PMEUCM TCPOPEN

Installation Manual

This manual covers the PMEUCM TCPOPEN custom application and installation procedures.

Effective: January 18, 2022

Niobrara Research & Development Corporation
P.O. Box 3418 Joplin, MO 64803 USA

Telephone: (800) 235-6723 or (417) 624-8918
Facsimile: (417) 624-8920
http://www.niobrara.com

Contents

1 Introduction

The Niobrara PMEUCM is a user programmable communication card for the Schneider Electric x80 PAC platform. It is capable of running a custom application for performing communication translations between serial and/or Ethernet protocols for the Modicon M580 Automation platform.

The PMEUCM is ideal for interfacing 3rd party Ethernet devices that do not communicate using a 'Standard' industrial Ethernet protocol like Modbus/TCP or Ethernet/IP. For example, consider an inkjet printer that uses XML messages across an Ethernet TCP socket. The M580 CPU is capable of generating the XML strings required to print, but is not capable of opening an Ethernet client connection to the printer to send the message to the printer.

The 'standard' method for using the PMEUCM in this type of application would be to write a custom UCM program to interface with the printer and simply pass variables to the M580 across the backplane. The UCM would control the operation of the socket connections, generate the XML messages to send to the printer, and parse the XML replies from the printer. A custom DTM would be built to define the variable structures used in the transfer of data to/from the UCM from the M580. The M580 PLC program would simply set values in variables sent to the UCM to define things like the target IP Address, string to print, connect to the printer, and start/stop the print. The UCM would send data to the M580 such as connection status, ink status, current printed value.

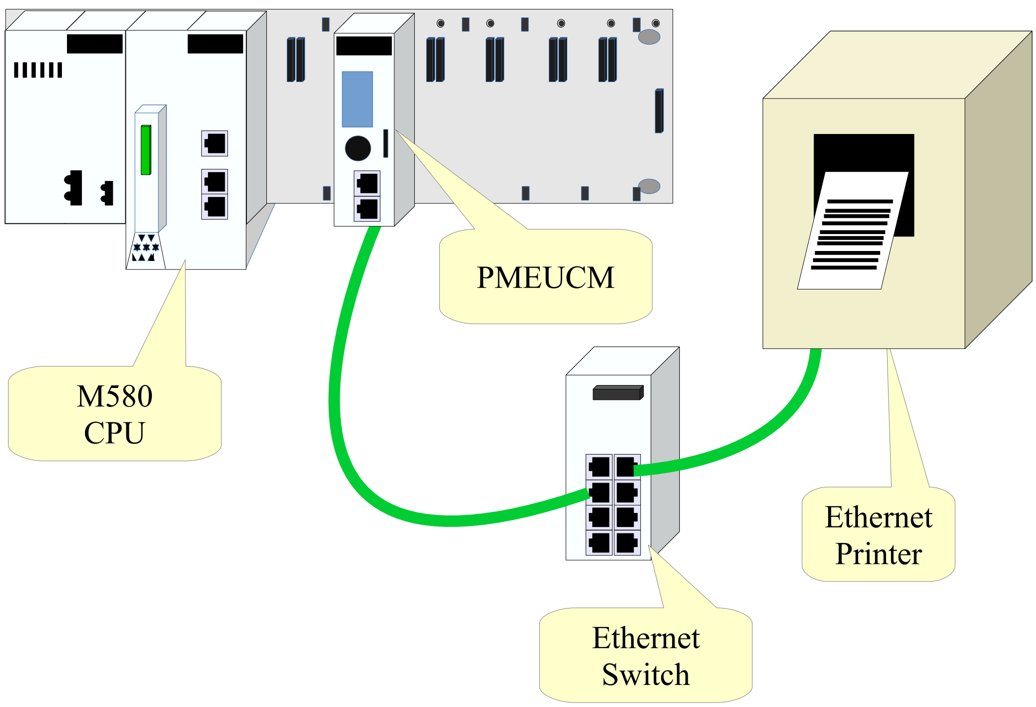
This 'standard' method of building a custom application for the PMEUCM provides the highest system performance and optimum troubleshooting features, but requires the end user to learn a new programming language and support multiple versions of code for a given PLC system (or hire Niobrara to write the application).

Niobrara recognizes that many customers would prefer to do 'all' of the coding in the M580 Unity environment. This is especially true for customers migrating from the Premium PLC as it had a TCP Open library for select versions of the ETY Ethernet modules.

The Premium TCP Open library involved special libraries of Elementary Function Blocks (EFBs) and Derived Function Blocks (DFBs) that allowed direct control of TCP/IP socket connections and data transmission.

Niobrara has developed an application for the PMEUCM that mimics the Premium TCP Open system by passing direct control of client and server socket operation to the M580 PLC program. A specific DTM and a pair of DFBs have been developed to provide simple variables to be used in the M580 for socket control.

Figure 1.1: Typical Application



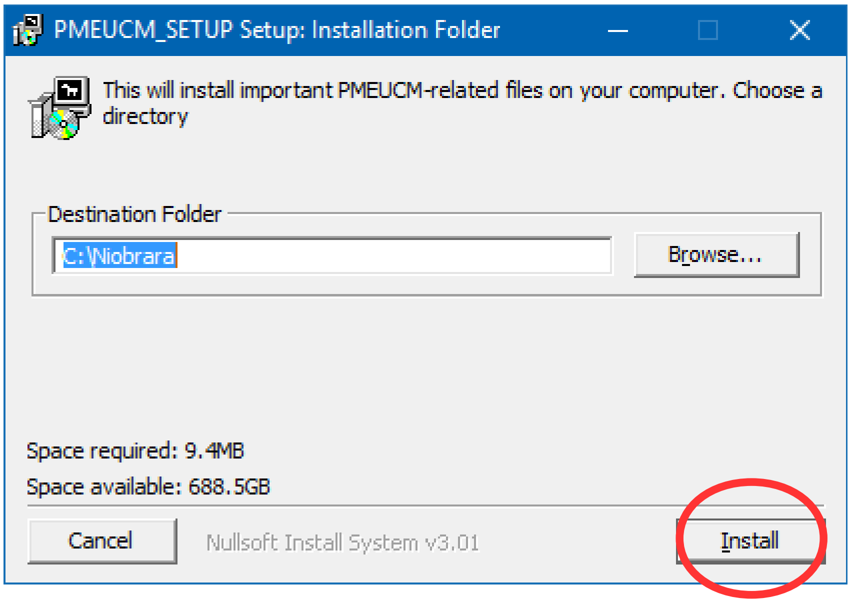
If the example in Figure 1.1 where done using the TCPOpen UCM application, the M580 program would generate the XML messages, direct the UCM to open a client TCP connection to the printer, and the M580 would send and receive all messages to and from the printer.

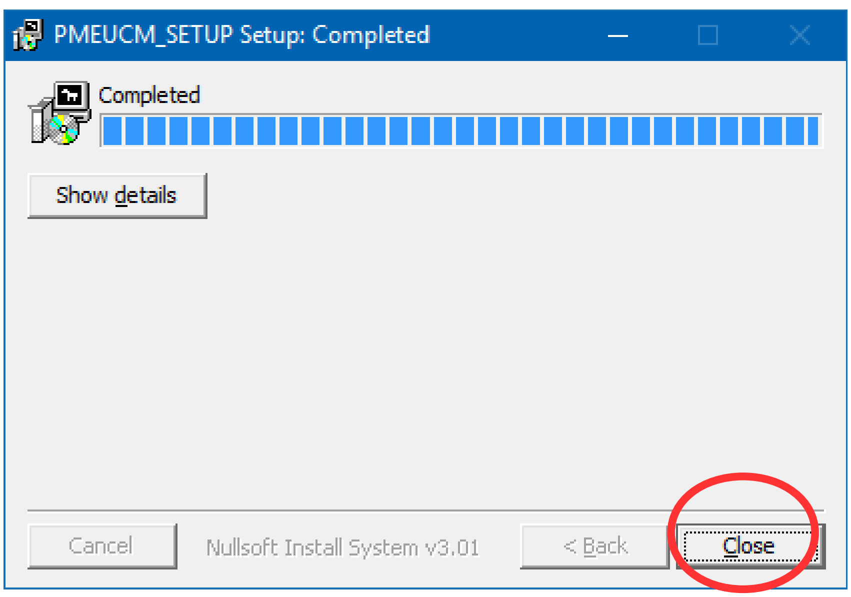
2 PMEUCM_SETUP.EXE

The latest version of Niobrara's PMEUCM_SETUP must be installed before attempting to use the PMEUCM. This setup installs many utilities needed to configure the PMEUCM. The user may access this file at:

http://www.niobrara.com/html/pmeucm_cut.html

Download and run PMEUCM_SETUP.EXE. A box will appear prompting the user to choose a directory in which to install. The default is C:\Niobrara, as shown below.

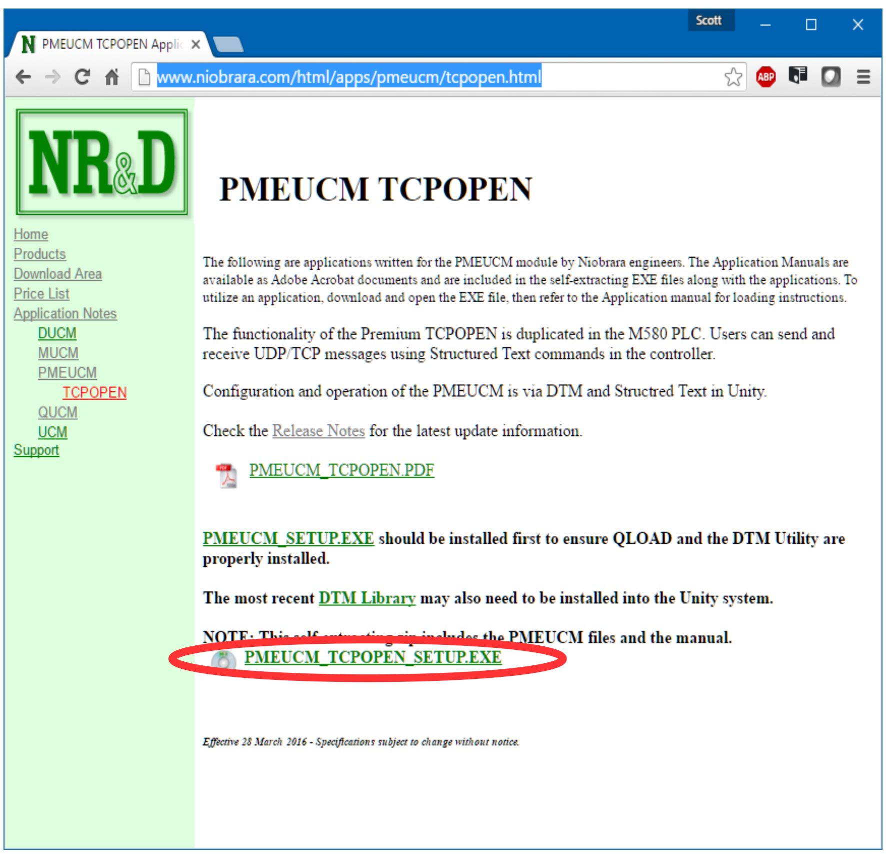




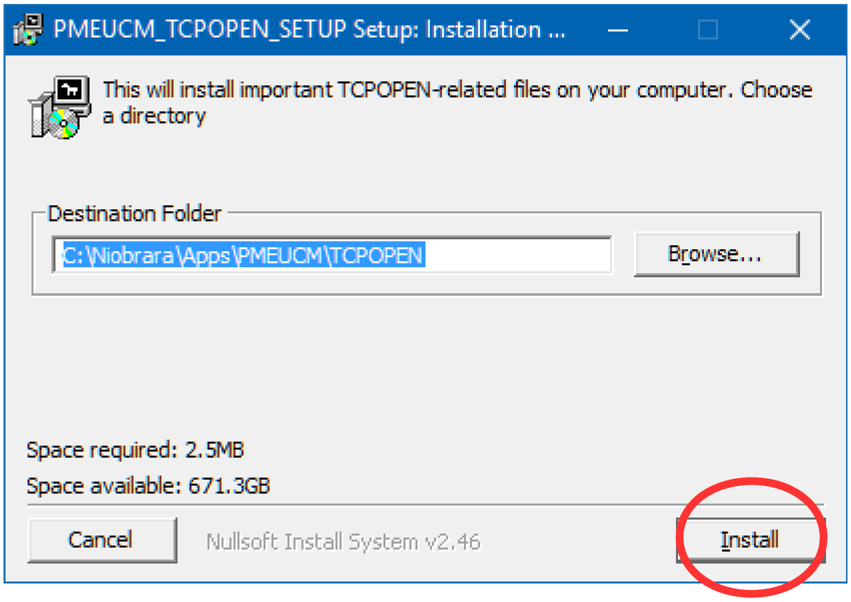
3 PMEUCM_TCPOPEN_SETUP.EXE

The latest version of Niobrara's PMEUCM_TCPOPEN_SETUP must be installed. This setup installs many files needed to configure the PMEUCM for the TCPOPEN application. The user may access this file at:

http://www.niobrara.com/html/apps/pmeucm/tcpopen.html



Download and run PMEUCM_TCPOPEN_SETUP.EXE. A box will appear prompting the user to choose a directory in which to install. The default is C:\Niobrara\Apps\PMEUCM\TCPOPEN, as shown below.



4 NRD DTM UTILITY

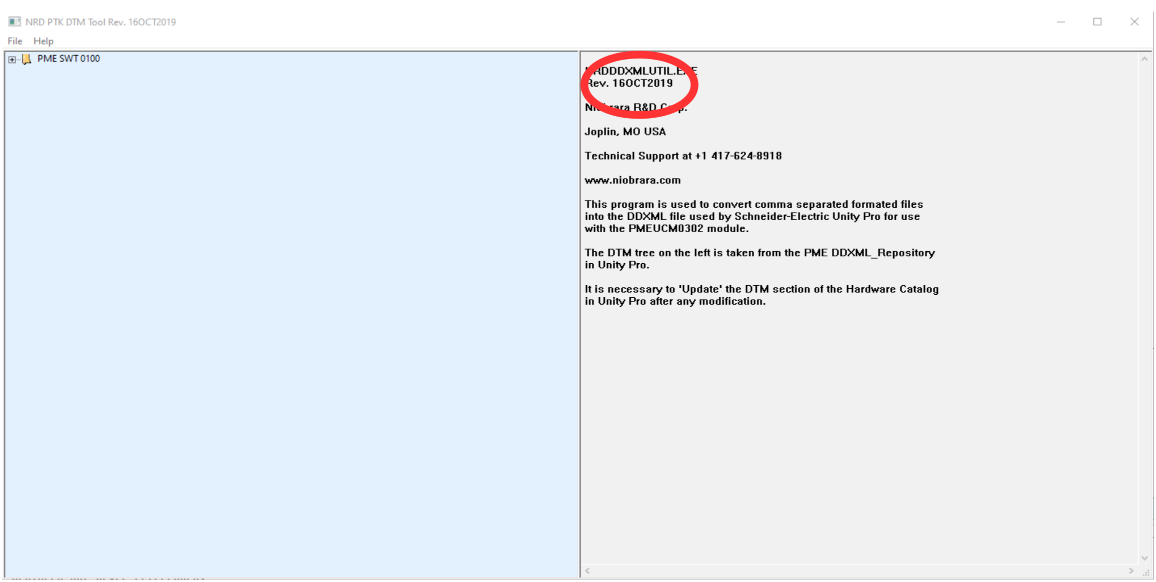
The NRD DTM Utility is installed by the PMEUCM_SETUP.EXE program. The user may access this file at:
http://www.niobrara.com/programs/PMEUCM_SETUP.EXE,

Open the NRD DTM Utility

The next step is to open the Niobrara DTM Tool. Select Programs > Niobrara > PMEUCM > DTM > DTM Utility.

NOTE: User permissions in Windows may cause the following steps to not actually take effect. A better practice each time may be to right-click on the icon for the DTM Utility, and choose "Run as Administrator."

The DTM Tool must be at least revision 16OCT2019 to operate properly with the PMEUCM0302.



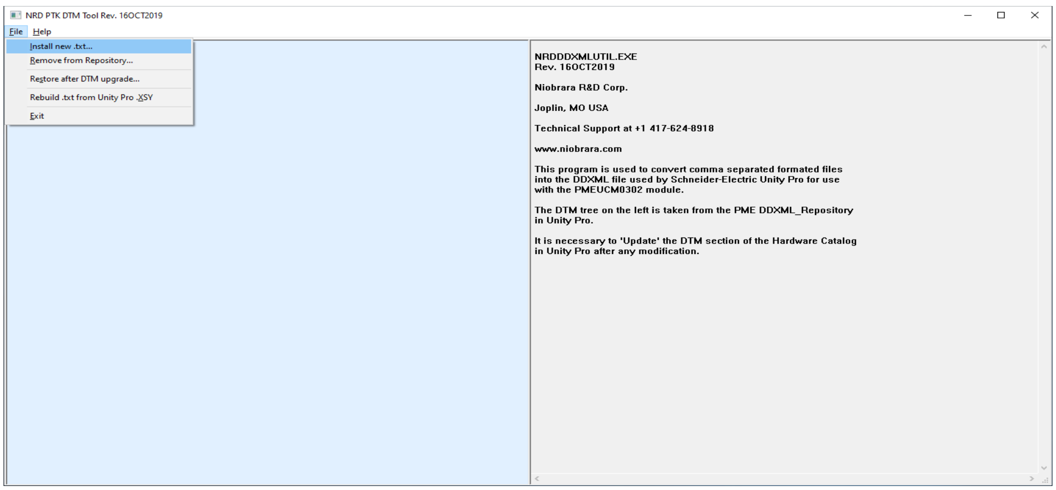
The tree on the left of the screen shows the PTK DTMs installed in Unity Pro. In this case, it shows the PME SWT 0100 Weighing Module from SCAIME.

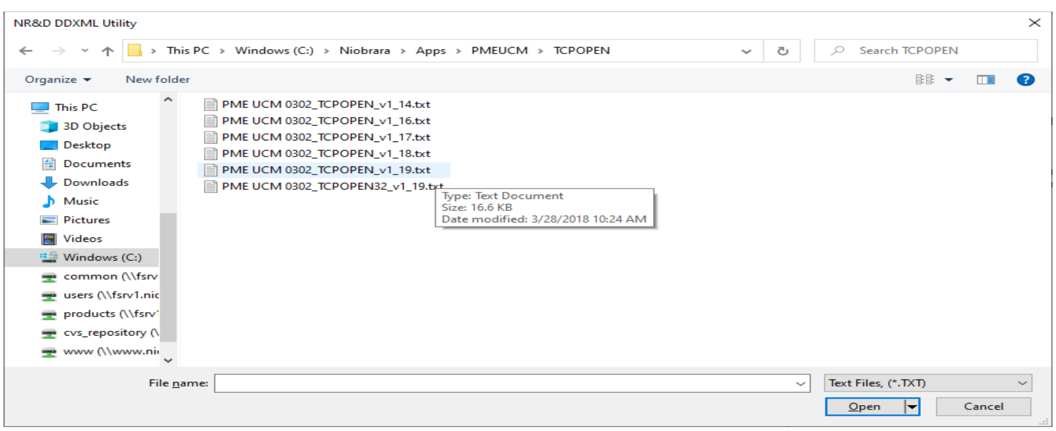
Installing a new file

Select File > “Install new .txt...” and then browse to the location where the txt file is located and select the file to install:

“c:\Niobrara\apps\PMEUCM\TCPOPEN\PME UCM 0302_TCPOPEN_v1_19.txt”

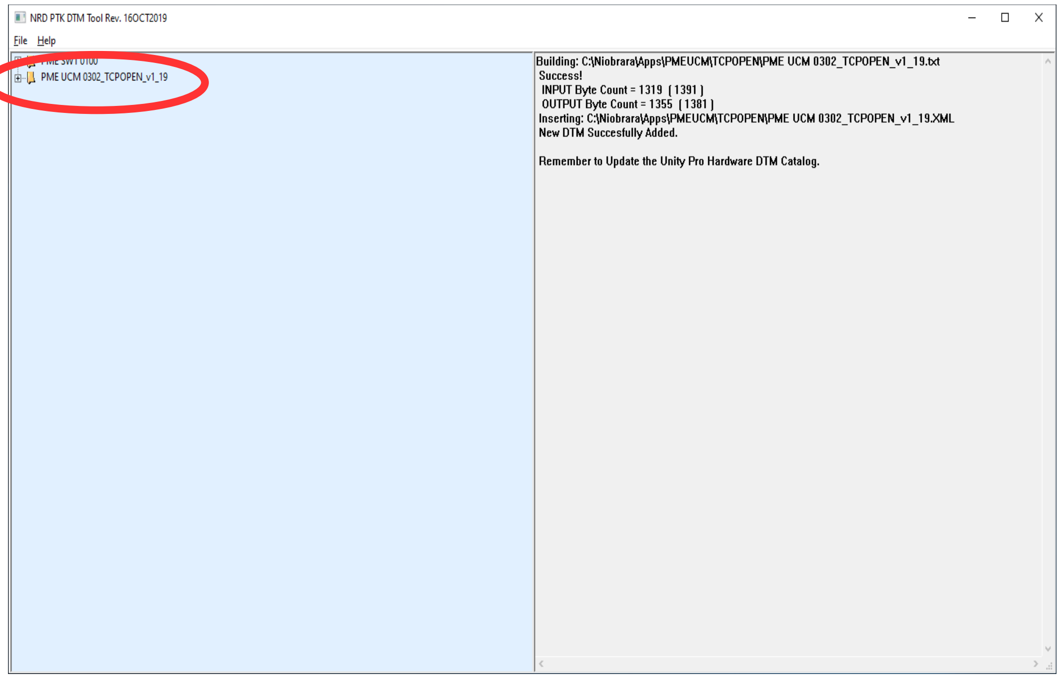
NOTE: The file version number in this example is 1.19 but may be different for a newer version of the setup file. If multiple versions of the .txt file are present, it is normally advised to choose the highest version numbered file.



