

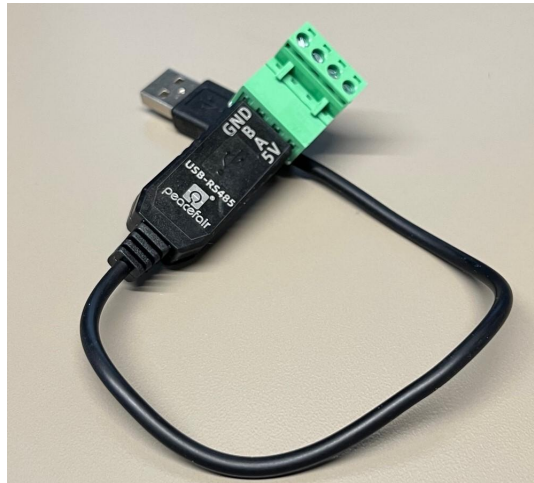
# MICS Lab 08 Outline

The 3 PLCs were connected via RS-485 to communicate over Modbus RTU, where the controller (.25) was polling the inputs of one PLC (.31, id "4") and used those to drive the outputs of another PLC (.27, id "2").

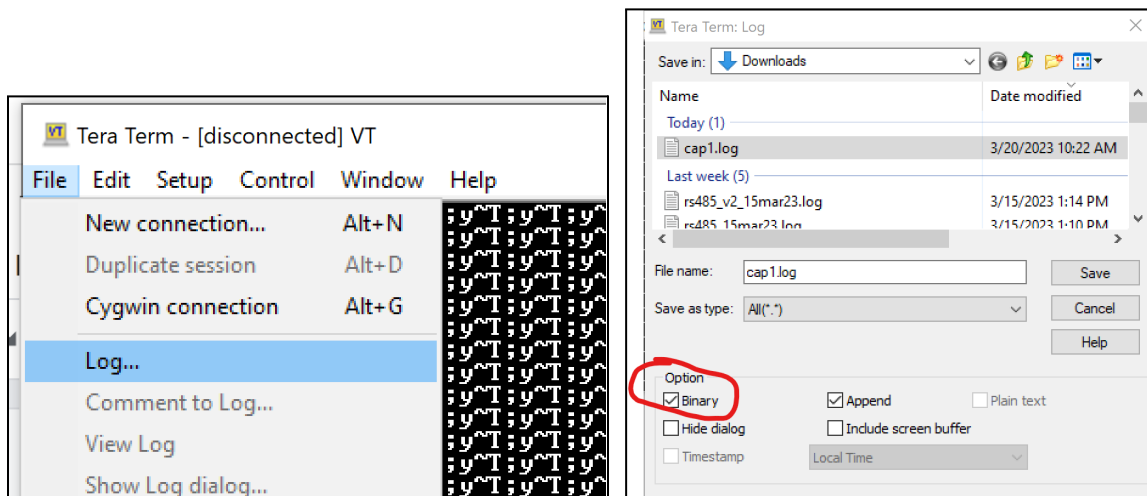
The first question on the form asks you to describe how the wiring between the RS-232 version that you did as part of Lab 07 differs from that of the current one using RS-485. What else needed to change with the PLC program?

## 1. Modbus RTU Data Collection

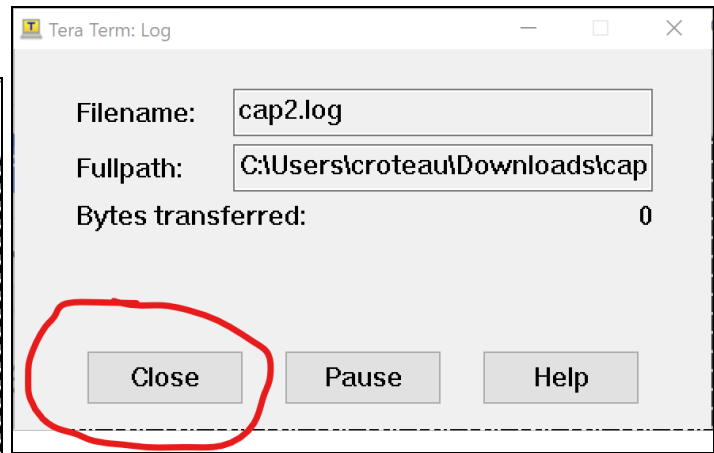
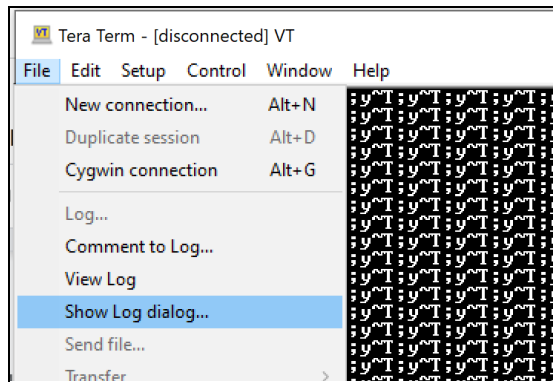
- With all three PLCs on and running, connect a single PC using the RS-485 to USB device.



