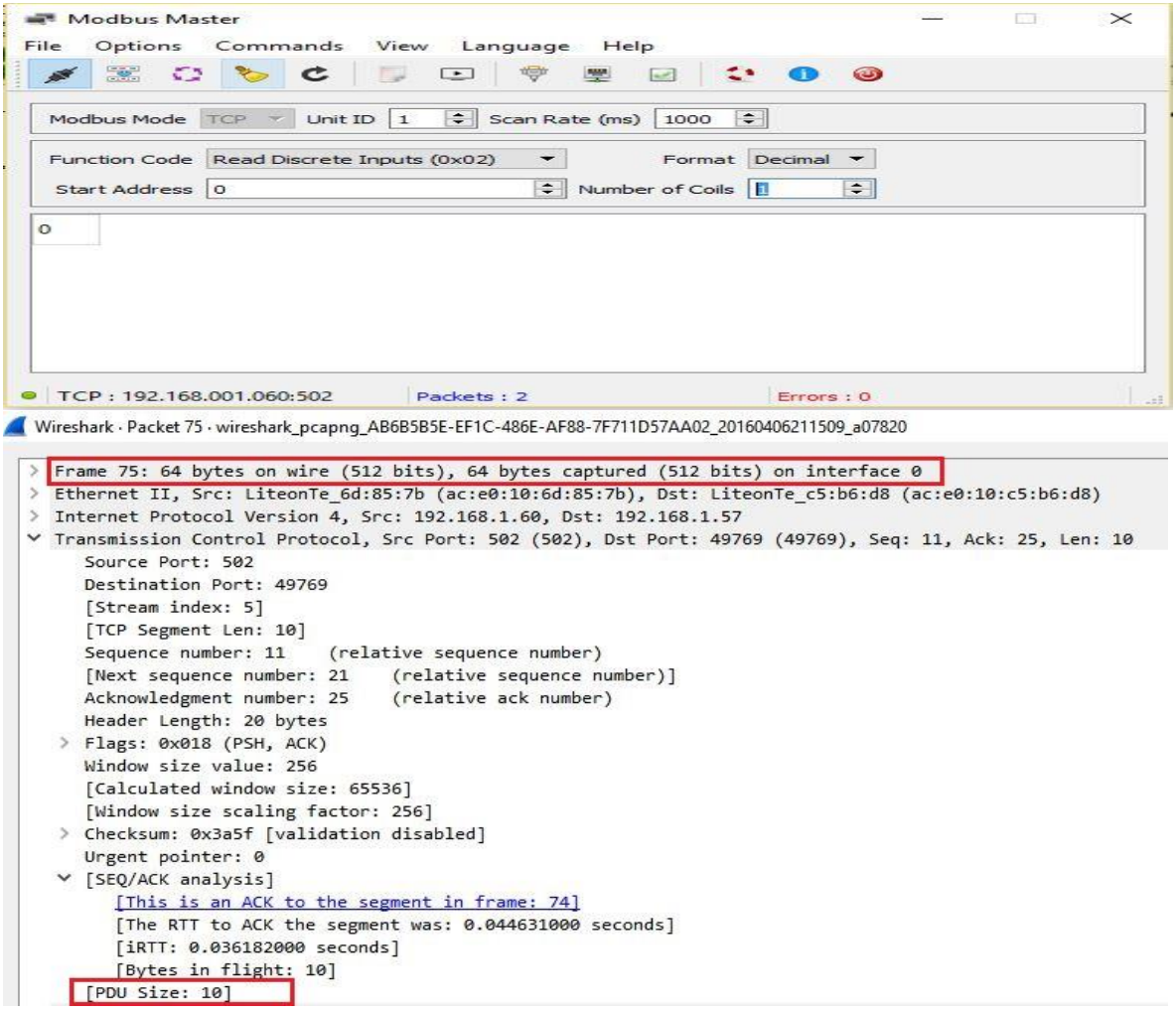
- Source port & destination port

- Acknowledgement No.
- Flags
- Checksum
- PDU Size

```
Modbus/TCP
    Transaction Identifier: 1
    Protocol Identifier: 0
    Length: 6
    Unit Identifier: 1
```

- MBAP Header description.
- Protocol Identifier 0 = Modbus Protocol.

- **Communication Overhead for reading one discrete input:**



Capture 4: Reading one discrete input request by Modbus Master & Wireshark response packet analysis

From the Wireshark message, it is shown that the data frame size is 64 bytes and the PDU size is 10 bytes.

So, the overhead for reading one discrete input is (64-10) = 54 Bytes.

- **Function Code 0*04: Read Input Register**

It is a Public Function code. Using this Function code, the master client can read the value of the input registers from the slave. It uses 16 bits to store a value. Its starting address can be 0*0000 to 0*FFFF. And the quantity of input registers can be 0*0001 to 0*007D.

Request & Response PDU:

Request PDU:
- Function Code 1 Byte = 0*04
- Starting Address = 2 Bytes [0*0000 to 0*FFFF]
- Quantity of Input Registers = 2 Bytes [*0001 to 0*007D]

Response PDU:
- Function Code 1 Byte = 0*04
- Byte Count = 1 Byte [2*Number of Registers]
- Input Registers = Number of Registers *2 Bytes

- **Transmission Time for 10 Mbps:**

  From Wireshark packet the average request data frame got was 64 Bytes and response data frame was 66 bytes. So for this amount of data the transmission time would be

  $$(66+64) * 8 \text{ Bits}/10 \text{ Mbps} = 0.000104 \text{ s} = 104 \ \mu s$$

- The average transmission time was 0.04138700 seconds.

  From Wireshark message for Modbus TCP/IP the transmission time can be found.