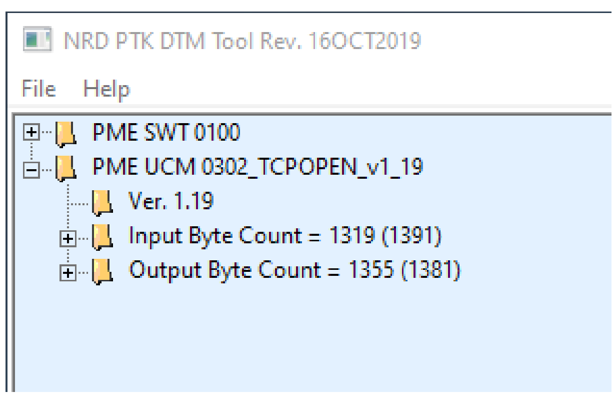


After selecting “Open”, the main screen should now change to show a new entry in the tree.

Status information is displayed on the right side of the screen. If there is an error during the compile, the error description and source code line number will be displayed.



Clicking on the + will expand the TCPOPEN structure to show an overview of the PLC Inputs and Outputs.

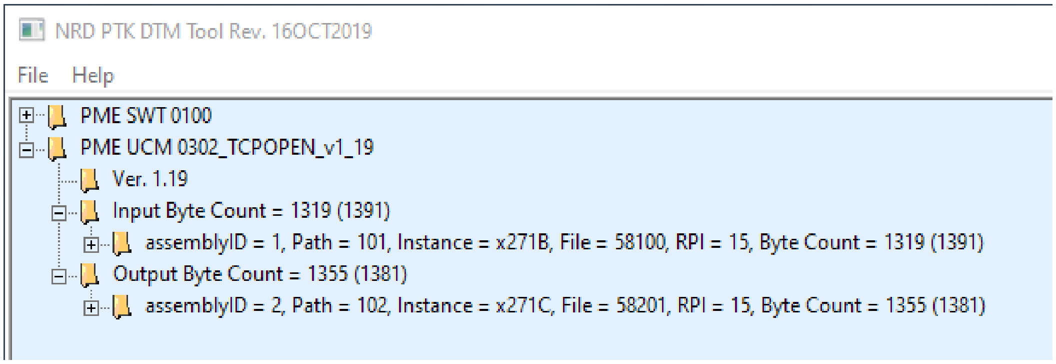


The version number here is shown to be 01.19. This version number must agree with the TCPOPEN application running in the PMEUCM.

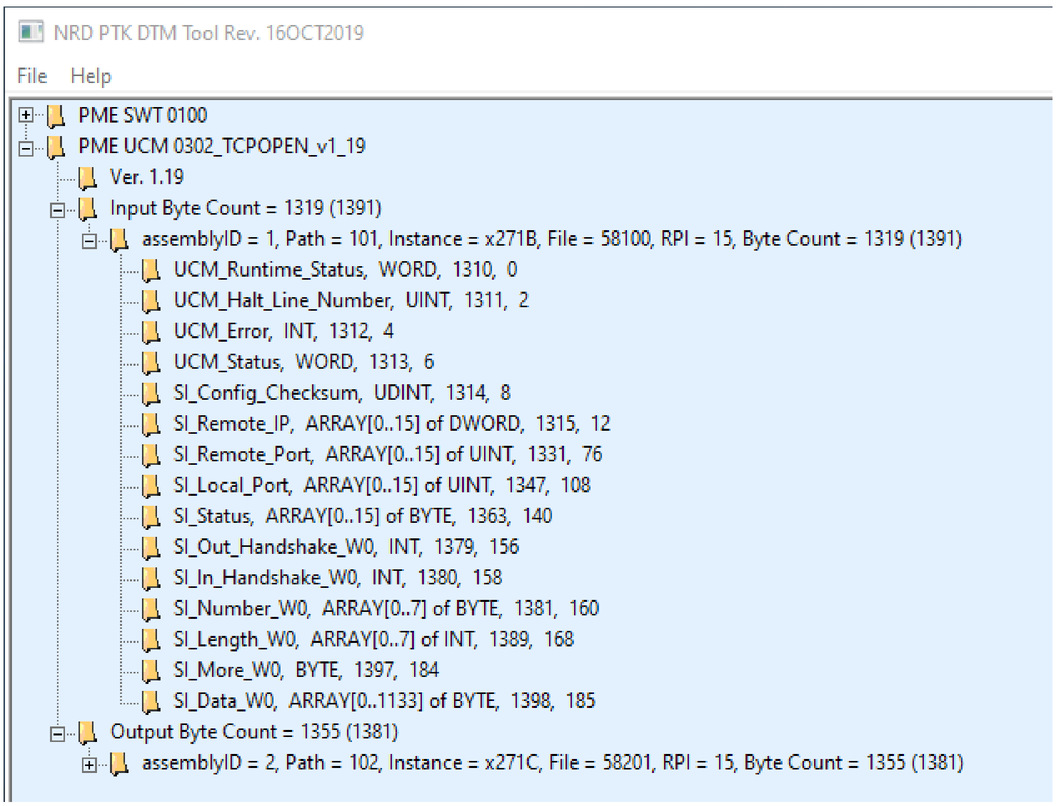
Expanding the tree further reveals the Ethernet/IP Assemblies configured for this application.

The total byte counts for PLC Inputs and Outputs are 1391 and 1381 bytes respectively.

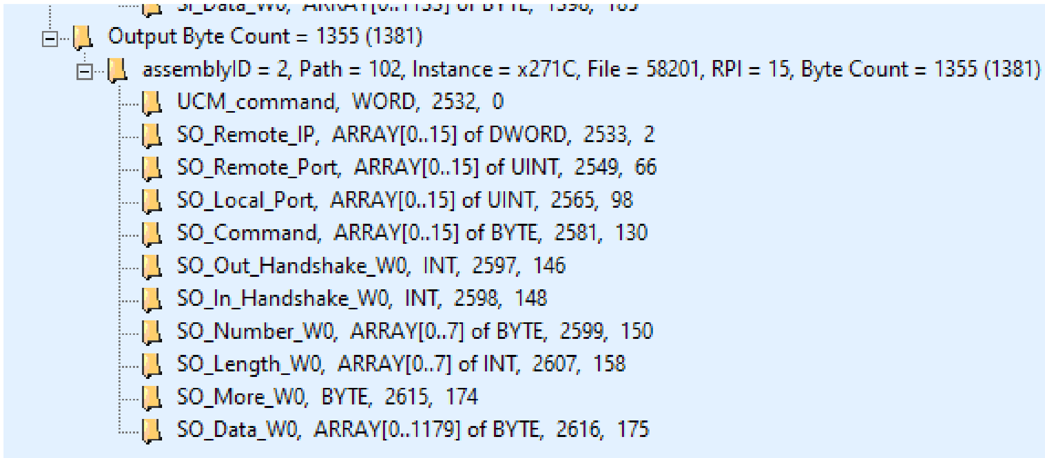
The Ethernet/IP RPI is defaulted to 15mS updates.



Expanding the Inputs reveals the variable passed from the UCM to the M580 CPU.



Expanding the Outputs shows the variables from the CPU to the UCM.


Output Byte Count = 1355 (1381)
assemblyID = 2, Path = 102, Instance = x271C, File = 58201, RPI = 15, Byte Count = 1355 (1381)
UCM_command, WORD, 2532, 0
SO_Remote_IP, ARRAY[0..15] of DWORD, 2533, 2
SO_Remote_Port, ARRAY[0..15] of UINT, 2549, 66
SO_Local_Port, ARRAY[0..15] of UINT, 2565, 98
SO_Command, ARRAY[0..15] of BYTE, 2581, 130
SO_Out_Handshake_W0, INT, 2597, 146
SO_In_Handshake_W0, INT, 2598, 148
SO_Number_W0, ARRAY[0..7] of BYTE, 2599, 150
SO_Length_W0, ARRAY[0..7] of INT, 2607, 158
SO_More_W0, BYTE, 2615, 174
SO_Data_W0, ARRAY[0..1179] of BYTE, 2616, 175

5 System Operation

The DTM for the TCPOPEN application defines the variables used for transferring data between the M580 CPU and the PMEUCM. Some of the variables are passed on every transaction between the CPU and the UCM (Remote IP Address arrays, commands, status, etc.) The actual data that needs to be sent or received on a particular socket is too large to be sent for all 16 sockets on each transaction. A window scheme is used for this data and DFBs are provided to automatically transfer the socket data to/from the UCM through the data window. The DFBs use a set of global variables to provide a 1452 byte array for each of the 16 sockets for both inbound and outbound messages.

Full Frame Data Arrays

The inbound data from the UCM for each socket is placed in a 16 element array of UCM_Full_Frame. The actual byte count for each array of bytes is placed in a 16 element array of INT. The Inbound data is placed in UCM_Data_IN with the length for each socket in UCM_Data_IN_Length. The outbound data is placed in UCM1_Data_OUT for each of the 16 sockets and the corresponding length is placed in the array UCM1_OUT_Length.

UCM1_Data_IN	ARRAY[0..15] OF UCM_Full_Frame
UCM1_Data_IN_Length	ARRAY[0..15] OF INT
UCM1_Data_OUT	ARRAY[0..15] OF UCM_Full_Frame
UCM1_Data_OUT_Length	ARRAY[0..15] OF INT

UCM_Full_Frame is a DDT array of 1452 bytes.

Name	Type
UCM_Full_Frame	ARRAY[0..1451] OF BYTE

The length field is used as the handshake to let both sides know that new data has arrived or needs to be sent.

Outbound Operations

In this example, assume that socket 3 is already connected to a remote server (or client).

The M580 needs to send a message on socket 3 of 110 bytes. The following sequence would be followed:

1. The PLC code would wait until UCM1_Data_OUT_Length[3] = 0. If this length is non-zero then the last transmitted frame is not finished being moved to the UCM. Wait for this value to become zero before loading new data into UCM1_Data_OUT[3].
2. Copy the 110 bytes that need to be transmitted into

   UCM1_Data_OUT[3][0] through UCM1_Data_OUT[3][109].
3. Set the new length value UCM1_Data_OUT_Length[3] := 110.
4. The DFB TCPOPEN_Outbound watches for UCM1_Data_OUT_Length to be > 0 and will then transfer the new data to the UCM across the backplane. When the data is completely moved to the UCM, the DTM will zero UCM1_Data_OUT_Length[3] so the process may start again.

Inbound Operations

As in the above example, Socket 3 is already connected and 55 bytes will be received by the UCM from the remote end of the connection.

1. The PLC code will wait until UCM1_Data_IN_Length[3] > 0. The DFB TCPOPEN_Inbound will set the value to be greater than zero when the complete frame data is ready to be used by the PLC program. In this case, the length value will be set to 55 bytes.
2. The PLC code will then parse the incoming message as contained in UCM1_Data_IN[3][0] through UCM1_Data_IN[3][54].
3. When the PLC code is finished pulling the data out of UCM1_Data_IN[3] then it will set UCM1_Data_IN_Length[3] := 0.

NOTE: Inbound and Outbound length values must be within the range of 1-1452.

TCPOPEN DTM Variables

The TCPOPEN DTM provides the variables used in the Unity program to monitor and control the operation of the TCPOPEN PMEUCM application.

NOTE: Many of these variables are used by the Inbound and Outbound DFBs and should not be altered elsewhere in the PLC application.

NOTE: All variable names will be preceded by the name of the DTM added for the specified PMEUCM. In this example, that will be “UCM1.”

Freshness

Name	Value	Type	Comment
UCM1		T_UCM1	
Freshness	1	BOOL	Global Freshness
Freshness_1	1	BOOL	Freshness of Object
Inputs		T_UCM1_IN	Input Variables
Outputs		T_UCM1_OUT	Output Variables

A read-only BOOL variable named 'Freshness' is provided to report the status of the data connection between the DIO Master (M580 CPU, usually) and the PMEUCM across the Ethernet backplane. Freshness becomes TRUE when all of the Ethernet/IP connections between the M580 and the PMEUCM are active and passing data.

It is good policy to use the state of 'Freshness' to control the logic relating to the TCPOPEN application in the M580.

```
if PME_UCM_0302_TCPOPEN.Freshness then
        (* process the UCM data *)
end_if;
```

There will be at least one additional 'Freshness_1' BOOL variable present which will provide the status of each Ethernet/IP connection to the UCM. The standard PME UCM 0302_TCPOPEN DTM only uses a single Ethernet/IP connection so it will only show Freshness_1.

The 'Freshness' variable will only be TRUE if the 'Freshness_1' variable is also true.

UCM1	T_UCM1		
Freshness	BOOL		Global Freshne...
Freshness_1	BOOL		Freshness of O...
Inputs	T_UCM1_IN		Input Variables
Outputs	T_UCM1_OUT		Output Variables

Inputs

The PLC Input data is read-only and includes a block of 'overhead' data required by the PME Generic DTM provided by Schneider Electric. This data is normally ignored in the M580 project.

NOTE: The Device_Name does give the current version of the running TCPOPEN application '27APR201'.

Name	Value	Type	Comment
UCM1		T_UCM1	
UCM1.Freshness	1	BOOL	Global Freshness
UCM1.Freshness_1	1	BOOL	Freshness of Object
UCM1.Inputs		T_UCM1_IN	Input Variables
UCM1.Inputs.DEVICE_NAME	'TCPOpen 27APR201'	string[64]	
UCM1.Inputs.DEVICE_STATUS	3	UINT	
UCM1.Inputs.DEVICE_STATE	1	BOOL	
UCM1.Inputs.DEVICE_HEALTH	1	BOOL	
UCM1.Inputs.SPI_ERROR	0	BOOL	
UCM1.Inputs.INIT_PARAM_CORRUPT...	0	BOOL	
UCM1.Inputs.RUNTIME_CONFIGURA...	0	BOOL	
UCM1.Inputs.CONTROL_FW_MISSIN...	0	BOOL	
UCM1.Inputs.CONTROL_FW_DOWNL...	0	BOOL	
UCM1.Inputs.Free0	0	BYTE	Unused Variable
UCM1.Inputs.Free1	0	BYTE	Unused Variable
UCM1.Inputs.ETH_STATUS	129	BYTE	
UCM1.Inputs.PORT1_LINK	1	BOOL	
UCM1.Inputs.CCOTF_IN_PROGRESS	0	BOOL	
UCM1.Inputs.REDUNDANCY_OWNER	0	BOOL	
UCM1.Inputs.GLOBAL_STATUS	1	BOOL	
UCM1.Inputs.SERVICE_STATUS	16	BYTE	
UCM1.Inputs.SNTP_SERVICE	0	BOOL	
UCM1.Inputs.SNMP_SERVICE	0	BOOL	
UCM1.Inputs.FDR_SERVICE_B1	1	BOOL	
UCM1.Inputs.FDR_SERVICE_B2	0	BOOL	
UCM1.Inputs.FDR_SERVICE_B3	0	BOOL	
UCM1.Inputs.FDR_SERVICE_B4	0	BOOL	
UCM1.Inputs.ETH_PORT1_INFO	1	BYTE	
UCM1.Inputs.ETH_PORT1_FUNCTIO...	1	BOOL	
UCM1.Inputs.ETH_PORT1_FUNCTIO...	0	BOOL	
UCM1.Inputs.Free2	0	BYTE	Unused Variable
UCM1.Inputs.UCM_Runtime_Status	49152	WORD	.15=Run; LSB=Runtime Err
UCM1.Inputs.UCM_Halt_Line_Number	0	UINT	UCM Runtime Error Halt Lin
UCM1.Inputs.UCM_Error	0	INT	0=No Error; 1=Wrong DTM
UCM1.Inputs.UCM_Status	16	WORD	.0=E1 LinkOK; .1=Dup IP E
UCM1.Inputs.SI_Config_Checksum	1311178907	UDINT	CRC32 of DTM configuratio
UCM1.Inputs.SI_Remote_IP		ARRAY[0..15] O...	Socket IP Address
UCM1.Inputs.SI_Remote_Port		ARRAY[0..15] O...	TCP or UDP Remote Port N
UCM1.Inputs.SI_Local_Port		ARRAY[0..15] O...	TCP or UDP Local Port Nur
UCM1.Inputs.SI_Status		ARRAY[0..15] O...	.0=Connect; .1=1/2 closed;
UCM1.Inputs.SI_Out_Handshake_W0	0	INT	Outbound handshake
UCM1.Inputs.SI_In_Handshake_W0	0	INT	Inbound handshake
UCM1.Inputs.SI_Number_W0		ARRAY[0..7] OF...	Socket Number for Data (0-

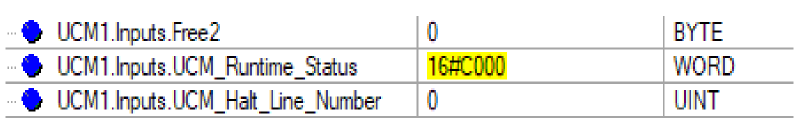
The actual UCM provided data starts with UCM_Runtime_Status.

UCM_Runtime_Status

WORD – The UCM_Runtime_Status provides an indication that the TCPOPEN application is running properly in the UCM. The bits of this word may be monitored. This value is best viewed in hexadecimal.

Bits	Meaning	Notes
Bit 15	1=Running 0=Halted	Normally bit 15 is ON.
Bit 14	1=Module Configured and Running Normal 0=Module not fully configured	Normally bit 14 is ON.
0-13	Last Halt Error condition	See table below

Code	Description
C0xx	Application Running, if xx nonzero, xx=last halting error (in hex)
0	Terminated by clearing all thread run bits
1	STOP statement executed
2	Illegal instruction exception
3	Division by Zero
4	Out of heap space for ON CHANGE
5	Out of heap space for ON RECEIVE
6	Unsupported run-time call, likely compiler/firmware mismatch
7	Parameter or array index out of range
8	Downloaded code corrupt, CRC Error
9	CPU Address exception
10	Stack Underflow
11	TCP Error -1, likely compiler/firmware mismatch
12	TCP Error -2, contact Niobrara
13	TCP Error -3, not enough sockets or buffers, See register 66. Also IP address or gateway not initialized
14	Hardware not authorized to run user code



In the above screenshot, the UCM_Runtime_Status = 16#C000 which is the normal value.

UCM_Halt_Line_Number

UINT – The Halt Line Number value provides the source code line number where the most recent runtime halting error has occurred. This value should be zero during normal operation. Contact Niobrara Technical Support if this value is non-zero.

UCM_Error

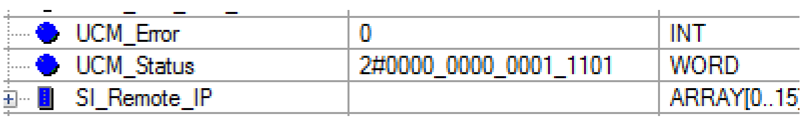
INT – The UCM_Error value provides an indication that the TCPOPEN application has an issue with configuration.

Value	Meaning	Notes
0	No Configuration Errors	
1	PLC Modbus/TCP Connection Error	M580 May have TCP Port 502 Security Enabled.
2,3	Reserved	
4	Bad DTM Filename	
4	Bad DLL Version	
5	Bad DTM File Version	
8	Bad Assembly Size	
9	Reserved	
10	Bad E1 IP Address	Reverts to 10.10.10.10
11	Bad E1 Subnet Mask	Reverts to 255.0.0.0
12	Bad E1 Default Gateway	Reverts to 0.0.0.0
13	Bad E2 IP Address	Reverts to 10.10.10.11
14	Bad E2 Subnet Mask	Reverts to 255.0.0.0
15	Bad E2 Default Gateway	Reverts to 0.0.0.0
16,17	Reserved	
18	PLC in STOP	
19	Reserved	
20	No Link on UCM Backplane	
21	Duplicate IP on BP	
22-26	Reserved	
27	Watchdog Expired	
28	Bad DIO Ch Count	
29	Bad DIO Output Count	
30	Bad DIO Input Count	
31	HSBY FDR Do NOT Match	

UCM_Status

WORD – The UCM_Status provides a bit-mapped indication of conditions in the module. Presently, the first 6 bits provide indication of the three Ethernet ports.

Bit	Meaning	Notes
0 (lsb)	1 = E1 Link OK 0 = E1 Link off	
1	1 = E1 duplicate IP Address 0 = E1 not in duplicate IP	LCD displays offending MAC
2	1 = E2 Link OK 0 = E2 Link off	
3	1 = E2 duplicate IP Address 0 = E2 not in duplicate IP	LCD displays offending MAC
4	1 = BP Link OK 0 = BP Link Off	
5	1 = BP duplicate IP Address 0 = BP not in duplicate IP	LCD displays offending MAC
6-15	Reserved	



UCM_Error	0	INT
UCM_Status	2#0000_0000_0001_1101	WORD
SI_Remote_IP		ARRAY[0..15

In this screenshot, the following bits are TRUE:

- Bit 0 – E1 Link is ON
- Bit 2 – E2 Link is ON
- Bit 3 – E2 is in Duplicate IP Address
- Bit 4 – Backplane Link is ON

SI_Config_Checksum

UDINT – This 32-bit variable shows the checksum of the DTM configuration as read by the PMEUCM from the FDR server. The user may use this value to verify that the DTM has been configured as intended.