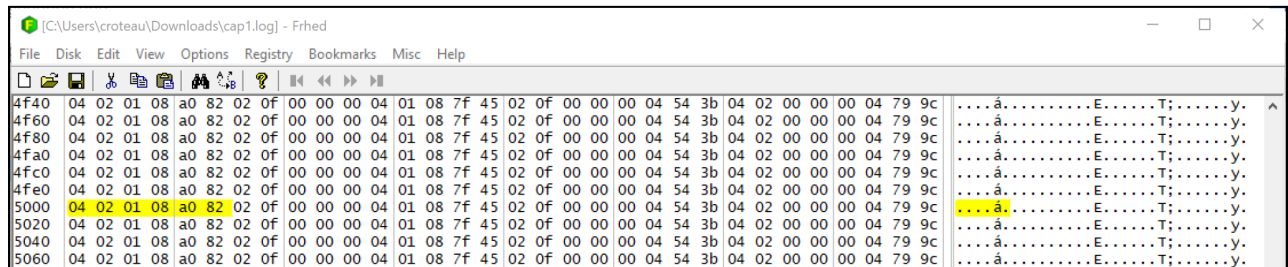
- Verify what COM port number the "C340" device is mapped to.
- Open up TeraTerm (or another equivalent terminal program) and open a new connection with the noted COM port above.
- Under "Setup->Serial Port" change the baud rate to what was set in CCW 19,200 bps. You should share seeing something like ".ET;" on the display.
- Open the "File->Log..." dialog and make sure the "Binary" option is checked. Enter a filename to a place where you can find the file later.



- Once the log file is running, flick the switches on the left (or .31) PLC to that the different lights show up on the right (.27) PLC.
- Top stop the recording select "File->Show Log dialog..." then click on the "Close" button



- Open that log file up in Frhed (or another hex/binary editor/viewer), and resize the window until the ASCII data on the right side makes vertical patterns.



- Find one of the messages that are being sent by the controller, and decode each of its parts to show what message was being sent and what the response back is from the polled peripheral.
- Verify that the CRC makes sense using [this website](#) (select CRC16\_MODBUS) and inputting the data as bytes.

CRC width

Bit length: ☐ CRC-8 ☒ CRC-16 ☐ CRC-32 ☐ CRC-64

CRC parametrization

☒ Predefined 

CRC16\_MODBUS

☐ Custom

CRC detailed parameters

Input reflected: ☒ Result reflected: ☒

Polynomial: 

0x8005

Initial Value: 

0xFFFF

Final Xor Value: 

0x0

CRC Input Data

☐ String ☒ Bytes ☐ Binary string

0x04 0x02 0x01 0x08

Show reflected lookup table: ☐ (This option does not affect the CRC calculation)

Calculate CRC!

Result CRC value: 0x82A0

- Repeat for all four of the messages and find the portion that changes when different switch selections were inputted on the PLC being read from.

## 2. Writing arbitrary Modbus RTU data

- Turn off the middle (.25) Controller PLC
- Download the modscan64.zip program from [this link](https://www.win-tech.com/html/demos.htm) and install:  
<https://www.win-tech.com/html/demos.htm>
- After unzipping the program to a new folder within your home directory, run the ModScan64.exe program.
- When the registration dialog comes up, just click "Ok" with the other fields blank. You will have to wait for two other windows to timeout with Shareware messages.
- Select "Connection->Connect" then scroll down to find the same COM port used previously.