SI_Remote_IP

ARRAY[0..15] of DWORD – These 16 variables show the IP Address of the remote end of a connection for each socket compressed into a DWORD. Each 8 bit BYTE of the DWORD is the octet of the IP Address.

For example, if the remote IP Address for socket 1 is 192.168.0.111 then

SI_Remote_IP[1] = 3232235631(dec) = 16#C0A8_006F(hex)

Looking at the value in hexadecimal reveals the IP Address:

C0 = 192

A8 = 168

00 = 0

6F = 111

A simple method for moving this data to BYTE variables is to use the 'Shift Right' (SHR) ST command. For example, the program includes an array of bytes RemIP[0..3]. The following code would move the Remote IP address of socket 3 into this array.

```
RemIP[0] := SHR(UCM1.Inputs.SI_Remote_IP[3],24);

RemIP[1] := SHR(UCM1.Inputs.SI_Remote_IP[3],16);

RemIP[2] := SHR(UCM1.Inputs.SI_Remote_IP[3],8);

RemIP[3] := SHR(UCM1.Inputs.SI_Remote_IP[3],0);
```

The first SRH shifts the DWORD 24 bits and loads the BYTE variable. The next lines shift 16 bits and 8 bits. The last line doesn't shift at all and could just be a straight assignment.

UCM_Status	2#0000_0000_0001_0101	WORD
SI_Remote_IP		ARRAY[0..15] OF DWORD
SI_Remote_IP[0]	0	DWORD
SI_Remote_IP[1]	0	DWORD
SI_Remote_IP[2]	2886729928	DWORD
SI_Remote_IP[3]	0	DWORD
SI_Remote_IP[4]	2886729739	DWORD
SI_Remote_IP[5]	0	DWORD
SI_Remote_IP[6]	0	DWORD
SI_Remote_IP[7]	0	DWORD
SI_Remote_IP[8]	2886729738	DWORD
SI_Remote_IP[9]	0	DWORD
SI_Remote_IP[10]	0	DWORD
SI_Remote_IP[11]	0	DWORD
SI_Remote_IP[12]	0	DWORD
SI_Remote_IP[13]	0	DWORD
SI_Remote_IP[14]	0	DWORD
SI_Remote_IP[15]	0	DWORD
SI_Remote_Port		ARRAY[0..15] OF UINT

The above screenshot shows three active socket connections on sockets 2, 4, and 8.

- SI_Remote_IP[2] = 2886729928 (dec) = AC1000C8 (hex)
- SI_Remote_IP[4] = 2886729739 (dec) = AC10000B (hex)
- SI_Remote_IP[8] = 2886729838 (dec) = AC10000A (hex)

Splitting this data by bytes shows the remote IP Address: 172.16.0.200 for socket 2.

- AC (hex) = 172 (decimal)
- 10 (hex) = 16 (decimal)

- 00 (hex) = 0 (decimal)
- C8 (hex) = 200 (decimal)

Socket 4 has a remote IP Address of 172.16.0.11.

- AC (hex) = 172 (decimal)
- 10 (hex) = 16 (decimal)
- 00 (hex) = 0 (decimal)
- 0B (hex) = 11 (decimal)

Socket 8 has a remote IP Address of 172.16.0.10.

- AC (hex) = 172 (decimal)
- 10 (hex) = 16 (decimal)
- 00 (hex) = 0 (decimal)
- 0A (hex) = 10 (decimal)

SI_Remote_Port

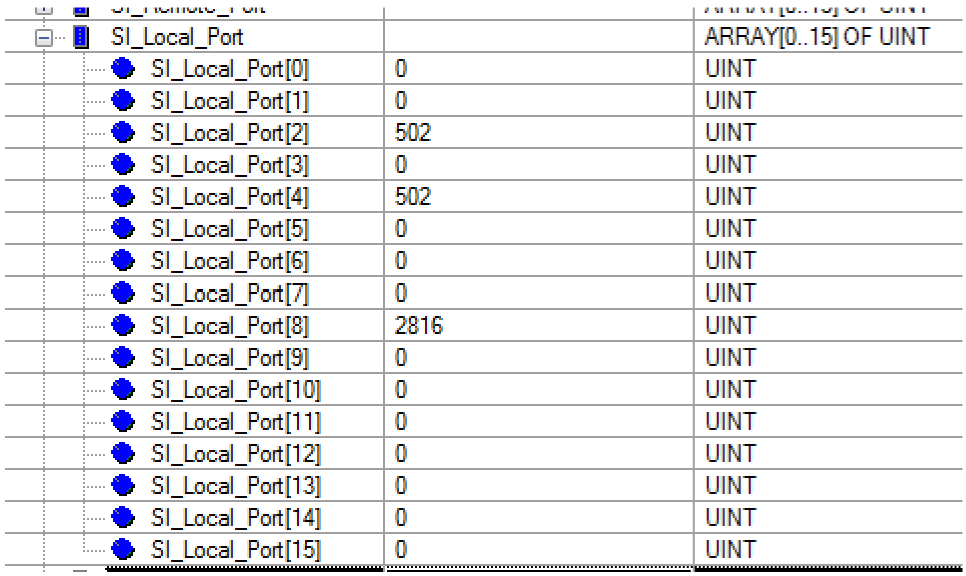
ARRAY[0..15] of UINT – These 16 variables show the TCP (or UDP) port number of the remote end of a connection for each socket.

SI_Remote_Port		ARRAY[0..15] OF UINT
SI_Remote_Port[0]	0	UINT
SI_Remote_Port[1]	0	UINT
SI_Remote_Port[2]	61633	UINT
SI_Remote_Port[3]	0	UINT
SI_Remote_Port[4]	2816	UINT
SI_Remote_Port[5]	0	UINT
SI_Remote_Port[6]	0	UINT
SI_Remote_Port[7]	0	UINT
SI_Remote_Port[8]	502	UINT
SI_Remote_Port[9]	0	UINT
SI_Remote_Port[10]	0	UINT
SI_Remote_Port[11]	0	UINT
SI_Remote_Port[12]	0	UINT
SI_Remote_Port[13]	0	UINT
SI_Remote_Port[14]	0	UINT
SI_Remote_Port[15]	0	UINT

The screen shot above shows socket 2 connected to remote port 61633 while socket 4 is connected to remote port 2816 and socket 8 on port 502.

SI_Local_Port

ARRAY[0..15] of UINT – These 16 variables show the TCP (or UDP) port number of the local end of a connection for each socket.



SI_Local_Port		ARRAY[0..15] OF UINT
SI_Local_Port[0]	0	UINT
SI_Local_Port[1]	0	UINT
SI_Local_Port[2]	502	UINT
SI_Local_Port[3]	0	UINT
SI_Local_Port[4]	502	UINT
SI_Local_Port[5]	0	UINT
SI_Local_Port[6]	0	UINT
SI_Local_Port[7]	0	UINT
SI_Local_Port[8]	2816	UINT
SI_Local_Port[9]	0	UINT
SI_Local_Port[10]	0	UINT
SI_Local_Port[11]	0	UINT
SI_Local_Port[12]	0	UINT
SI_Local_Port[13]	0	UINT
SI_Local_Port[14]	0	UINT
SI_Local_Port[15]	0	UINT

The screen shot below shows sockets 0 and 4 connected to local port 502 and socket 8 connected to port 2816.

SI_Status

ARRAY[0..15] of BYTE – The SI_Status provides a bit-mapped indication of condition of each socket.

Bit	Meaning	Notes
0	1 = Connection ACTIVE 0 = No connection	
1	1 = ½ closed 0 = normal	½ closed means a FIN was received from the remote end but the socket has not been formally closed.
2	1 = Server Listening (or Client trying to connect) 0 = Not listening or trying	
3-7	Reserved	

SI_Status		ARRAY[0..15] OF BYTE
SI_Status[0]	4	BYTE
SI_Status[1]	4	BYTE
SI_Status[2]	1	BYTE
SI_Status[3]	4	BYTE
SI_Status[4]	1	BYTE
SI_Status[5]	4	BYTE
SI_Status[6]	4	BYTE
SI_Status[7]	4	BYTE
SI_Status[8]	1	BYTE
SI_Status[9]	0	BYTE
SI_Status[10]	0	BYTE
SI_Status[11]	0	BYTE
SI_Status[12]	0	BYTE
SI_Status[13]	0	BYTE
SI_Status[14]	0	BYTE
SI_Status[15]	0	BYTE

The screenshot above shows sockets 2, 4, and 8 connected (value = 1) while sockets 0, 1, 3, 5, 6, and 7 are listening (value = 4). Sockets 9-15 are not enabled.

SI_Out_Handshake_W0

NOTE: This value is used by the TCPOPEN_Outbound DFB and should not be used elsewhere in the PLC program.

INT – This is the feedback handshake value from the UCM to acknowledge the reception of an SO_Data_W0 block. The UCM echoes the value of the SO_Out_Handshake_W0 to the SI_Out_Handshake_W0 after parsing the data block. TCPOPEN_Outbound DFB code watches this value and when the outbound and inbound handshake values are equal, the outbound window may be used for the next transfer of data.

SI_In_Handshake_W0

NOTE: This value is used by the TCPOPEN_Inbound DFB and should not be used elsewhere in the PLC program.

INT – This is the handshake value from the UCM to indicate new data is present in the SI_Data_W0 block. The TCPOPEN_Inbound_DFB compares this value with the SO_In_Handshake_W0 value and if they are different then it can process the new data block. After retrieving the data block, the PLC echoes the value of the SI_In_Handshake_W0 to the SO_In_Handshake_W0 and the UCM seeing that the Out value matches the In value is allowed to place new data into the In block.

SI_Number_W0

NOTE: This value is used by the TCPOPEN_Inbound DFB and should not be used elsewhere in the PLC program.

ARRAY[0..7] of BYTE – This array holds the socket number for each of the possible 8 blocks of inbound data in the SI_Data_W0 window. Valid numbers are 0 through 15. The UCM places a 255 value in unused block location. The TCPOPEN_Inbound DFB code will look at this array to know where to place the incoming socket data.

SI_Length_W0

NOTE: This value is used by the TCPOPEN_Inbound DFB and should not be used elsewhere in the PLC program.

ARRAY[0..7] of INT – This array holds the length for each of the possible 8 blocks of inbound data in the SI_Data_W0 window. Valid lengths are 1 through 1134. The UCM places a 0 value in unused block location. The TCPOPEN_Inbound DFB code will look at this array to know how much of the data is to be placed in the appropriate array.

SI_More_W0

NOTE: This value is used by the TCPOPEN_Inbound DFB and should not be used elsewhere in the PLC program.

BYTE – This byte is a bit-map of the 8 possible data blocks in the SI_Data_W0 window. If a bit in this byte is TRUE then the data for that particular socket is too large to transfer in the current SI_Data_W0. 'More' data is queued to be sent to the PLC in the next handshake transfer.

SI_Data_W0

NOTE: This value is used by the TCPOPEN_Inbound DFB and should not be used elsewhere in the PLC program.

ARRAY[0..1133] OF BYTE – This is the data block for window W0 from the UCM to the PLC. Valid data is placed starting at element [0] of this block up to element [1133]. The data in this block may be from 1 to 8 different sockets.

The UCM loads a value of 252 (decimal) 16#FC (hex) into unused bytes in the window.

Name	Value	Type
SI_Out_Handshake_W0	-1282	INT
SI_In_Handshake_W0	-1318	INT
SI_Number_W0		ARRAY[0..7] OF BYTE
SI_Number_W0[0]	2	BYTE
SI_Number_W0[1]	8	BYTE
SI_Number_W0[2]	255	BYTE
SI_Number_W0[3]	255	BYTE
SI_Number_W0[4]	255	BYTE
SI_Number_W0[5]	255	BYTE
SI_Number_W0[6]	255	BYTE
SI_Number_W0[7]	255	BYTE
SI_Length_W0		ARRAY[0..7] OF INT
SI_Length_W0[0]	12	INT
SI_Length_W0[1]	33	INT
SI_Length_W0[2]	0	INT
SI_Length_W0[3]	0	INT
SI_Length_W0[4]	0	INT
SI_Length_W0[5]	0	INT
SI_Length_W0[6]	0	INT
SI_Length_W0[7]	0	INT
SI_More_W0	0	BYTE
SI_Data_W0		ARRAY[0..1133] OF BYTE
SI_Data_W0[0]	28	BYTE
SI_Data_W0[1]	180	BYTE
SI_Data_W0[2]	0	BYTE
SI_Data_W0[3]	0	BYTE
SI_Data_W0[4]	0	BYTE
SI_Data_W0[5]	6	BYTE
SI_Data_W0[6]	2	BYTE
SI_Data_W0[7]	3	BYTE
SI_Data_W0[8]	0	BYTE
SI_Data_W0[9]	0	BYTE
SI_Data_W0[10]	0	BYTE
SI_Data_W0[11]	20	BYTE
SI_Data_W0[12]	12	BYTE
SI_Data_W0[13]	180	BYTE
SI_Data_W0[14]	0	BYTE
SI_Data_W0[15]	0	BYTE
SI_Data_W0[16]	0	BYTE
SI_Data_W0[17]	27	BYTE
SI_Data_W0[18]	3	BYTE
SI_Data_W0[19]	3	BYTE
SI_Data_W0[20]	24	BYTE
SI_Data_W0[21]	0	BYTE
SI_Data_W0[22]	0	BYTE
SI_Data_W0[23]	24	BYTE
SI_Data_W0[24]	32	BYTE
SI_Data_W0[25]	0	BYTE
SI_Data_W0[26]	0	BYTE
SI_Data_W0[27]	0	BYTE
SI_Data_W0[28]	0	BYTE
SI_Data_W0[29]	4	BYTE
SI_Data_W0[30]	60	BYTE
SI_Data_W0[31]	0	BYTE
SI_Data_W0[32]	0	BYTE
SI_Data_W0[33]	0	BYTE
SI_Data_W0[34]	0	BYTE
SI_Data_W0[35]	0	BYTE
SI_Data_W0[36]	0	BYTE
SI_Data_W0[37]	0	BYTE
SI_Data_W0[38]	0	BYTE
SI_Data_W0[39]	0	BYTE
SI_Data_W0[40]	0	BYTE
SI_Data_W0[41]	0	BYTE
SI_Data_W0[42]	0	BYTE
SI_Data_W0[43]	0	BYTE
SI_Data_W0[44]	0	BYTE
SI_Data_W0[45]	252	BYTE
SI_Data_W0[46]	252	BYTE

The above screenshot shows data from sockets 2 and 8 placed into the SI_Data_W0 window. The references for socket 8 are highlighted.

The data from socket 2 has a length of 12 bytes. The data for socket 2 is located in the first 12 bytes of the SI_Data_W0 array (SI_Data_W0[0]..[11]).

The data from socket 8 has a length of 33 bytes. The data for this socket is in the next 33 bytes after the data from socket 2 (SI_Data_W0[12]...[44]).

Note: SI_More_W0 = 0 which tells the DFB that this is the complete socket frame data for all entries in the list.

Outputs

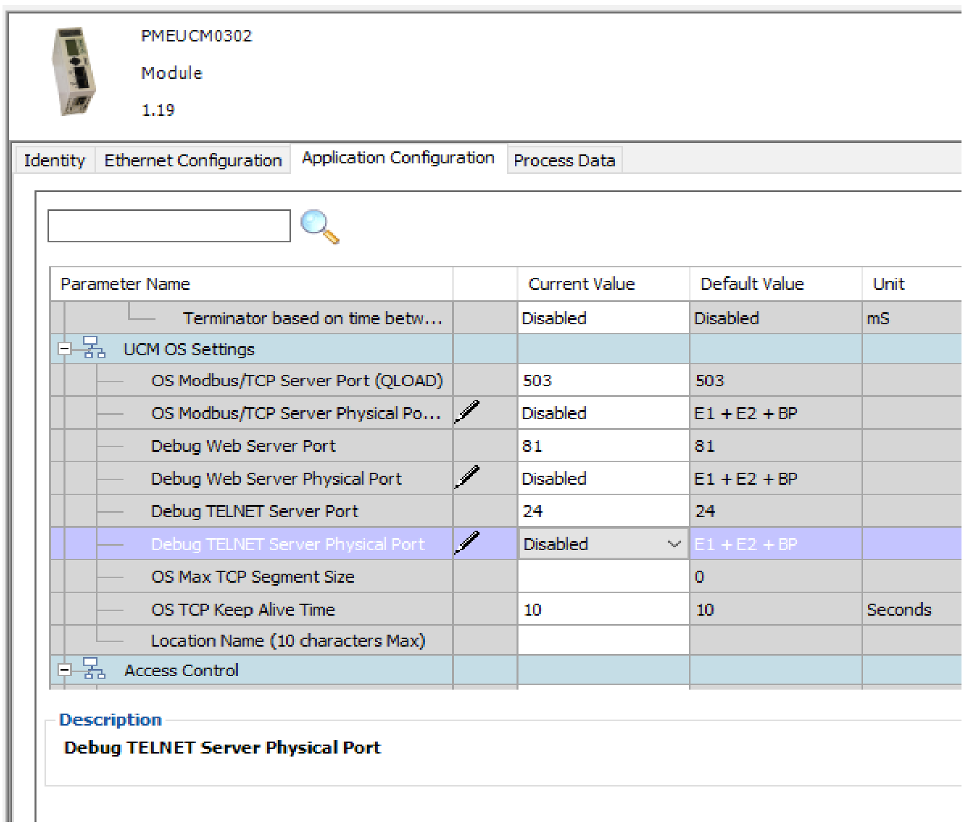
UCM_Command

WORD – This is a bit-mapped command to temporarily override operation of the TCPOPEN application. These settings are normally controlled by the DTM.

Bits	Meaning	Notes
0	1 = Force on OS MBTCP Server 0 = Use DTM Setting	Uses TCP port 503 when ON.

Bits	Meaning	Notes
1	1 = Force on Web Server 0 = Use DTM Setting	Uses TCP port 81 when ON.
2	1 = Force on Telnet Server 0 = Use DTM Setting	Uses TCP port 24 when ON.
3	1 = Turn on Red LED behind Screen 0 = Turn off LED behind screen	
4	1 = Force a single read of Primary PRN file from FDR server. 0 = Normal	Rising edge triggers a single read operation. Bit must be zeroed before triggered again.
5	1 = Force a single read of Secondary PRN file from HSBY Secondary FDR server. 0 = Normal	Rising edge triggers a single read operation. Bit must be zeroed before triggered again.
6	1 = Force a single write of Primary PRN file into Secondary HSBY FDR server.	Rising edge triggers a single write. Bit must be zeroed before triggered again. Write only occurs if Primary PRM is valid and secondary PRM has zero length or miss-matched checksum.
7	1 = Force off ACL	Temporarily allows inbound connections to web server or OS Modbus/TCP server.
8	1 = Mute buzzer	Turns off all system sounds in the PMEUCM.

In the screenshot below, the DTM is configured to disable all three features. The PLC may set the three bits in the UCM_Command word to override the DTM settings and enable the OS Modbus/TCP server (port 503), Debug Web Server (port 81), and Debug Telnet Server (port 24).



SO_Remote_IP

ARRAY[0..15] of DWORD – These 16 variables declare the target IP Address for a client connection on a socket compressed into a DWORD. Each 8 bit BYTE of the DWORD is the octet of the IP Address.

For example, if the remote IP Address for socket 3 is 206.223.51.16 then

SO_Remote_IP[1] = 3470734096(dec) = 16#CEDF_3310(hex)

Looking at the value in hexadecimal reveals the IP Address:

CE (hex) = 206 (decimal)

DF (hex) = 223 (decimal)

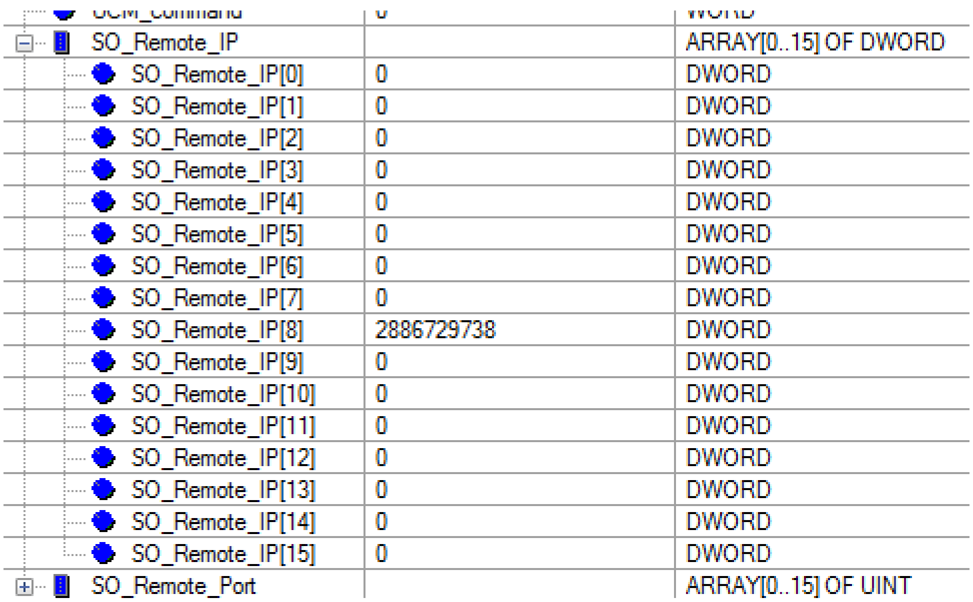
33 (hex) = 51 (decimal)

10 (hex) = 16 (decimal)

A simple method for moving this data to the DWORD variable is to use the ‘Shift Left’ (SHL) ST command. The following code would move the address 206.223.51.16 into the SO_Remote_IP of socket 12.

```
PME_UCM_0302_TCPOPEN.Outputs.SO_Remote_IP[12] :=
        DINT_TO_DWORD(SHL(206,24)+SHL(223,16)+SHL(51,8)+SHL(16,0));
```

The first SRL shifts the 206 value 24 bits into the MSB of the DWORD. The next lines shift 16 bits and 8 bits. The last line doesn’t shift at all and could just be a straight assignment. The DINT_TO_DWORD conversion is required because Unity treats the addition of the SHL of constants as a DINT.