Connection Details

Connect Using: Direct Connection to COM19

Phone Number:

Service Port: 502

Alias: USB\_dongle

Configuration

Baud Rate: 19200

Word Length: 8

Parity: NONE

Stop Bits: 1

Hardware Flow Control

☐ Wait for DSR from slave

☐ Wait for CTS from slave

DTR Control: Disable

RTS Control: Disable

Delay 0 ms after RTS before transmitting first character

Delay 0 ms after last character before releasing RTS

Protocol Selections

OK Cancel

- After connecting, pick a message and remote id that you want to send, and set the length to an appropriate length for the mapping of the remote device. Here you can start with id = 2 (or 4), FC = 0x01, and Length of 4

ModScan64 - ModScan641 (Not Connected)

File Connection Setup View Window Help

ModScan641 (Not Connected)

Address: 0001

Device Id: 1

MODBUS Point Type: 01: COIL STATUS

Number of Polls: 0

Valid Slave Responses: 0

Length: 4

Reset Ctrs

\*\* Device NOT CONNECTED! \*\*

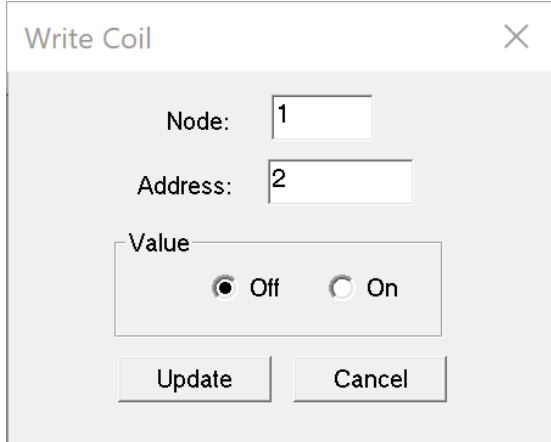
00001: <0>

00002: <0>

00003: <0>

00004: <0>

- Double click on one of the <0> values next to one of the coil addresses 0000X. In the dialog that pops up toggle the Value radio button to "On" and you should see the corresponding light come on the selected peripheral.

A screenshot of a 'Write Coil' dialog box. The dialog has a title bar with the text 'Write Coil' and a close button (X). Inside the dialog, there are two input fields: 'Node:' with the value '1' and 'Address:' with the value '2'. Below these is a 'Value' section containing two radio buttons: 'Off' (which is selected) and 'On'. At the bottom of the dialog are two buttons: 'Update' and 'Cancel'.

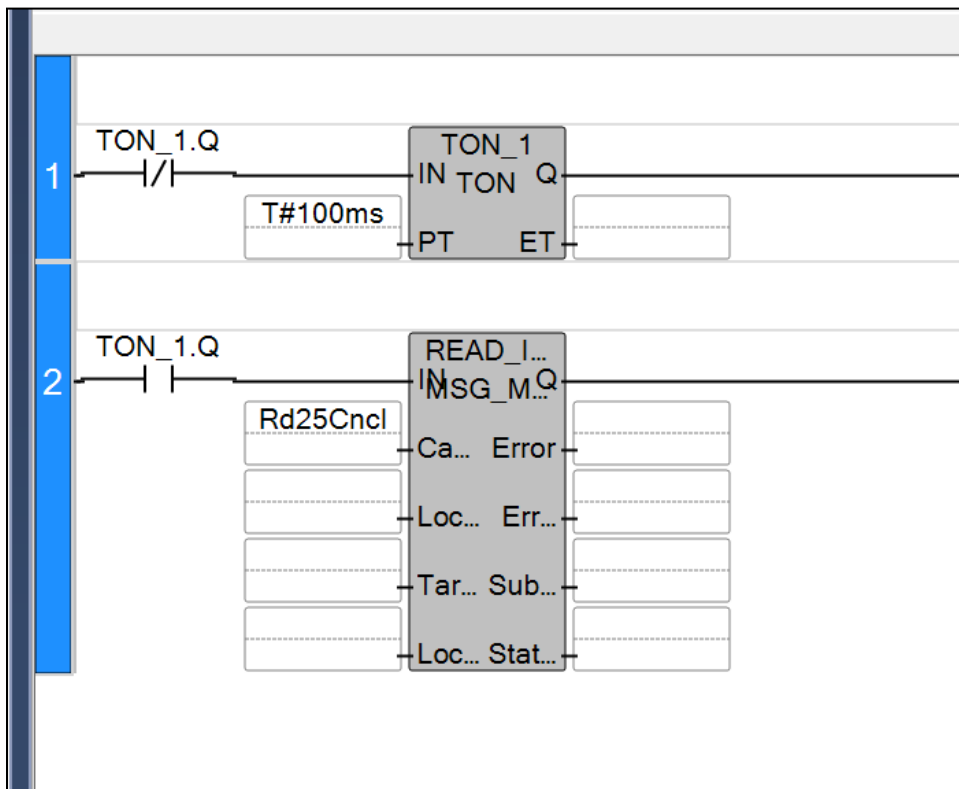
- Try a few more of the available message types. What would you need to change on the peripheral devices to use the "Holding Register" commands?
- Think about the security implications of this protocol architecture that doesn't seem to require authentication to send data to any node on the network.