UCM_command	0	WORD
SO_Remote_IP		ARRAY[0..15] OF DWORD
SO_Remote_IP[0]	0	DWORD
SO_Remote_IP[1]	0	DWORD
SO_Remote_IP[2]	0	DWORD
SO_Remote_IP[3]	0	DWORD
SO_Remote_IP[4]	0	DWORD
SO_Remote_IP[5]	0	DWORD
SO_Remote_IP[6]	0	DWORD
SO_Remote_IP[7]	0	DWORD
SO_Remote_IP[8]	2886729738	DWORD
SO_Remote_IP[9]	0	DWORD
SO_Remote_IP[10]	0	DWORD
SO_Remote_IP[11]	0	DWORD
SO_Remote_IP[12]	0	DWORD
SO_Remote_IP[13]	0	DWORD
SO_Remote_IP[14]	0	DWORD
SO_Remote_IP[15]	0	DWORD
SO_Remote_Port		ARRAY[0..15] OF UINT

The above screenshot shows socket 8 (as a client) connecting to remote IP Address 2886729738 (decimal) = AC10000A (hex) which indicates the remote IP Address of 172.16.0.10.

- AC (hex) = 172 (decimal)
- 10 (hex) = 16 (decimal)
- 00 (hex) = 0 (decimal)
- 0A (hex) = 10 (decimal)

SO_Remote_Port

ARRAY[0..15] of UINT – These 16 variables declare the target TCP (or UDP) port number for a client connection on a socket. This value is ignored for Server connections.

SO_Remote_Port		ARRAY[0..15] OF UINT
SO_Remote_Port[0]	0	UINT
SO_Remote_Port[1]	0	UINT
SO_Remote_Port[2]	0	UINT
SO_Remote_Port[3]	0	UINT
SO_Remote_Port[4]	0	UINT
SO_Remote_Port[5]	0	UINT
SO_Remote_Port[6]	0	UINT
SO_Remote_Port[7]	0	UINT
SO_Remote_Port[8]	502	UINT
SO_Remote_Port[9]	0	UINT
SO_Remote_Port[10]	0	UINT
SO_Remote_Port[11]	0	UINT
SO_Remote_Port[12]	0	UINT
SO_Remote_Port[13]	0	UINT
SO_Remote_Port[14]	0	UINT
SO_Remote_Port[15]	0	UINT
SO_Local_Port		ARRAY[0..15] OF UINT

The above screenshot shows socket 8 targeting port 502.

SO_Local_Port

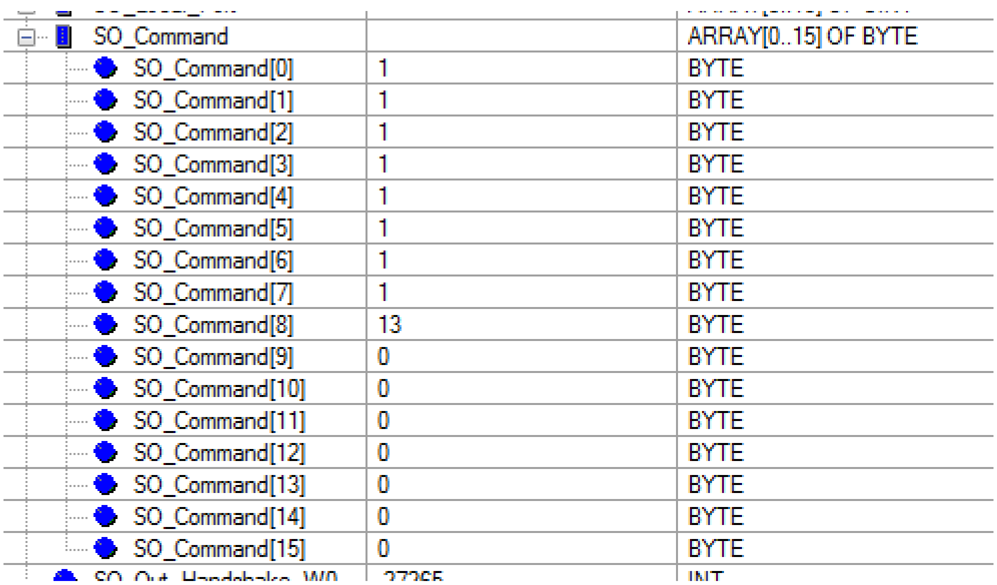
ARRAY[0..15] of UINT – These 16 variables declare the local TCP (or UDP) port number for a server or client connection on a socket. A value of zero for a client connection allows the UCM OS to choose an appropriate ephemeral port number. The screenshot below shows sockets 0-7 listening on port 502 and sockets 8-15 are not configured.

SO_Remote_Port		ARRAY[0..15] OF UINT	TCP or UDP Remote Port Number; Ignred during Server Operation
SO_Local_Port		ARRAY[0..15] OF UINT	TCP or UDP Local Port Number; 0=Automatic for Client Operation
SO_Local_Port[0]	502	UINT	
SO_Local_Port[1]	502	UINT	
SO_Local_Port[2]	502	UINT	
SO_Local_Port[3]	502	UINT	
SO_Local_Port[4]	502	UINT	
SO_Local_Port[5]	502	UINT	
SO_Local_Port[6]	502	UINT	
SO_Local_Port[7]	502	UINT	
SO_Local_Port[8]	0	UINT	
SO_Local_Port[9]	0	UINT	
SO_Local_Port[10]	0	UINT	
SO_Local_Port[11]	0	UINT	
SO_Local_Port[12]	0	UINT	
SO_Local_Port[13]	0	UINT	
SO_Local_Port[14]	0	UINT	
SO_Local_Port[15]	0	UINT	
SO_Command		ARRAY[0..15] OF BYTE	.0=Connect; .1=Reset; .2=Port2; .3=Client; .4=UDP; .5=No Nagle;
SO_Socket_State		ARRAY[0..15] OF BYTE	Socket State for display on front LCD

SO_Command

ARRAY[0..15] of BYTE – These 16 bit-mapped variables control the sockets in the UCM.

Bit	Meaning	Notes
0	1 = Connect 0 = Close	Clearing bit sends RST on open connections.
1	1 = Reserved for Serial 0 = Use Ethernet	Current version ignores this bit. Leave this bit FALSE
2	1 = Use Port 2 0 = Use Port 1	
3	1 = Client 0 = Server	
4	1 = UDP 0 = TCP	



SO_Command		ARRAY[0..15] OF BYTE
SO_Command[0]	1	BYTE
SO_Command[1]	1	BYTE
SO_Command[2]	1	BYTE
SO_Command[3]	1	BYTE
SO_Command[4]	1	BYTE
SO_Command[5]	1	BYTE
SO_Command[6]	1	BYTE
SO_Command[7]	1	BYTE
SO_Command[8]	13	BYTE
SO_Command[9]	0	BYTE
SO_Command[10]	0	BYTE
SO_Command[11]	0	BYTE
SO_Command[12]	0	BYTE
SO_Command[13]	0	BYTE
SO_Command[14]	0	BYTE
SO_Command[15]	0	BYTE

The above screenshot shows sockets 0 through 7 commanded to connect as a TCP server on E1.

- Bit 0 = 1 (Connect)
- Bit 1 = 0 (Ethernet)
- Bit 2 = 0 (E1)
- Bit 3 = 0 (Server)
- Bit 4 = 0 (TCP)

Socket 8 has a value of 13 (decimal) = 1101 (binary) which indicates a TCP client on E2.

- Bit 0 = 1 (Connect)

- Bit 1 = 0 (Ethernet)
- Bit 2 = 1 (E2)
- Bit 3 = 1 (Client)
- Bit 4 = 0 (TCP)

Further SO_Command Examples:

To set socket 3 to be a TCP Server on E1 using window W0 with Nagle ON use the following commands to set the individual command bits:

UCM1.Outputs.SO_Command.0 = TRUE; {Connect}

UCM1.Outputs.SO_Command.1 = FALSE; {Enet}

UCM1.Outputs.SO_Command.2 = FALSE; {E1}

UCM1.Outputs.SO_Command.3 = FALSE; {Server}

UCM1.Outputs.SO_Command.4 = FALSE; {TCP}

SO_Out_Handshake_W0

NOTE: This value is used by the TCPOPEN_Outbound DFB and should not be used elsewhere in the PLC program.

INT – This is the feedback handshake value from the PLC to indicate to the UCM that new data is included in the DIO transfer. The UCM echos the value of the SO_Out_Handshake_W0 to the SI_Out_Handshake_W0 after parsing the data block.

SO_In_Handshake_W0

NOTE: This value is used by the TCPOPEN_Inbound DFB and should not be used elsewhere in the PLC program.

INT – This is the handshake value from the PLC to indicate new data has been parsed from SI_Data_W0 block. The TCPOPEN_Inbound_DFB copies the SI_IN_Handshake_W0 value to this variable to indicate that the new data has been received.

SO_Number_W0

NOTE: This value is used by the TCPOPEN_Outbound DFB and should not be used elsewhere in the PLC program.

ARRAY[0..7] of BYTE – This array holds the socket number for each of the possible 8 blocks of outbound data in the SO_Data_W0 window. Valid numbers are 0 through 15. The DFB places a 255 value in unused block location. The

UCM will look at this array to know where to transmit the outbound socket data.

SO_Length_W0

NOTE: This value is used by the TCPOPEN_Outbound DFB and should not be used elsewhere in the PLC program.

ARRAY[0..7] of INT – This array holds the length for each of the possible 8 blocks of inbound data in the SO_Data_W0 window. Valid lengths are 1 through 1180. The UCM places a 0 value in unused block location. The TCPOPEN_Inbound DFB code will look at this array to know how much of the data is to be placed in the appropriate array.

SO_More_W0

NOTE: This value is used by the TCPOPEN_Outbound DFB and should not be used elsewhere in the PLC program.

BYTE – This byte is a bit-map of the 8 possible data blocks in the SO_Data_W0 window. If a bit in this byte is TRUE then the data for that particular socket is too large to transfer in the current SO_Data_W0. 'More' data is queued to be sent to the PLC in the next handshake transfer.

SO_Data_W0

NOTE: This value is used by the TCPOPEN_Outbound DFB and should not be used elsewhere in the PLC program.

ARRAY[0..1179] OF BYTE – This is the data block for window W0 from the UCM to the PLC. Valid data is placed starting at element [0] of this block up to element [1179]. The data in this block may be from 1 to 8 different sockets.

Note: The DFB loads the value 253 (decimal) into unused bytes below the actual data.

Name	Value	Type
SO_Out_Handshake_W0	19733	INT
SO_In_Handshake_W0	19696	INT
SO_Number_W0		ARRAY[0..7] OF BYTE
SO_Number_W0[0]	2	BYTE
SO_Number_W0[1]	8	BYTE
SO_Number_W0[2]	255	BYTE
SO_Number_W0[3]	255	BYTE
SO_Number_W0[4]	255	BYTE
SO_Number_W0[5]	255	BYTE
SO_Number_W0[6]	255	BYTE
SO_Number_W0[7]	255	BYTE
SO_Length_W0		ARRAY[0..7] OF INT
SO_Length_W0[0]	49	INT
SO_Length_W0[1]	12	INT
SO_Length_W0[2]	0	INT
SO_Length_W0[3]	0	INT
SO_Length_W0[4]	0	INT
SO_Length_W0[5]	0	INT
SO_Length_W0[6]	0	INT
SO_Length_W0[7]	0	INT
SO_More_W0	0	BYTE
SO_Data_W0		ARRAY[0..1179] OF BYTE
SO_Data_W0[0]	69	BYTE
SO_Data_W0[1]	128	BYTE
SO_Data_W0[2]	0	BYTE
SO_Data_W0[3]	0	BYTE
SO_Data_W0[4]	0	BYTE
SO_Data_W0[5]	43	BYTE
SO_Data_W0[6]	2	BYTE
SO_Data_W0[7]	3	BYTE
SO_Data_W0[8]	40	BYTE
SO_Data_W0[9]	26	BYTE
SO_Data_W0[10]	71	BYTE
SO_Data_W0[11]	0	BYTE
SO_Data_W0[12]	9	BYTE
SO_Data_W0[13]	139	BYTE
SO_Data_W0[14]	64	BYTE
SO_Data_W0[15]	0	BYTE
SO_Data_W0[16]	0	BYTE
SO_Data_W0[17]	16	BYTE
SO_Data_W0[18]	218	BYTE
SO_Data_W0[19]	0	BYTE
SO_Data_W0[20]	0	BYTE
SO_Data_W0[21]	0	BYTE
SO_Data_W0[22]	0	BYTE
SO_Data_W0[23]	0	BYTE
SO_Data_W0[24]	0	BYTE
SO_Data_W0[25]	0	BYTE
SO_Data_W0[26]	0	BYTE
SO_Data_W0[27]	0	BYTE
SO_Data_W0[28]	0	BYTE
SO_Data_W0[29]	0	BYTE
SO_Data_W0[30]	0	BYTE
SO_Data_W0[31]	0	BYTE
SO_Data_W0[32]	0	BYTE
SO_Data_W0[33]	0	BYTE
SO_Data_W0[34]	0	BYTE
SO_Data_W0[35]	0	BYTE
SO_Data_W0[36]	0	BYTE
SO_Data_W0[37]	0	BYTE
SO_Data_W0[38]	0	BYTE
SO_Data_W0[39]	0	BYTE
SO_Data_W0[40]	0	BYTE
SO_Data_W0[41]	0	BYTE
SO_Data_W0[42]	0	BYTE
SO_Data_W0[43]	0	BYTE
SO_Data_W0[44]	0	BYTE
SO_Data_W0[45]	0	BYTE
SO_Data_W0[46]	0	BYTE
SO_Data_W0[47]	98	BYTE
SO_Data_W0[48]	217	BYTE
SO_Data_W0[49]	50	BYTE
SO_Data_W0[50]	144	BYTE
SO_Data_W0[51]	0	BYTE
SO_Data_W0[52]	0	BYTE
SO_Data_W0[53]	0	BYTE
SO_Data_W0[54]	6	BYTE
SO_Data_W0[55]	2	BYTE
SO_Data_W0[56]	3	BYTE
SO_Data_W0[57]	0	BYTE
SO_Data_W0[58]	0	BYTE
SO_Data_W0[59]	0	BYTE
SO_Data_W0[60]	11	BYTE
SO_Data_W0[61]	253	BYTE
SO_Data_W0[62]	253	BYTE

The above screenshot shows output data for both sockets 2 and 8. The information for socket 8 is highlighted for clarity.

Socket 2 data has a length of 49 bytes so the actual frame data is located in SO_Data_W0[0]...[48].

Socket 8 data has a length of 12 bytes which corresponds to SO_Data[49]...[60].

TCPOPEN DFBs

Two DFBs are provided to automatically control the socket data to/from the UCM. They both use the DTM structure for the PME UCM 0302_TCPOPEN and the UCM1_Data_IN (or OUT) array and the UCM1_Data_IN_Length (or _OUT) array.

These two DFBs will automatically be installed if both of these section files are imported into a Unity Pro project:

c:\Niobrara\apps\PMEUCM\TCPOPEN\tcpopen_example1.xbd

or with these two:

c:\Niobrara\apps\PMEUCM\TCPOPEN\tcpopen_data_in.xbd

and

c:\Niobrara\apps\PMEUCM\TCPOPEN\tcpopen_data_out.xbd

These DFBs may be installed manually into a Unity Pro project by using these files:

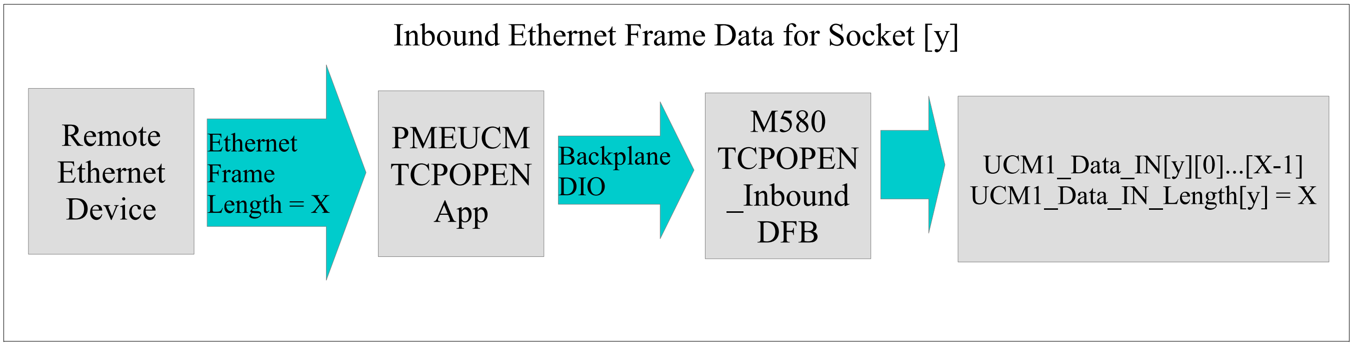
c:\Niobrara\apps\PMEUCM\TCPOPEN\tcpopen_inbound.xdb

and

c:\Niobrara\apps\PMEUCM\TCPOPEN\tcpopen_outbound.xdb

TCPOPEN_Inbound

NOTE: For the following discussion, it is assumed that the sockets in question are connected to remote servers (or clients). TCP and UDP operate in the same manner.



The above figure shows the overview of the process of an Ethernet frame of length X sent to the PMEUCM and the same data arriving in the UCM1_Data_IN array for the socket with the length X.

The TCPOPEN UCM application works with the DTM and TCPOPEN_Inbound DFB to successfully transfer the full Ethernet frame data to the correct array of bytes in the M580.

The incoming Ethernet frame from the remote device may have a length of 1 to 1452 bytes. (The UCM's Window size sets this limit of 1452 bytes.)

The TCPOPEN DTM provides an an inbound (to PLC) data window that is used to transfer the incoming socket data to the PLC. M580 DTM byte count restrictions limit this window to a maximum of 1134 bytes for the inbound DIO. The PMEUCM application and the TCPOPEN_Inbound DFB work together to break long messages into multiple DIO transactions and then re-assemble the original frame data for the PLC to use.

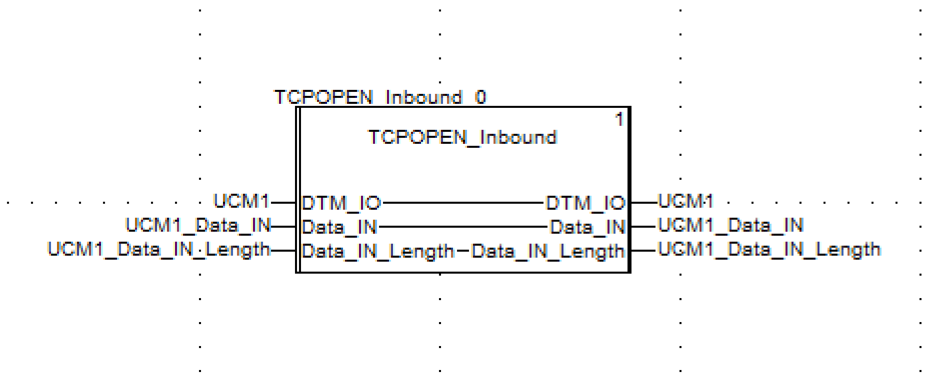
The TCPOPEN_Inbound DFB loads the complete frame data into the UCM1_Data_IN array of bytes for the particular socket and sets the UCM1_Data_IN_Length value for that socket when the entire frame data is present. The PLC code simply watches for

UCM1_Data_IN_Length[socket#] > 0

to know that new data has arrived and needs to be processed. When the PLC code is finished with the socket data, it sets

UCM1_Data_IN_Length[socket#] := 0

to signal TCPOPEN_Inbound that it can post new data as it arrives.



Most of the time, incoming Ethernet frames are smaller than 1134 bytes. Additionally, data from several sockets may come in to the UCM at the same time and get queued up for the DIO transfer to the M580. The UCM and the DFB work together to pack as many frame bytes into the 1134 byte window to optimize the DIO operation.

Up to 8 'blocks' of socket data may be packed into the window. For each 'block' there is a byte to indicate the socket number and an INT to indicate the length of the block. The amount of room in the last block may not be large enough to hold the queued data for the socket so the SI_More.[socketnumber] bit is set to tell the DFB that it needs to wait for the rest of the socket data before setting the UCM1_Data_IN_Length to the proper value.

The window will be filled until one of the following occurs:

- The data window is filled (1180 bytes PLC>UCM or 1134 bytes UCM > PLC)
- All 8 blocks are used (even if the window total byte count is < window size)
- Or, there is no more queued data to transfer.

For ease of example: let's assume the window is only 1000 bytes and Socket 1 needs to send 150 bytes while Socket 5 sends 500 and socket 9 sends 300. There are 50 unused bytes in the window but that is all of the data to be transferred.

Block	**Socket Number**	**Length**	**Window Bytes**	**More Flag**	**Socket Bytes**
0	1	150	Bytes 0..149	0	0..149
1	5	500	Bytes 150..649	0	0..499
2	9	300	Bytes 650..949	0	0..299
3	255	0		0	
4	255	0		0	
5	255	0		0	

6	255	0	0
7	255	0	0

Now, suppose that the above example also includes data from socket 10 of 100 bytes and socket 11 of 250 bytes that also needed to be sent at the same time. The rest of this window (until windowsize is matched) will be filled and the More flag will be set to tell the other side to buffer this data as more is coming (so wait for the next handshake exchange).

Block	Socket Number	Length	Window Bytes	More Flag	Block Bytes
0	1	150	Bytes 0..149	0	0..149
1	5	500	Bytes 150..649	0	0..499
2	9	300	Bytes 650..949	0	0..299
3	10	50	Bytes 950...999	1	0..49
4	255	0		0	
5	255	0		0	
6	255	0		0	
7	255	0		0	

The next transaction would look like this:

Block	Socket Number	Length	Window Bytes	More Flag	Block Bytes
0	10	50	Bytes 0..49	0	50..99
1	11	250	Bytes 50..299	0	0..249
2	255	0		0	
3	255	0		0	
4	255	0		0	
5	255	0		0	
6	255	0		0	
7	255	0		0	

When the other side sees the ‘More’ flag = 0, it can then send the entire 100 bytes out socket 10.

If a socket has more data bytes than the window size, it may take the entire window for this transaction and the first part of the next transaction window.

When one side (CPU or UCM) has data it needs to send to the other, that side waits for the directional handshake variables to become equal which means that the window is available. The side then claims the window, sets the block socket number(s), block socket length(s), copies the block data into the window, sets the more flag if needed, and then increments the handshake. When the other end sees the handshake become unequal, it looks at the socket number and then the code that handles that socket copies the data from the window and then echoes the handshake value to free up the window.

PLC State Machines

The programming languages of the M580 do not allow for multi-threaded operation where a given operation may wait for a long period of time before receiving a response. Therefore, a state machine is a useful method of program flow control. The ST examples included with the TCPOPEN files use some form of state machine.

The ucm1_modbus_server_sr_v1_01.xst Modbus Server example uses a machine with five states.

- State 0 – Forces the socket to be closed by setting the command byte to zero. The state is then advanced to 10.
- State 10 – The code stays in this state until the socket status reported by the UCM also returns to zero indicating that the socket is closed. The state is then advanced to 20.
- State 20 – Sets up the configuration of the sockets in use (port E1 or E2, TCP, remote TCP port number) and commands the UCM to listen on the socket. The state is then advanced to 30.
- State 30 – The socket is waiting for a connection from a remote client. As soon as a connection is established, the state is advanced to 100.
- State 100 – The socket is connected and waiting for a Modbus/TCP message from the client. When the UCM1_Data_IN_Length[UCM_socketnumber] > 0 then a new message has been received from the client. The inbound Modbus query is parsed and a reply is built and told to transmit by setting UCM1_Data_OUT_Length[UCM_socketnumber] to the new length.

There are other ways of moving through this state machine. If the socket is marked as not connected, the machine will revert to state 0. There are timers running to detect if the socket is idle too long causing it to revert to state 0 if needed.

Socket Control and Timing

It is advised that the PLC program take active control of how long a socket is allowed to attempt to connect or disconnect. Timers are needed for deciding how long to wait for a reply to a query. These timers do not need to be highly accurate or precise. Most Ethernet events can deal with several seconds of leeway.

The example ST segments included in the TCPOPEN setup use a very simple timer technique that is easy to implement with a minimum of code.

SYSUPTIME

The TCPOPEN_Inbound DFB includes a public UDINT called SYSUPTIME as used as a free-running upcounter that increments once per second from the time the PLC starts running the program.

```
PLC_Seconds := BCD_TO_INT(%SW50/256);
if PLC_Seconds <> Last_PLC_Seconds then
        Last_PLC_Seconds := PLC_Seconds;
        SYSUPTIME := SYSUPTIME + 1;
end_if;
```

System word %SW50 provides the PLC's Real Time Clock (RTC) count of seconds in BCD with the seconds in the MSB of %SW50. The first line of code converts this value into an INT and places the seconds in the variable PLC_Seconds. The remainder of this code increments the UDINT variable SYSUPTIME once per second.

This SYSUPTIME may then be used to set a future reference time by adding a number of seconds to this value and store the result in another UDINT variable.

For example, the following ST code would add 10 seconds to the current SYSUPTIME.

```
FutureTime := TCPOPEN_Inbound_0.SYSUPTIME + 10;
```

Later, a simple compare will determine if the ‘timer’ is expired:

```
if SYSUPTIME >= FutureTime then
        (* the timer has expired *)
end_if;
```

This type of timer is handy because it works with all sockets and the test for 'expired' is very simple.

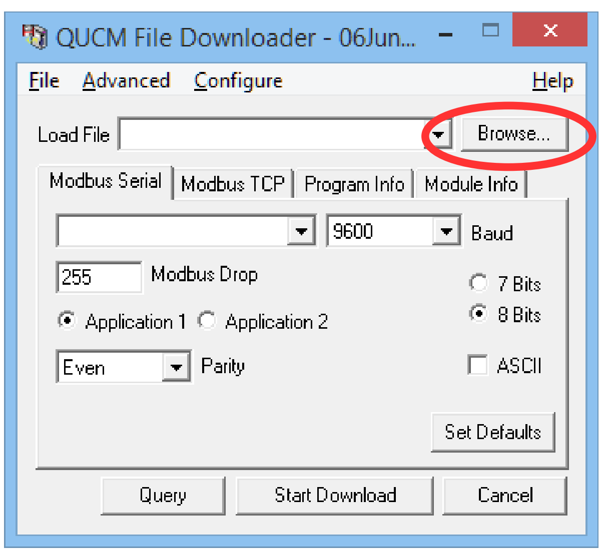
6 QLOAD the TCPOPEN UCM Application

The standard PMEUCM is shipped from the factory with the TCPOPEN application preloaded. QLOAD is used update the original version with new versions.

The QLOAD utility is installed by the PMEUCM_SETUP.EXE program. The user may access this file at:
http://www.niobrara.com/programs/PMEUCM_SETUP.EXE,

QLOAD the TCPOPEN Application

The QLOAD utility is used to load applications into the PMEUCM. Start QLOAD by Start > Programs > Niobrara > QLOAD. The first time QLOAD is started, it should look something like this:



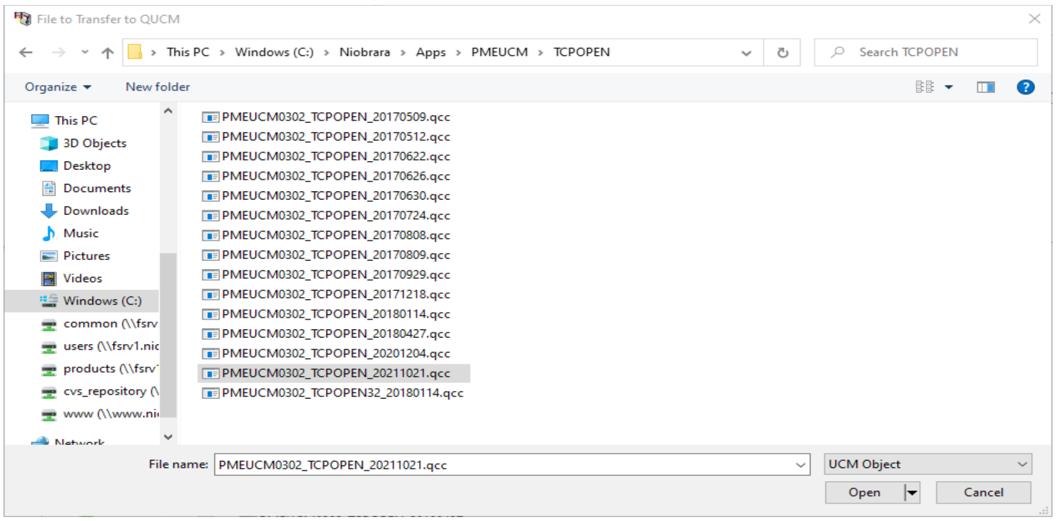
Click on the Browse button and select the TCPOPEN file.

NOTE: There may be multiple versions of the TCPOPEN file. These versions will have filenames of the form: PMEUCM0302_TCPOPEN_xxxzYYzz.qcc where xxxx is the

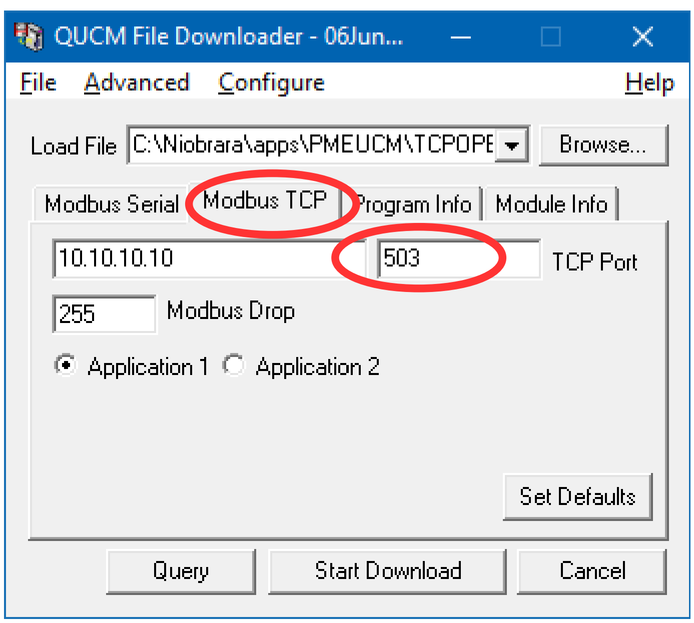
Year, YY is the Month, and zz is the Day.

For example, the TCPOPEN application of version 21OCT2021 would have this filename:

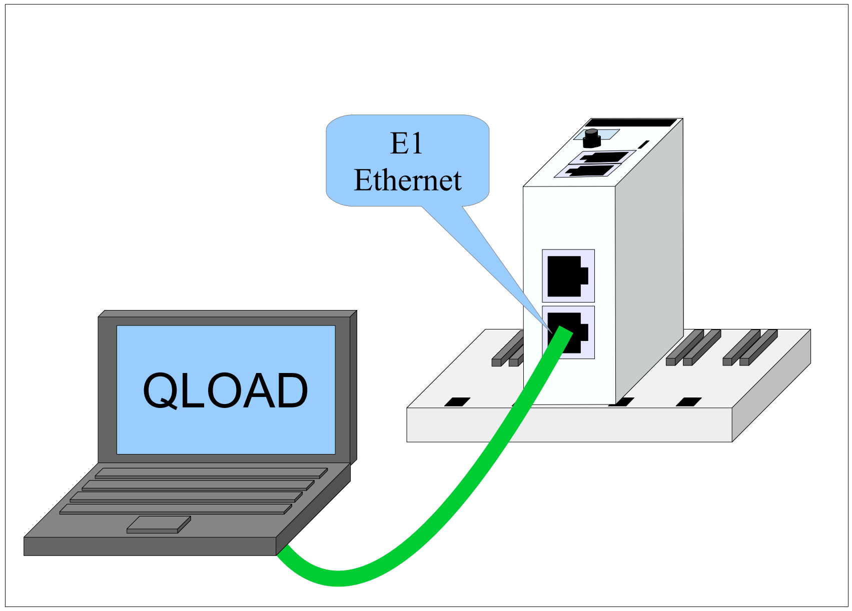
C:\Niobrara\apps\PMEUCM\TCPOPEN\PMEUCM0302_TCPOPEN_20211021.qcc



Now select the ModbusTCP tab.



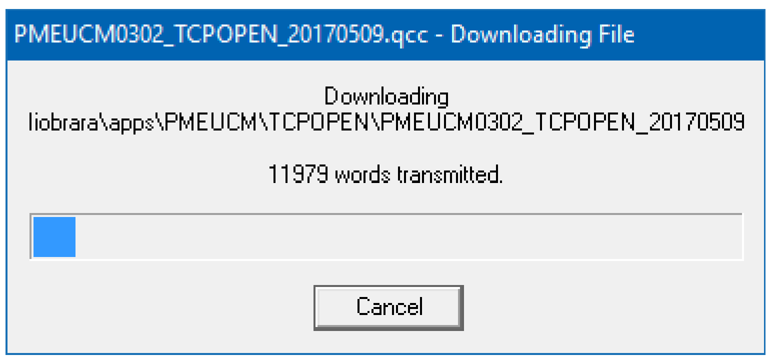
Make sure that the IP Address is set to match the PMEUCM E1 port of 10.10.10.10, the TCP Port is set to 503, Modbus Drop is 255, and Application 1 radio button is set.



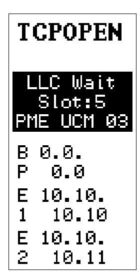
Connect the Ethernet port of the computer to E1 on the PMEUCM with a standard CAT5/6 cable.

Set the Ethernet port of the computer to be on the same 10.10.10.x subnet as the PMEUCM.

Press “Start Download” to begin the loading of the program into the PMEUCM.



When the download is finished, the program should automatically start and the screen should look something like this:



This screen shows that the UCM is located in Rack Slot 5 and is waiting on the M580 PLC to inform it of the Rack Name. Once the UCM has the name of its rack, it is allowed to perform DHCP to obtain the IP Address for the backplane.

In this screen shot, the UCM's backplane is set to an IP Address of 0.0.0.0 and is effectively disabled.

The UCM Ethernet port E1 is at the factory default IP Address of 10.10.10.10.

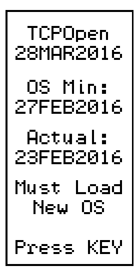
UCM E2 is at 10.10.10.11.

UCM BOOT firmware too old

It is possible that the screen shows that the UCM BOOT code is not current and must be updated. Please download the latest version of the PUCM_SETUP.EXE file from http://www.niobrara.com/programs/PMEUCM_SETUP.EXE and follow instructions in Chapter 6, Loading new firmware over Ethernet.

UCM OS too old

It is possible that the screen shows that the UCM OS is not current and must be updated. Please download the latest version of the PUCM_SETUP.EXE file from http://www.niobrara.com/programs/PMEUCM_SETUP.EXE and follow instructions in Chapter 6, Loading new firmware over Ethernet.



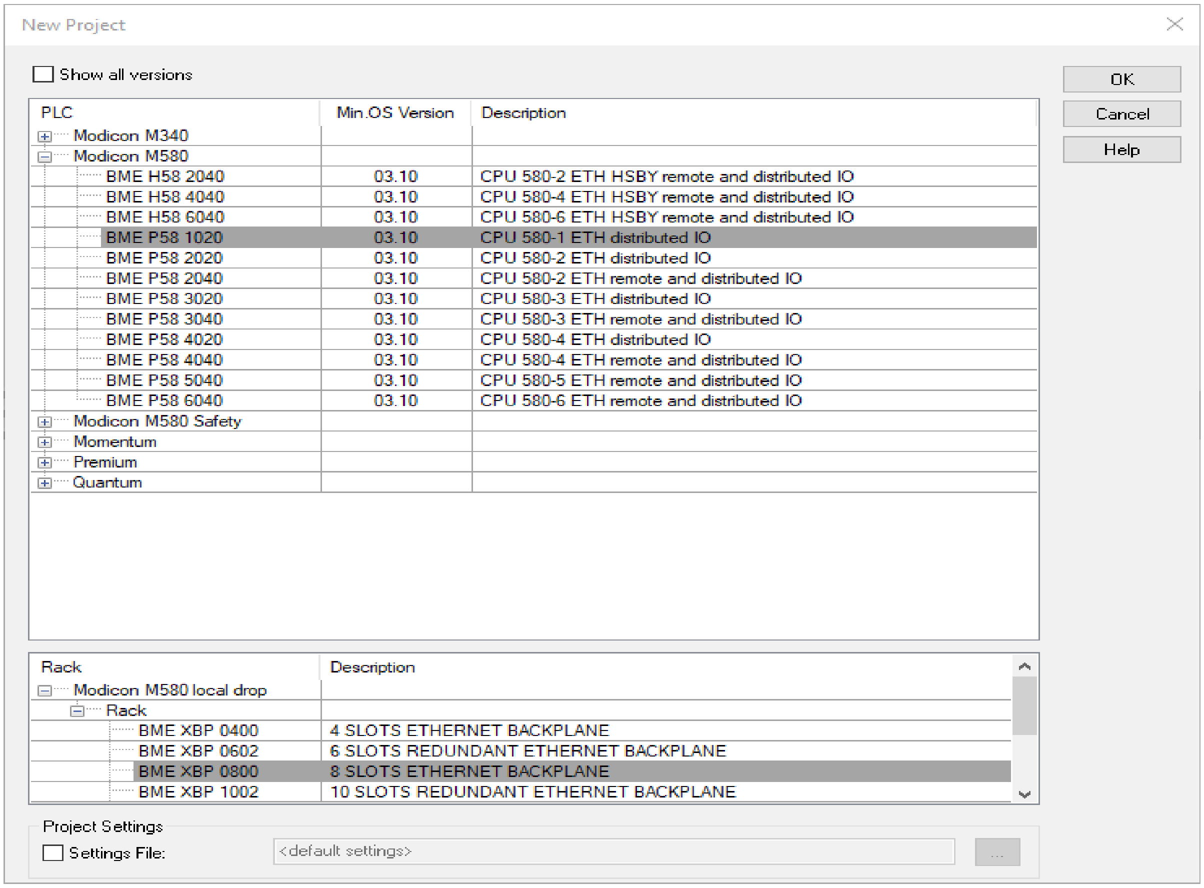
7 Control Expert Operations

New Project

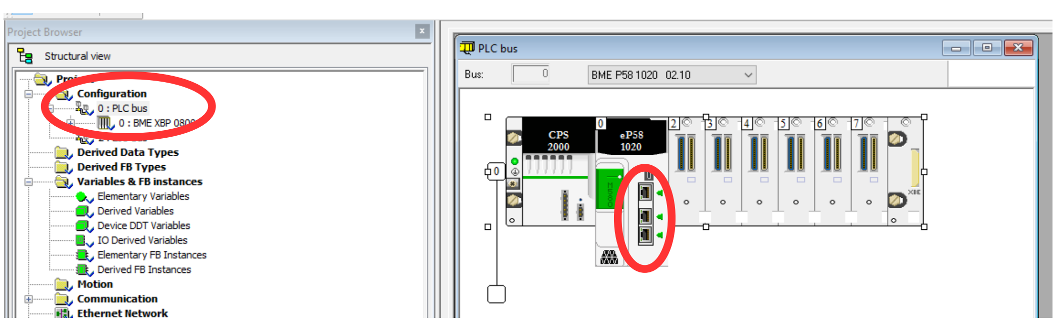
This example starts with a new project in Control Expert V14.1.

- The PME UCM 0302 will be installed in the CPU rack slot 6.
- The M580 P581020 is the chosen CPU.
- Most of the IP Addresses will be left at their default settings.
    - The CPU will be at the default IP Address of 192.168.10.1
    - The PME UCM backplane will be at 192.168.10.3
    - The PME UCM E1 and E2 ports will be set to 172.16.0.10 and 172.16.0.11

The BME P58 1020 CPU is chosen, along with a BME XBP 0800 eight slot Ethernet backplane.



Double-Click on the the 'PLC Bus' to see the CPU rack view:

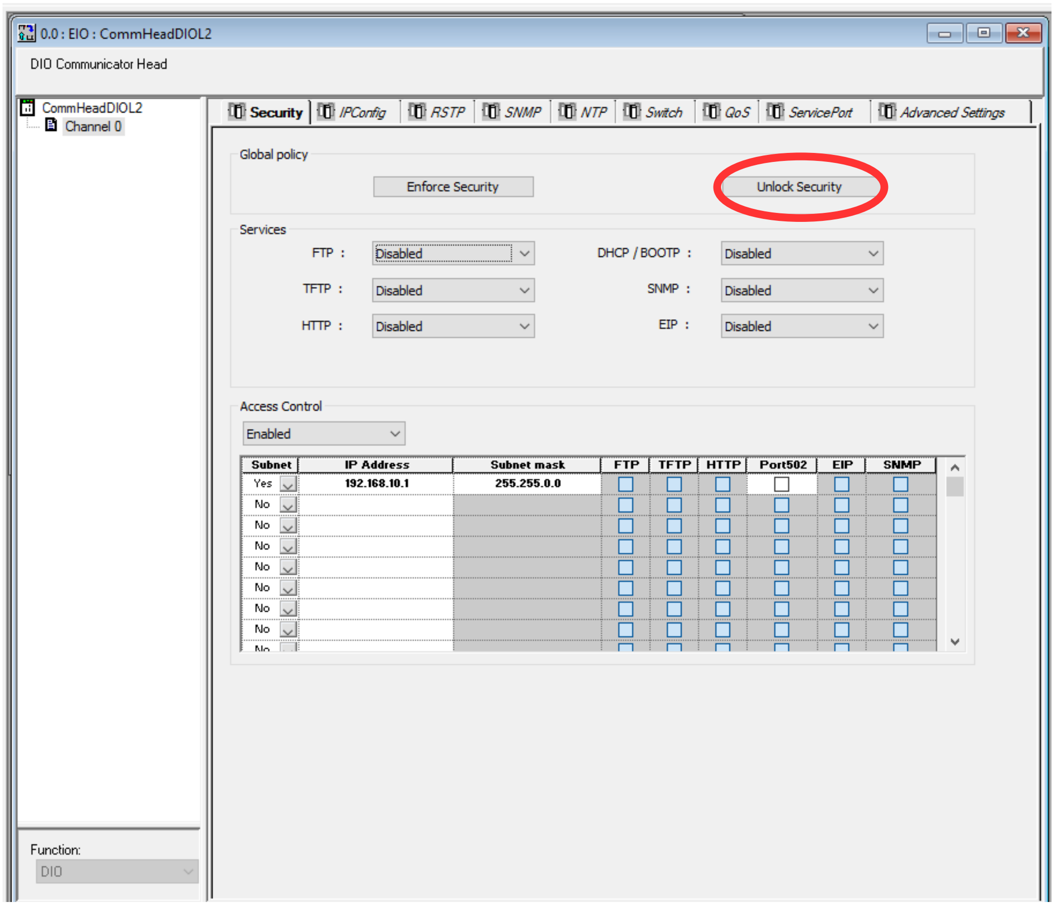


After selecting the "PLC Bus" in the Structural View Tree, double-click on the Ethernet ports of the CPU to open the configuration submodule.