### 3. Implementing Modbus TCP on the AB micro820

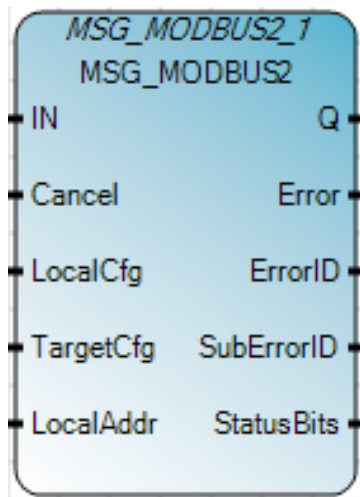
- Divide up the PLCs so that everyone is working on their own device.
- Make a new CCW project and select our m820 controller. Following the general directions from this video: <https://www.youtube.com/watch?v=pQpaucEIHeQ> turn on Modbus TCP server and configure the Modbus Mapping similar to what is shown below.

Variable Name	Data Type	Address	Addresses Used
ModbusDiscreteOutputCoils	BOOL[1..50]	00001	00001 - 00050
ModbusDiscreteInputCont...	BOOL[1..50]	10001	10001 - 10050
ModbusAnalogInput	INT[1..50]	30001	30001 - 30050
ModbusAnalogOutput	INT[1..50]	40051	30051 - 30100

- Create a new LD program and drag down a TON block to make a timer to control message rate.
- Pick one other PLC that you want to read from and create a MSG\_MODBUS block on a new rung as shown below.



- Hit (Function)-F1 to open the help on that block to read about its inputs. Also read through the pages for Local and Target Parameters.



## MODBUS2TARPARA data type

Use this table to help determine the parameter values for the MODBUS2TARPARA data type.

Parameter	Data type	Description
Addr	UDINT	Target device's Modbus data address: <ul style="list-style-type: none"> <li>1 - 65536.</li> <li>Decreases by one when sending.</li> <li>Firmware uses low-word of address if the address value is greater than 65536.</li> </ul>
NodeAddress[4]	USINT	Target device's IP address. The IP address should be a valid unicast address and cannot be 0, multicast, broadcast, local address or loop back address (127.x.x.x). For example, to specify 192.168.2.100: <ul style="list-style-type: none"> <li>NodeAddress[0]=192</li> <li>NodeAddress[1]=168</li> <li>NodeAddress[2]=2</li> <li>NodeAddress[3]=100</li> </ul>
Port	UINT	Target TCP port number. Standard Modbus/TCP port is 502. 1 - 65535 Set to 0 to use the default value 502
UnitId	USINT	Unit Identifier. Used to communicate with slave devices through a Modbus bridge. Refer Modbus specification for more details. Note that Micro800 shall not attempt to validate this value. 0 - 255 Set to 255 if Target device is not a bridge.