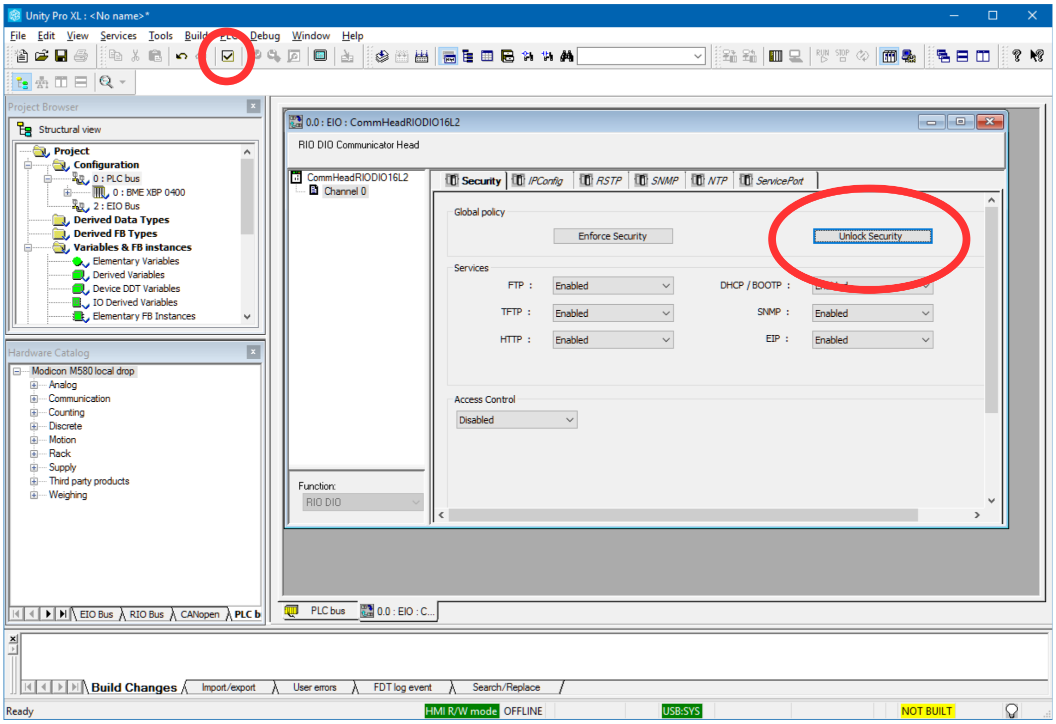
The following services must be enabled in the DIO Master:

- FTP
- TFTP
- DHCP/BOOTP
- EIP

The simple method to enable these services is to select the ‘Unlock Securty’ button.



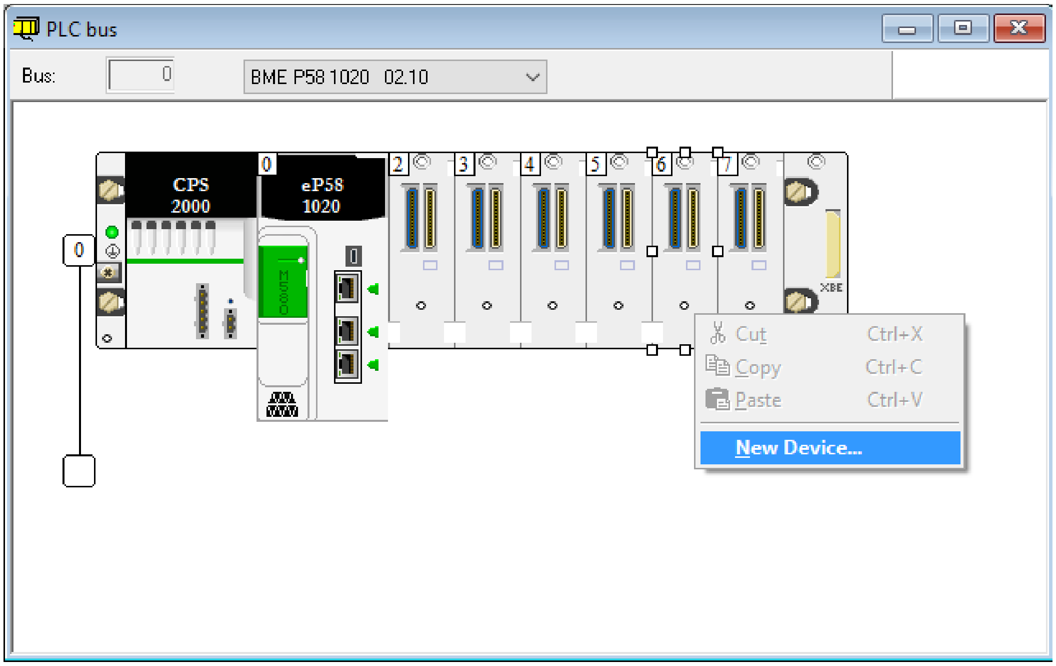
After unlocking the security, click the check box in the tool bar to accept the change.



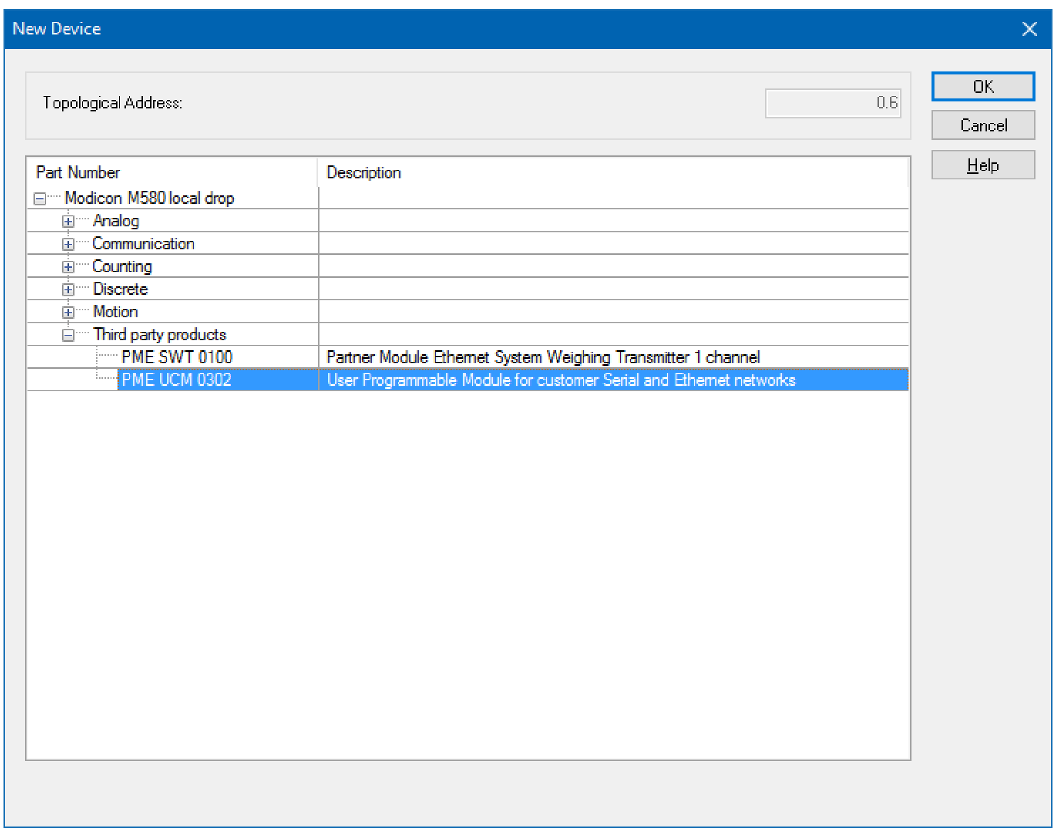
Now close the submodule.

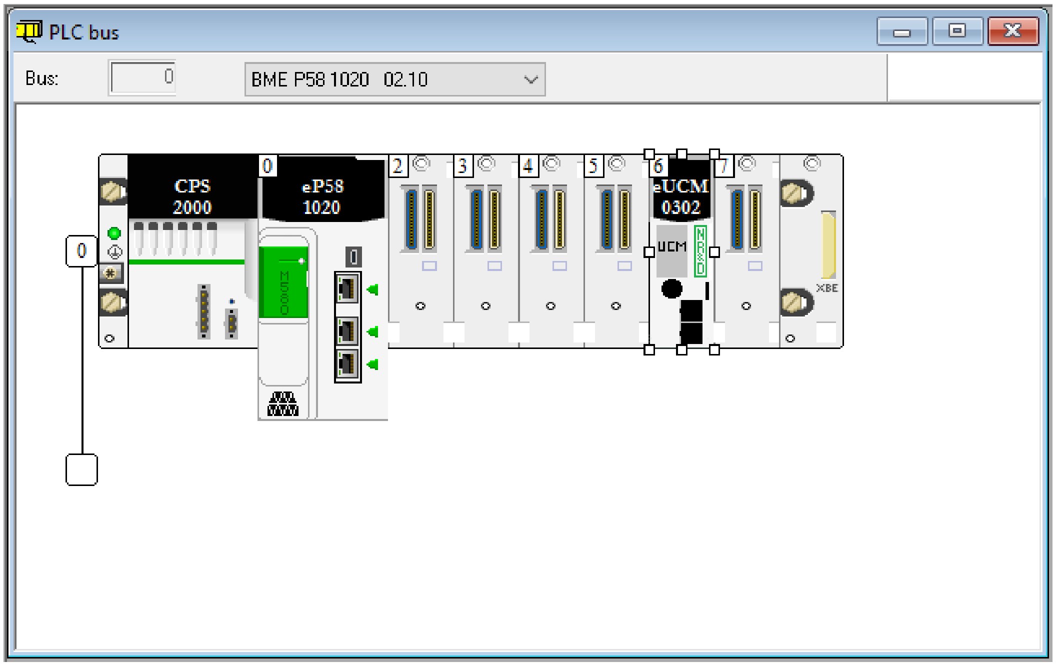
For this example, the UCM will be located in slot 6 of the local rack.

After right clicking on slot 6, a select “New Device”.



The PME UCM 0302 is located in the ‘Third Party products’ section. Select the UCM and click ‘OK’.



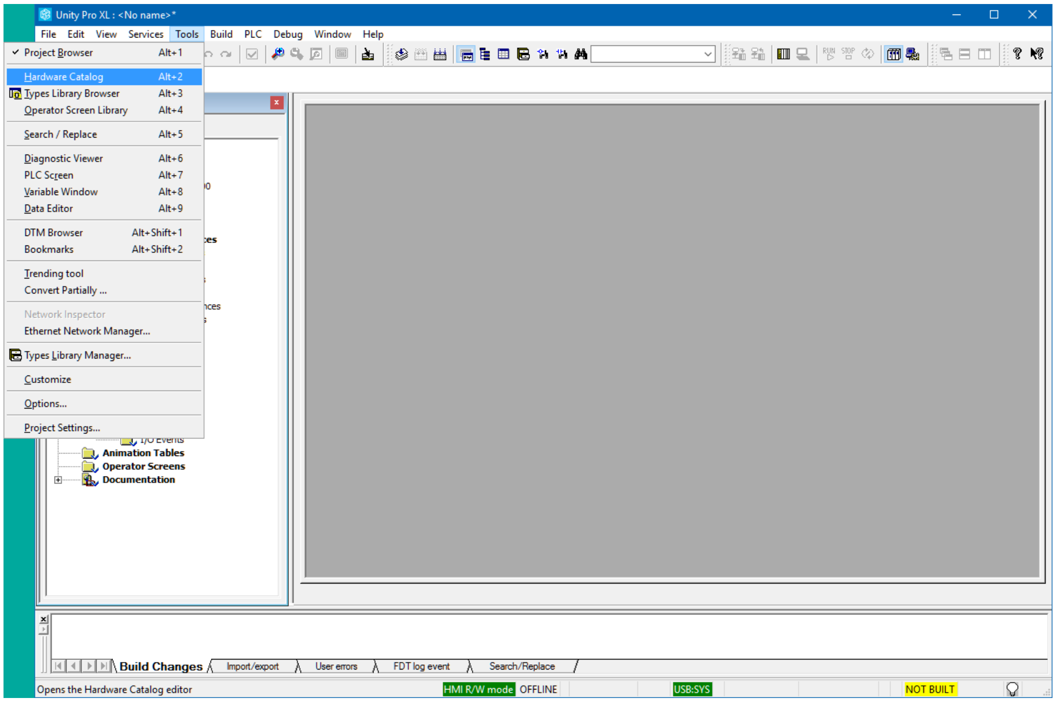


The UCM will now appear in the rack.

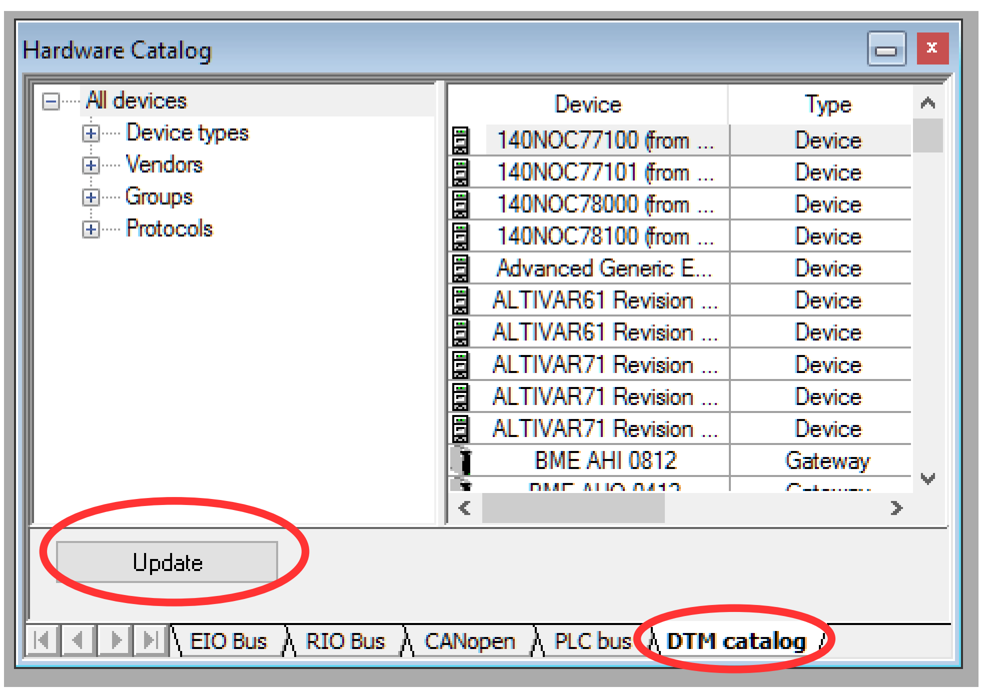
The PLC rack window may now be closed.

DTM Hardware Catalog Update

In chapter 4, Niobrara's DTM Utility was used to install the TCPOPEN DTM into Control Expert's database. The next step is to force an update of the DTM Catalog. The DTM Catalog is accessed through Tools > Hardware Catalog.



The Hardware Catalog Window should appear and look something like this:

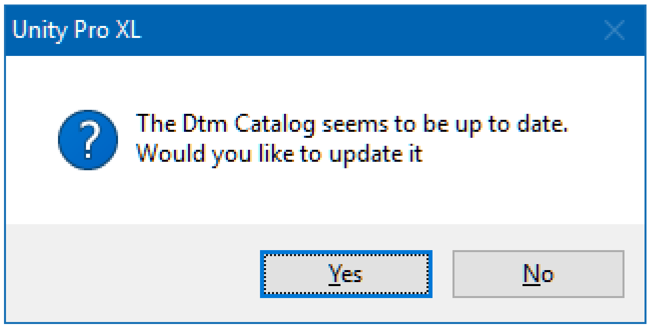


Click on the "DTM catalog" tab at the bottom.

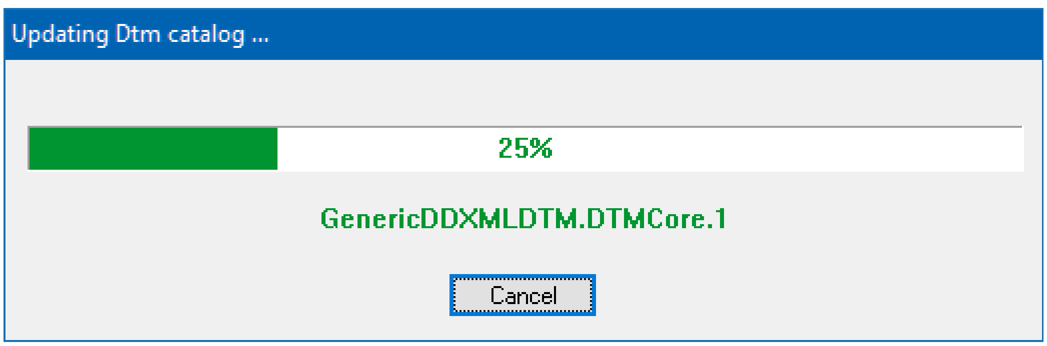
Then Click on the "Update" button.

A message box should pop up asking if it is ok to update the catalog. Select "Yes".

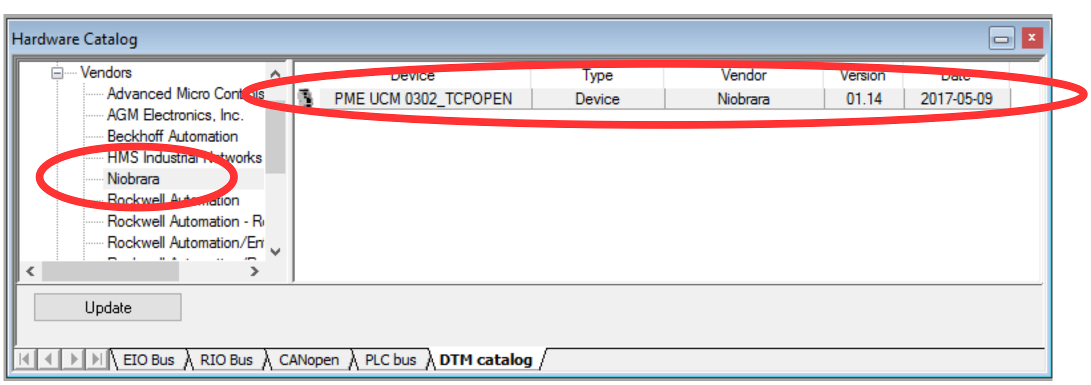
Note: This box opens every time the Update button is clicked.



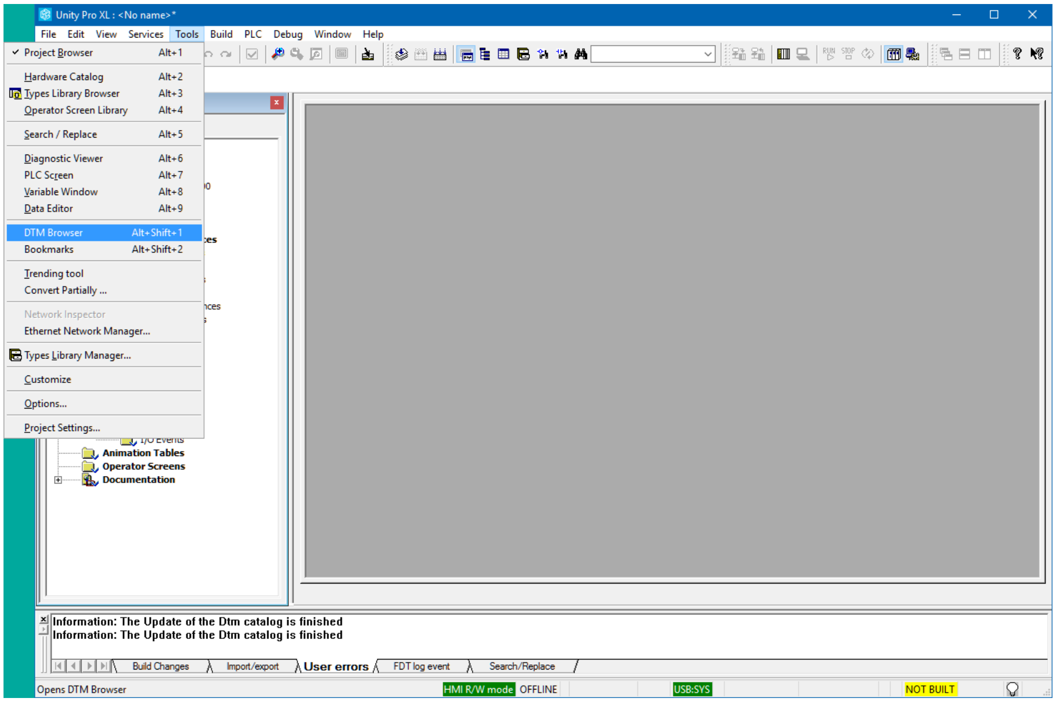
A progress window pops open.