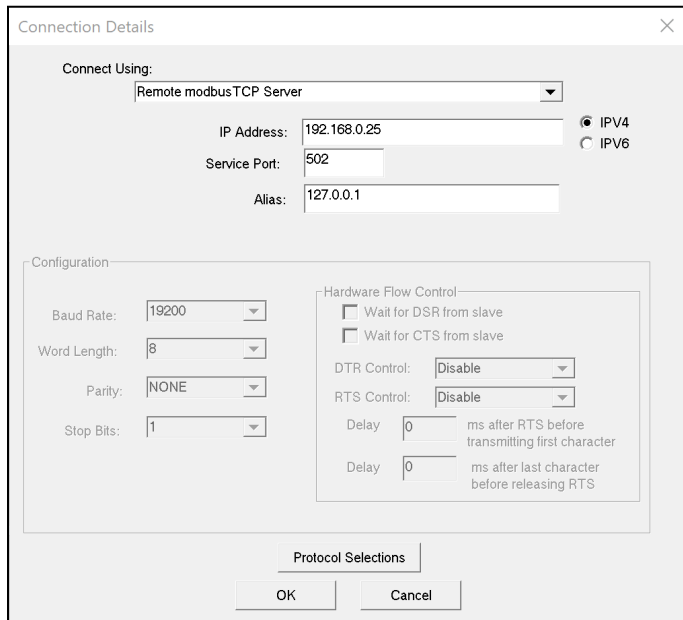
- Similar to what you did on Lab 07, create new local variables for all the inputs and map them to the appropriate data type.
- Once created, open the LCLCFG and TGTCFG and enter values as shown below. Note not shown here but you will need to expand the Target NodeAddress and enter the remote PLC's IP address as four individual integers. That is: "192", "168", "0", "XX".

Name	Alias	Data Type	Dimension	Project Value	Initial Value	Comment	Retained	String Size
> TON_1		TON		...	...		<input type="checkbox"/>	
> READ_IN_25		MSG_MOD...		...	...		<input type="checkbox"/>	
> Rd25Cncl		BOOL					<input type="checkbox"/>	
> Rd25LclPara		MODBUS2...		...	...		<input type="checkbox"/>	
> Rd25LclPara.Channel		UINT			4	Local Channel ...	<input type="checkbox"/>	
> Rd25LclPara.Trigg...		UDINT			0	0 = Trigger onc...	<input type="checkbox"/>	
> Rd25LclPara.Cmd		USINT			2	Modbus comm...	<input type="checkbox"/>	
> Rd25LclPara.Elem...		UINT			4	No. of element...	<input type="checkbox"/>	
> Rd25TgtParam		MODBUS2...		...	...		<input type="checkbox"/>	
> Rd25LclAddr		MODBUSL...		...	...		<input type="checkbox"/>	
+ New..							<input type="checkbox"/>	
> Rd25TgtParam.Addr		UDINT			1	Target's Modbu...	<input type="checkbox"/>	
> Rd25TgtParam.NodeAd...		MODBUS2NO...		...	...	Target node ad...	<input type="checkbox"/>	
> Rd25TgtParam.Port		UINT			0	Target TCP por...	<input type="checkbox"/>	
> Rd25TgtParam.UnitId		USINT			255	Unit Identifier	<input type="checkbox"/>	
> Rd25TgtParam.MsgTime...		UDINT			0	Message time ...	<input type="checkbox"/>	
> Rd25TgtParam.ConnTim...		UDINT			0	Connection tim...	<input type="checkbox"/>	
> Rd25TgtParam.ConnClose		BOOL				Connection clo...	<input type="checkbox"/>	