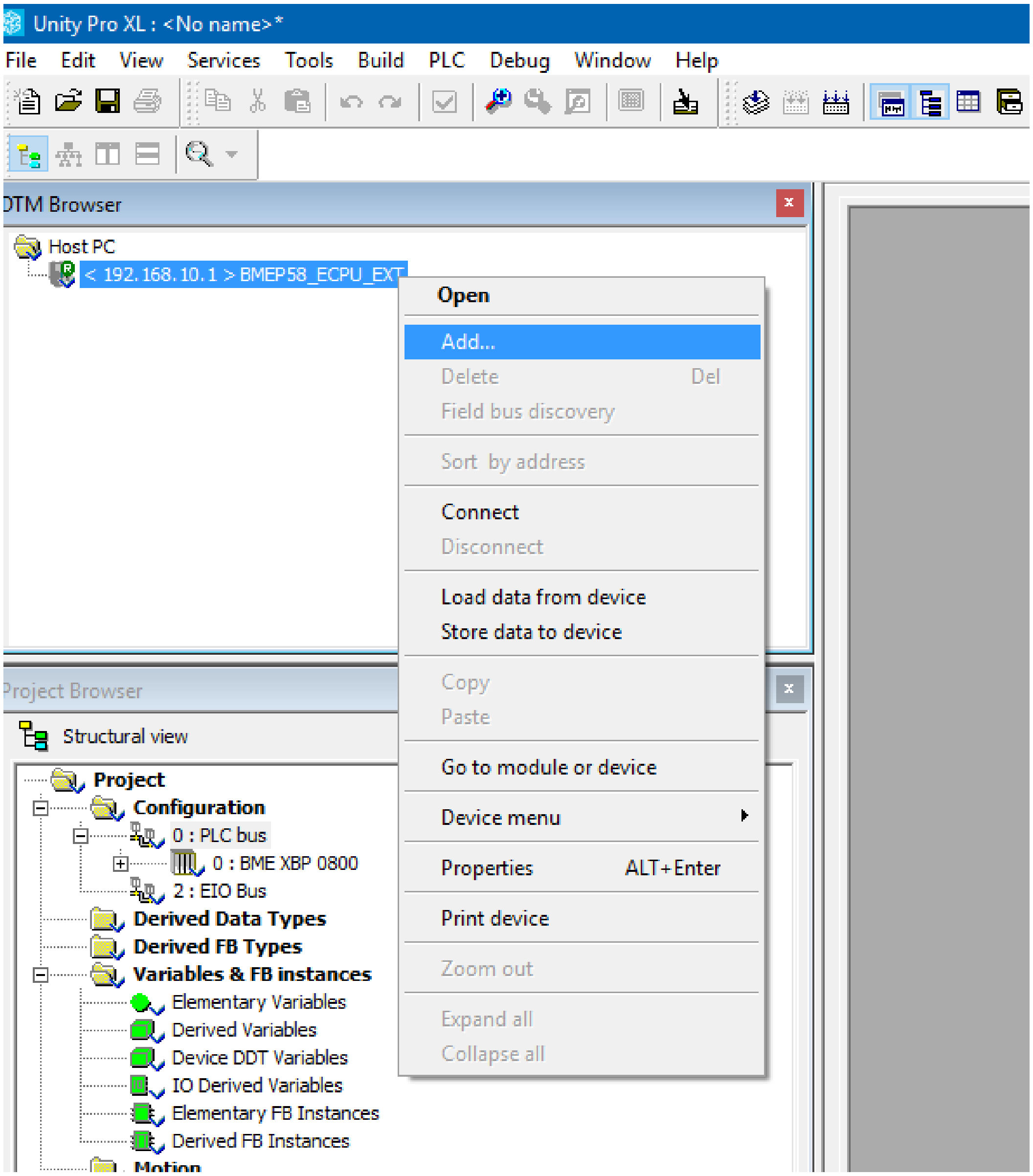
After the catalog update is complete, the new Niobrara DTM device should be listed in the hardware catalog. Also, the "User Errors" display should show "Information: The update of the DTM catalog is finished"



Now, Open the DTM browser by selecting Tools > DTM Browser.

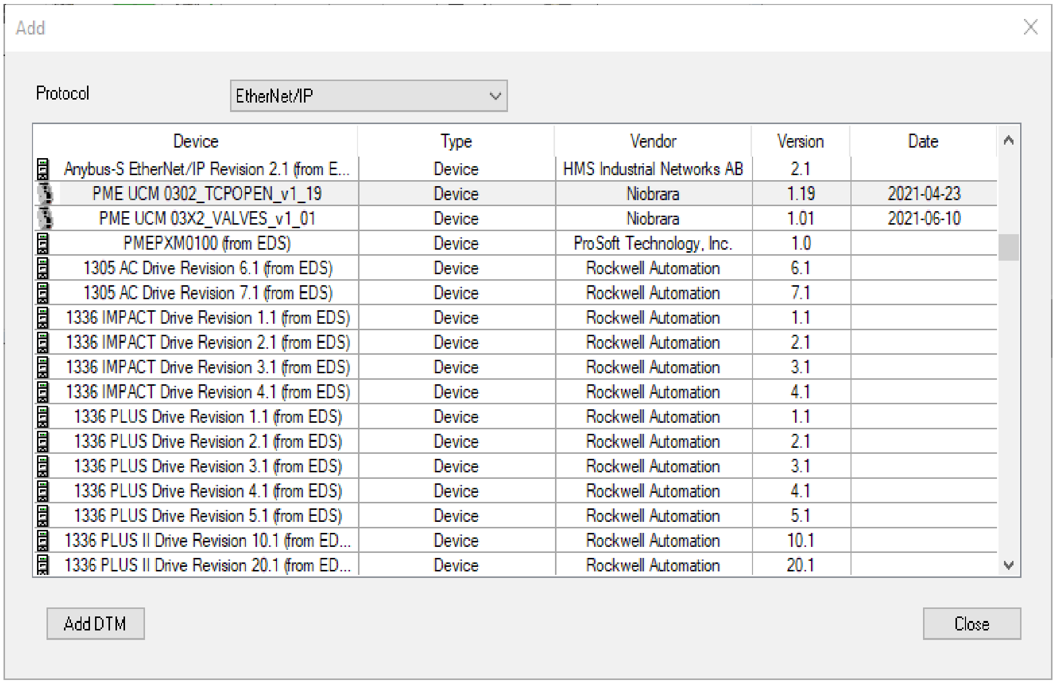


The DTM Browser will open and show a tree with the CPU at 192.168.10.1.
Right click on the CPU and select “Add”.



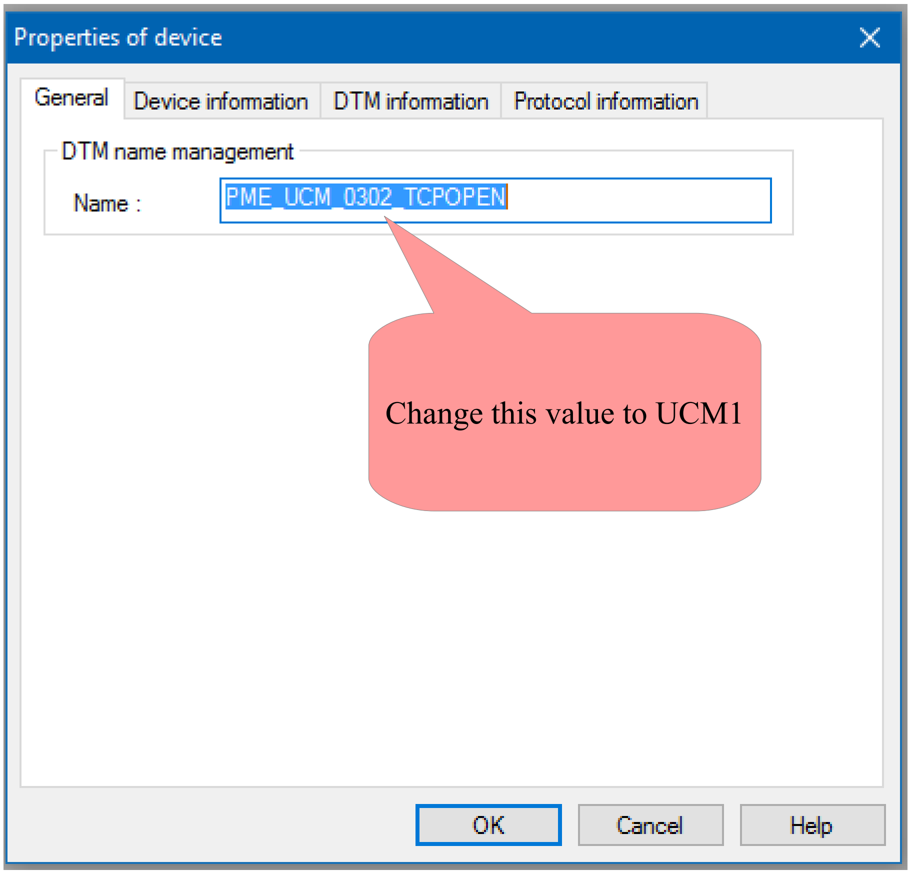
A window will pop up showing all of the installed DTMs. Scroll down until you reach the PME UCM 0302 TCPOPEN device by Niobrara.

Notice that it has the version 01.19 which matches the SW version in the txt file.

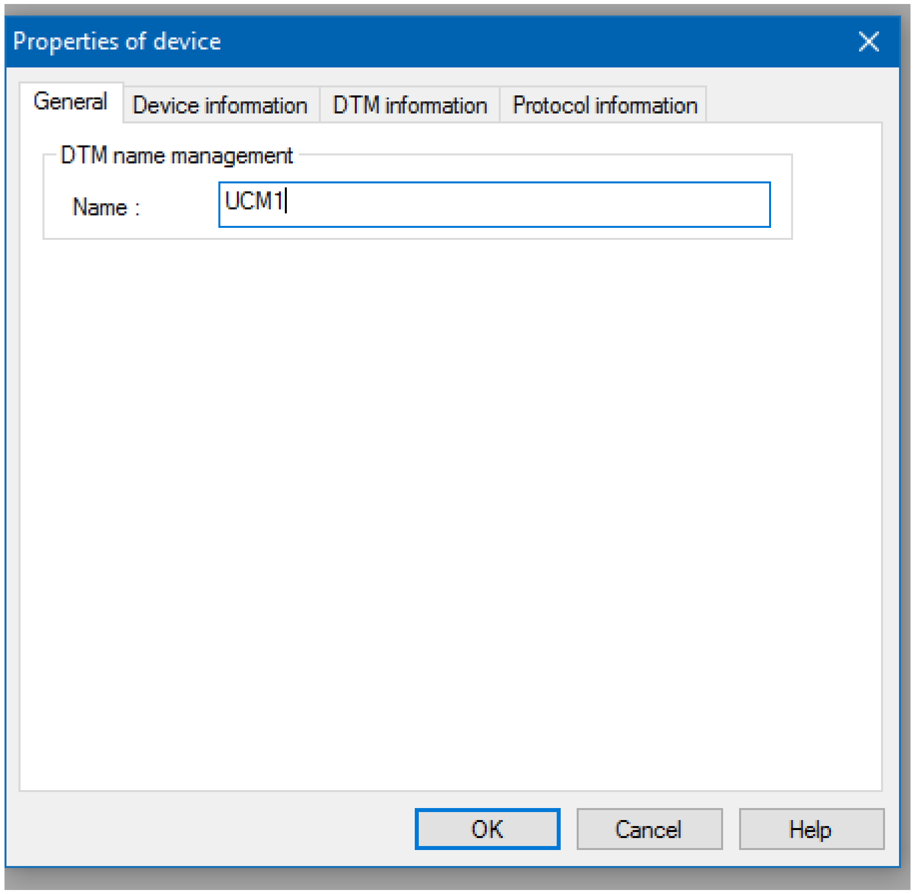


Press Enter or "Add DTM" to load the DTM for the PMEUCM. A window will pop up with information about the DTM.

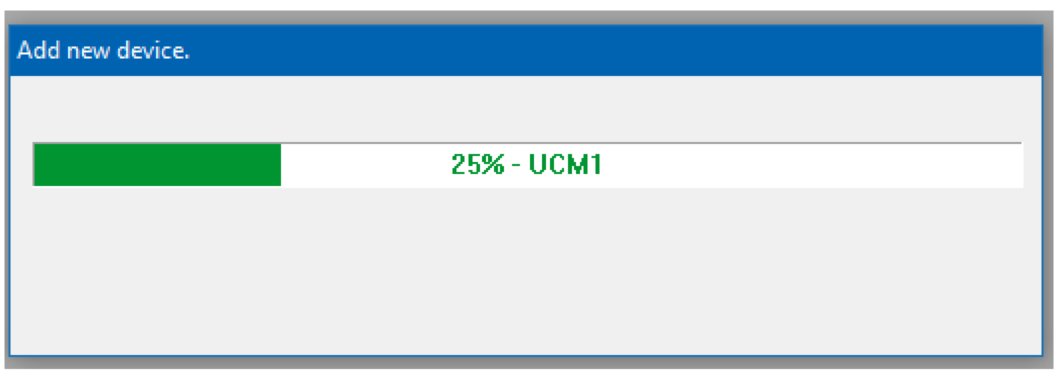
At this point the “Alias name” may be modified. The example ST and DFB code used later requires this “Alias name” to be set to ‘UCM1’.



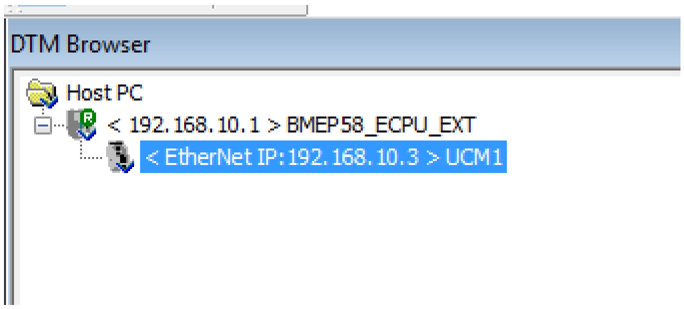
So change the name to ‘UCM1’.



Pressing “OK” will add the DTM device to the DTM Browser.



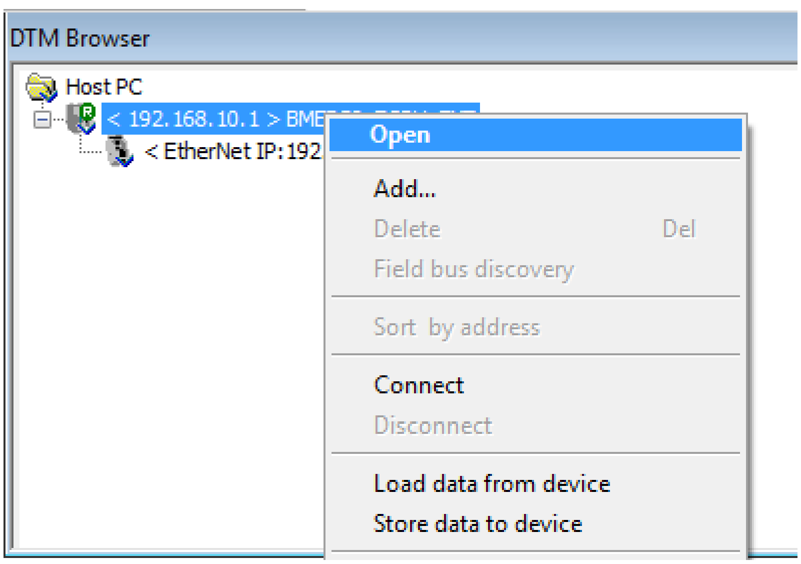
The PMEUCM is now added to the tree below the CPU.



Link the DTM to the PMEUCM Hardware

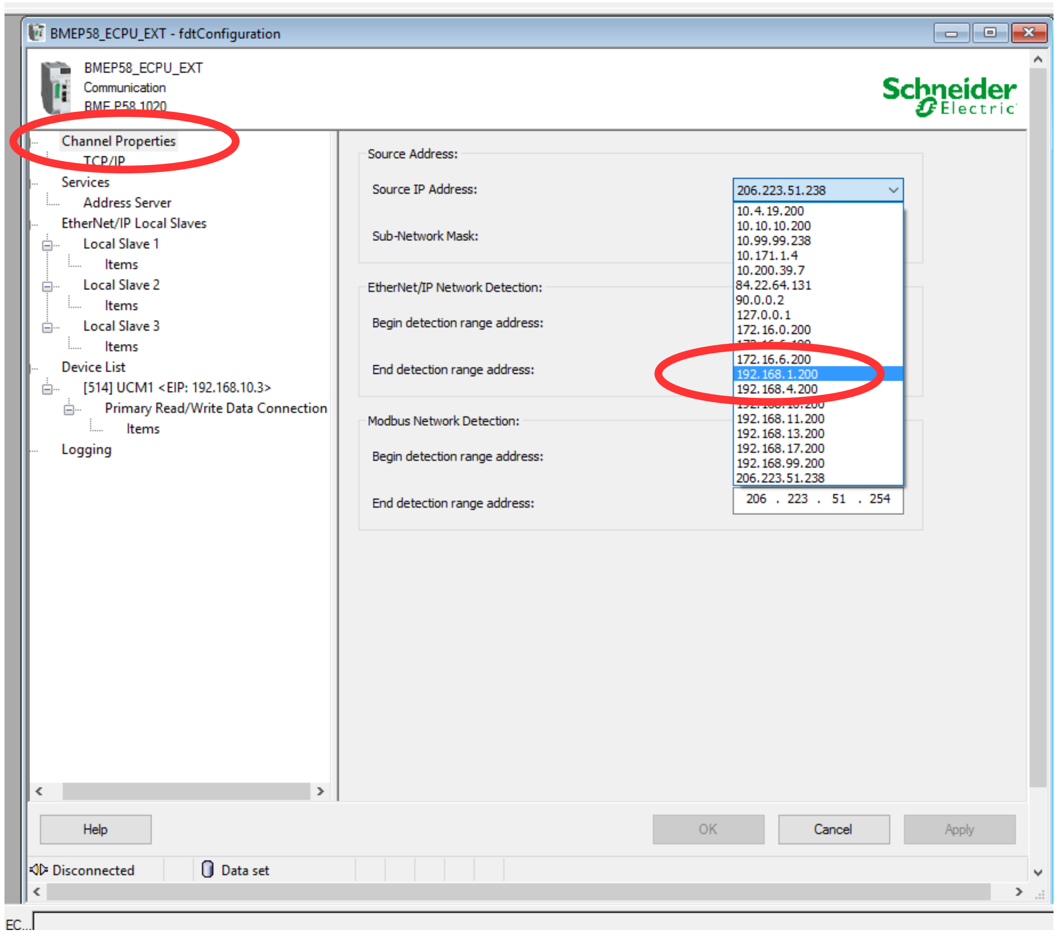
It is time to actually associate the DTM instance with the actual PMEUCM device. This is done inside the DTM Browser window.

Right click on the CPU and select Open.



NOTE: The "Source IP Address" is a pull-down listing of all of the IP Addresses of the Unity Pro PC. Make sure to select an address that is on the same subnet as the M580 PLC. In this case the IP Address of 192.168.10.200 is selected since the PLC is at 192.168.10.1.

NOTE: This list only shows the active IP Addresses for the Unity PC. The user must have an active Ethernet connection to proceed.



Now click on the PME_UCM_0302_… entry in the list on the left.

Select the "Address Setting" Tab.

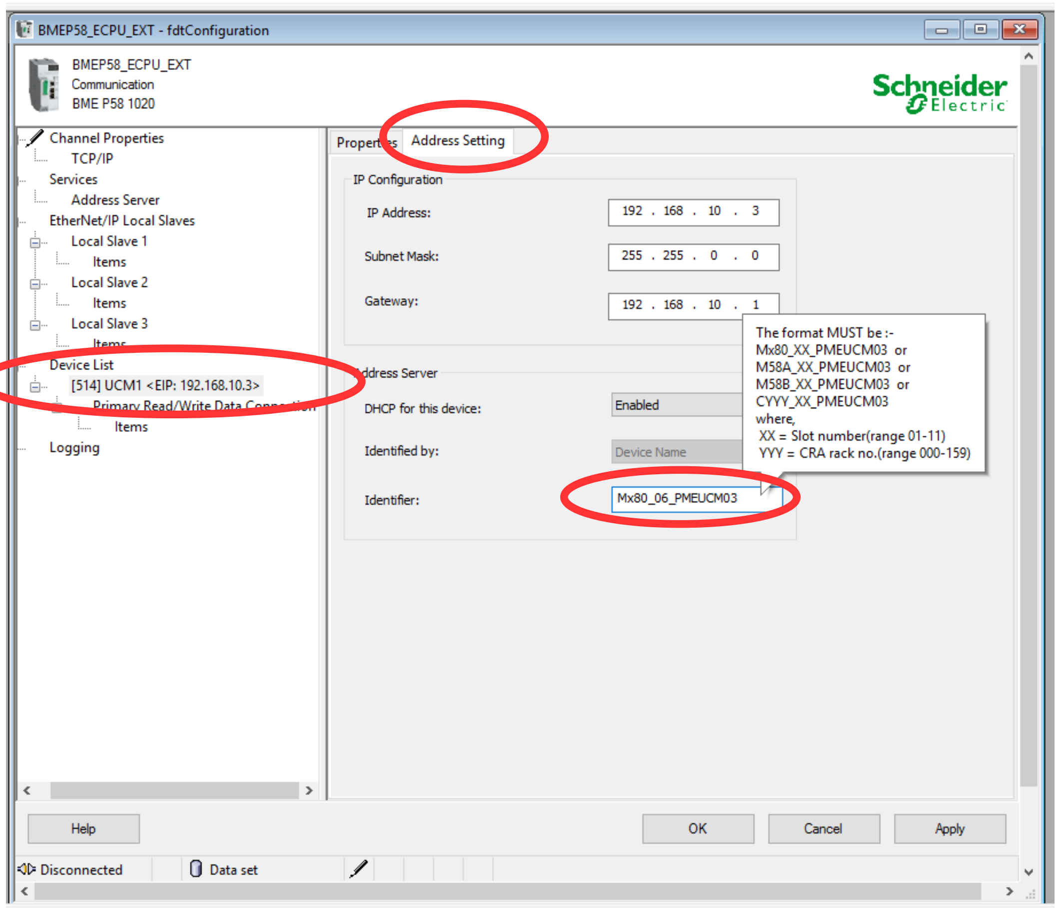
The "Identifier" must be modified to define the exact Rack and Slot occupied by the PMEUCM.