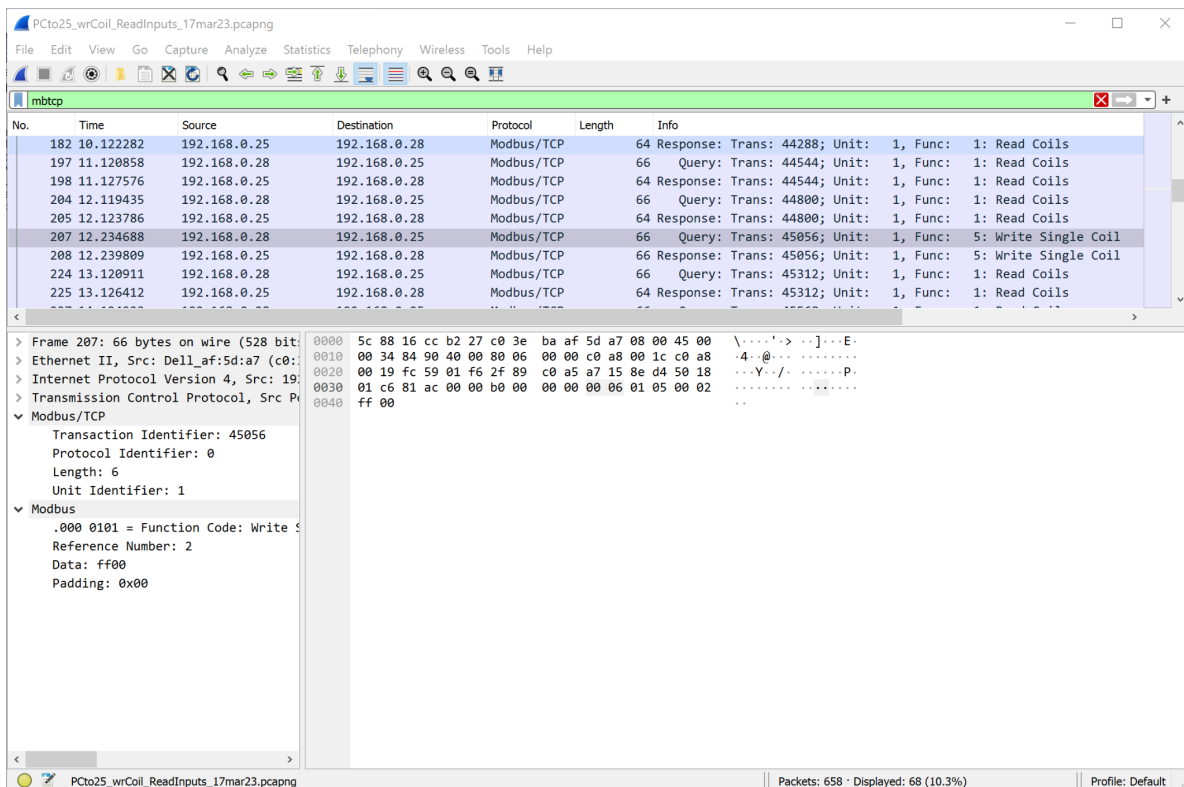
- Build and download the program to your selected PLC and make sure that your target PLC has also been updated so that they are also running the Modbus TCP server and will reply to your messages.
- Once running you should be able to expand the LclAddress variable and read in which Discrete Inputs are selected on the other PLC in address [1].
- Try to write coils or Holding Register values to the other PLC by creating more rungs and message blocks.

#### 4. Data collection for Modbus TCP messages

- Download Wireshark if you don't already have it on your machine.
- Plug an ethernet cable into your laptop within the classroom PLC environment.
- Verify which ethernet interface on your computer is in the 192.168.0.XX domain.
- Open up ModScan64.exe again, make a new connection and this time select "Remote modbus TCP server." And enter the IP address of the PLC that you want to connect to.






- Start a wireshark capture using the interface you identified previously then go back to the MadScan64 program and try to read or write values to the coils, input, or registers of any of the PLCs. Try a couple different function codes and values.
- Stop the Wireshark capture (and save as a pcapng file for later use) then work to filter the data using the "mbtcp" filter at the top. Look at the contents of the modbus messages, does that make sense for what you were doing with the ModScan64 interactions?





5. Give CIP messaging a go.

- We likely won't have time in class to do this, but if you are so inclined, check out the following videos and try to implement CIP messaging from one PLC to another by creating an address that holds a float value and reading (or writing) that value from another PLC. Remember that the data format that CIP uses to send that data will require you to do some manipulation (MOV on both ends) of the sent and received data to conform with CIP requirements.
-  Micro800 Message MSG to Read a Controllogix or Compactlogix  
<https://www.youtube.com/watch?v=wMNME11mwPs> 17:12
-  Communication Setup CIP, COP for Micro800 (part 1)  
<https://www.youtube.com/watch?v=a1cJdN2b8kQ> 1:18
-  Communication Setup CIP, COP for Micro800 (part 2)  
<https://www.youtube.com/watch?v=xJT8m8BREOw> 12:14