In this example, the PMEUCM is located in the CPU rack, Slot 6. Therefore, the Identifier must be set for "Mx80_06_PMEUCM03".

NOTE: If the PMEUCM is located in a remote rack, the YYY value is the thumbwheel (rotary switches) setting of the eCRA, not necessarily the logical rack number.

So, if the PMEUCM is in remote rack 1, slot 6, the Identifier would normally be C001_06_PMEUCM03.

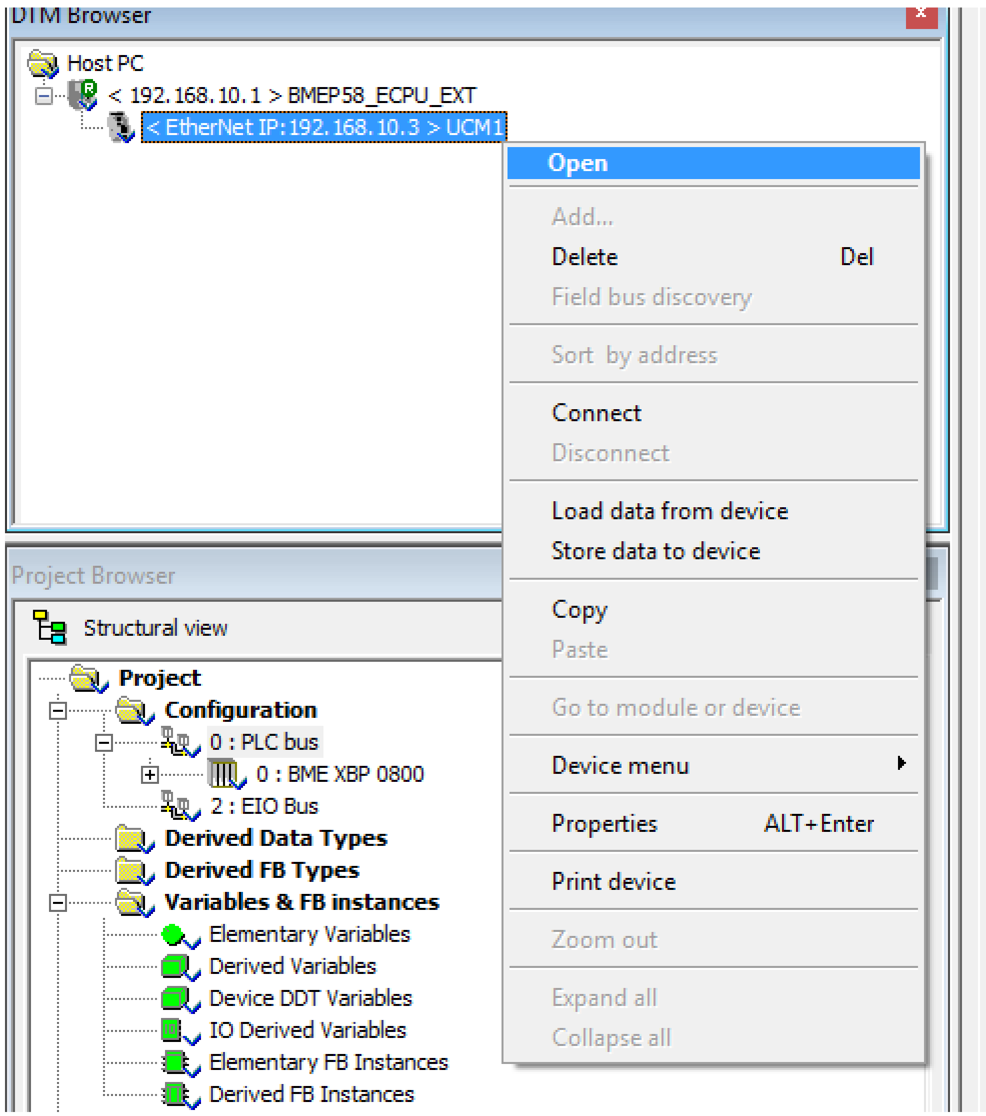
After setting the Identifier, click "Apply" to accept the settings and close the window.



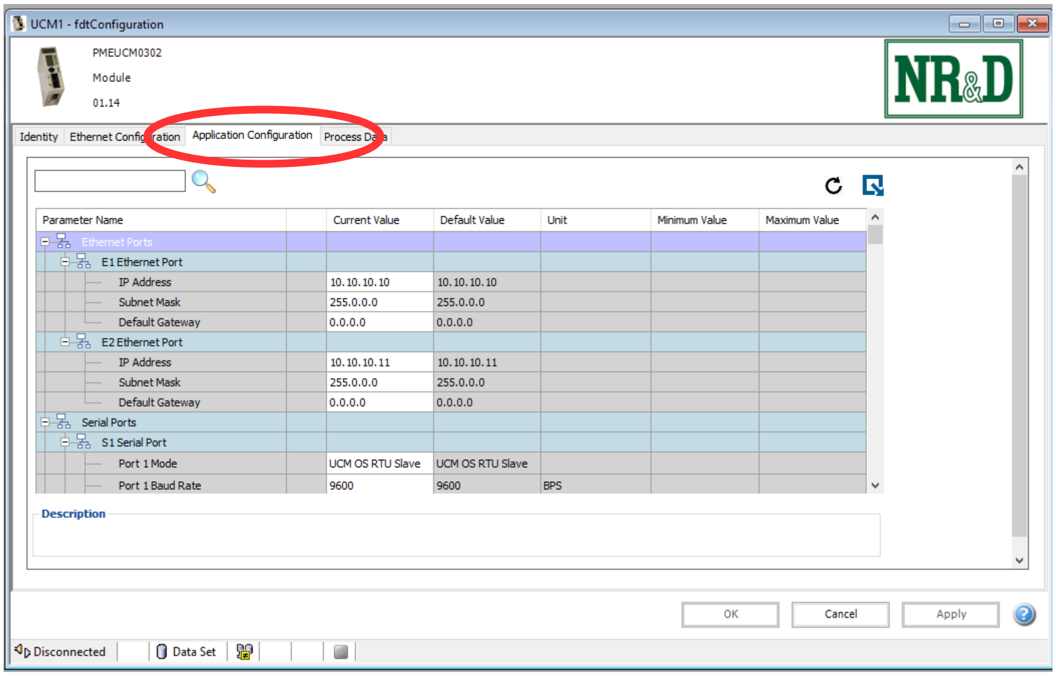
DTM Configuration

The PMEUCM must be configured through fields in the DTM screen.

Right click on the PMEUCM entry in the DTM Browser Tree and select “Open”



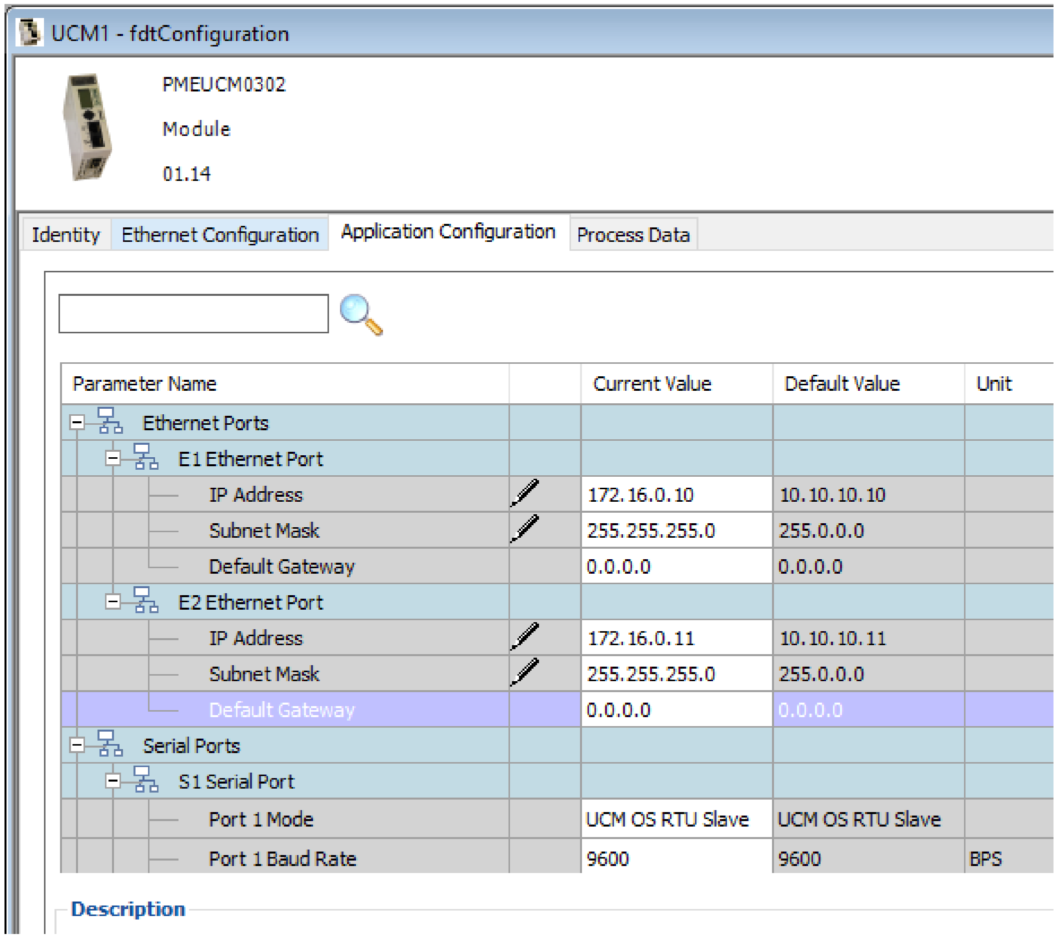
The DTM screen for the PMEUCM will open. Select the "Application Configuration" tab.



This tab shows the configuration settings for E1 and E2 ports, global OS settings, and S1 and S2 serial ports.

E1 and E2 Ethernet Port

Edit the strings for IP Address, Subnet Mask and Default Gateway for both E1 and E2. For this example, E1 will be set to 172.16.0.10 with a subnet mask of 255.255.255.0 while E2 is set to 172.16.0.11 and also with a subnet mask of 255.255.255.0.



UCM OS Settings

- OS Modbus/TCP Server Port – The PMEUCM Operating System has its own Modbus/TCP server used (used by QLOAD). Normally this is set to use TCP Port 503 to avoid situations where the M580 is serving on the standard Modbus/TCP port 502. Valid settings are:
  - Disabled
  - 502
  - 503 (Default)
- Debug Web Server Port – The TCPOPEN application has a built-in web server to assist in debugging an M580 application. Normally this value is set to use TCP port 81 to avoid situations where the M580 is serving on the standard port 80. Valid settings are:
  - Disabled

- 80
- 81 (default)

* Debug TELNET Server Port - The TCPOPEN application has a built-in TELNET server to assist in debugging an M580 application. Normally this value is set to use TCP port 24 to avoid situations where the M580 is serving on the standard port 23. Valid settings are:
  - Disabled
  - 23
  - 24 (default)
* OS Max TCP Segment Size – The UCM OS can have its maximum segment size adjusted for use in VPN applications. Valid settings are:
  - 1452 (default)
  - 750 (VPN)
* OS TCP Keep Alive Time – The number of seconds an idle socket waits before sending a Keep Alive to the remote end of the connection. Normally this value is set to 10 seconds to ensure unused sockets are closed quickly. Valid settings are:
  - 10 (default)
  - 30
  - 60
* Location Name – This is a 10 character (max) text string to name the UCM. This location name is shown on the front screen and web page.

S1 and S2 Serial Port

NOTE: The current release of TCPOPEN does not provide support for serial port operation. These configuration values are in the DTM for future use and are ignored by the module.

Parameter Name		Current Value	Default Value	Unit
S1 Serial Port				
Port 1 Mode		UCM OS RTU Slave	UCM OS RTU Slave	
Port 1 Baud Rate		9600	9600	BPS
Port 1 Parity		EVEN	EVEN	
Port 1 Data Bits		8	8	
Port 1 Stop Bits		1	1	
Terminator Characters		x0a	x0a	
Terminator based on time between ...		Disabled	Disabled	mS
S2 Serial Port				
Port 2 Mode		UCM OS RTU Slave	UCM OS RTU Slave	
Port 2 Baud Rate		9600	9600	Baud
Port 2 Parity		EVEN	EVEN	

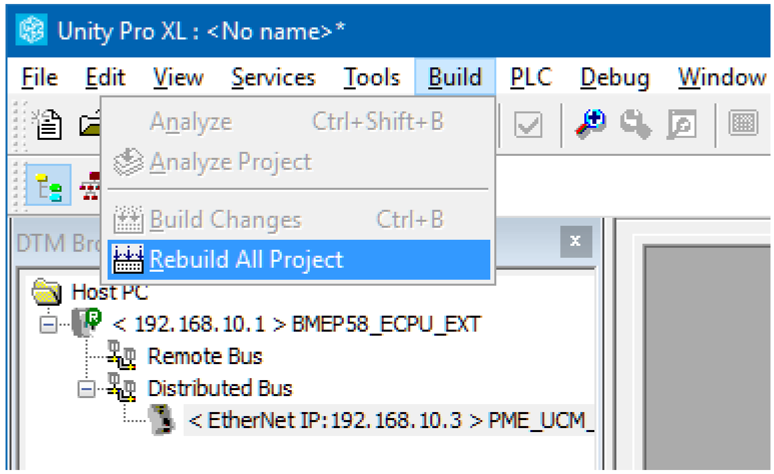
- Mode – The serial ports may be set to several different operating modes.
  - UCM OS RTU Slave – (default) This mode allows the operating system to control the serial port and respond to Modbus RTU messages.
  - Modbus RTU Framing – This mode configures the port to be either a Modbus RTU Master or Slave. The port is treated like TCPOPEN mode but the message data is prepended with a word of length of the RTU frame. The serial port automatically calculates and adds the CRC16 checksum on transmitted messages. It also automatically calculates and verifies and removes the CRC16 checksum on received messages. The port also automatically terminates received messages based on 3.5 character times of intercharacter timeout.
  - TCPOPEN – This mode allows the serial port to be treated like a UDP socket. The terminating characters and Termination based on intercharacter timeout are used to determine the end of a frame.
- Baud Rate – bit rate for the serial port
  - 1200
  - 2400
  - 4800
  - 9600 (default)
  - 19200
  - 38400
- Parity
  - NONE
  - ODD

    - EVEN (default)
- Data Bits
    - 7
    - 8 (default)
- Stop Bits
    - 1 (default)
    - 2
- Terminating Characters – This is a list of hexadecimal values preceded by a lower case 'x' that is used by the port to determine the end of a packet when the port is set to TCPOPEN mode. Multiple values may be added by placing a comma between fields. Some Examples:
    - x0a – Line Feed (default)
    - x0d – Carriage Return
    - x03 – ETX
    - empty – termination by specific characters is disabled
- Terminator based on time between characters – This value determines how long the UCM will wait on characters (without a termination character) to be received before sending the message to the M580. Allowed values are:
    - Disabled (default)
    - 10 mS
    - 50 mS
    - 100 mS
    - 200 mS

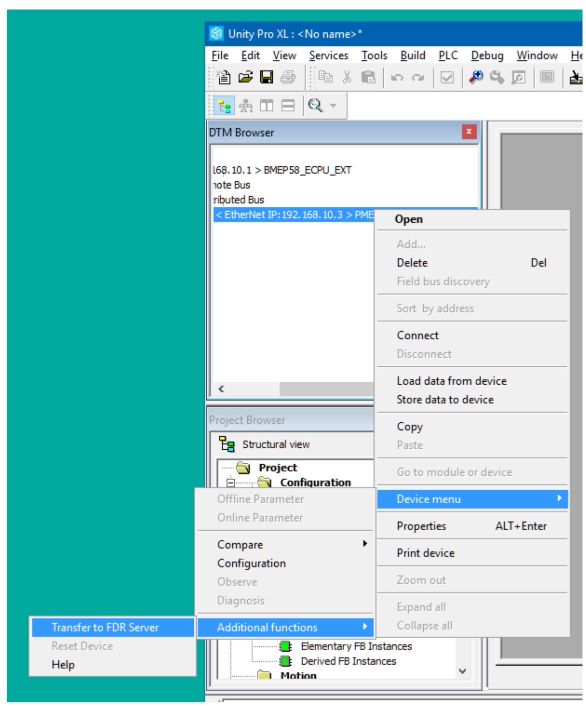
Applying and Installing Changes to the DTM

Any time one of these settings is adjusted, the following procedure must be followed:

1. After finishing the adjustments to the settings, click “OK” or “Apply” and then “Cancel” to close the DTM window.
2. Do a “Build”, “Build Changes” or “Rebuild all Project”.



3. Transfer this new prm file to the FDR server. Right click on the PME_UCM entry in the DTM tree and select "Device Menu" > "Additional Functions" > "Transfer to FDR Server". This action causes the PME Generic DTM dll to build a new prm file and send it to the FDR server.



4. The easiest thing to do now is to reboot the PMEUCM card to get it to read the PRM file from the FDR server.
    1. From the UCM front panel screen, select:
        1. 'Menu'
        2. 'System'

   3. 'Reboot' and the module should reboot.
2. Or, cycle power on the rack containing the PMEUCM.

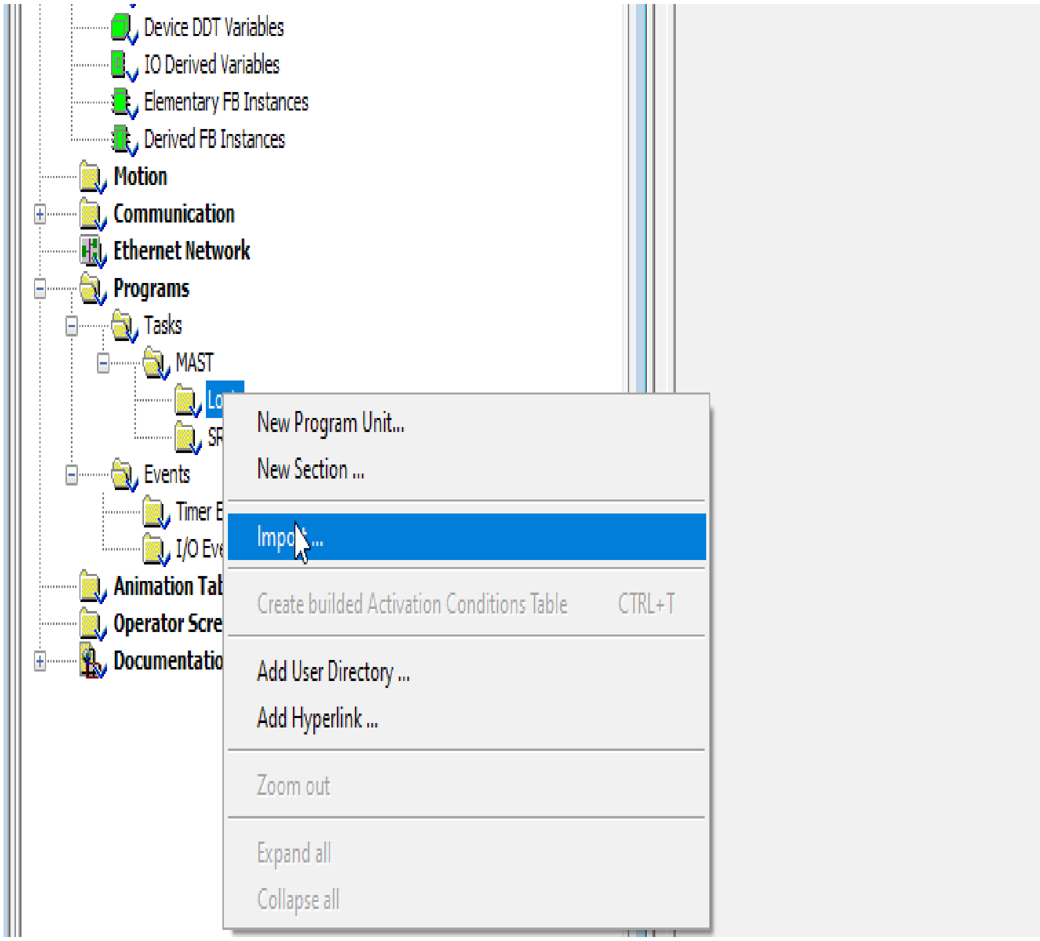
After the module boots and establishes a connection with the M580 CPU across the backplane, the new settings will be applied to the application.

If the IP Address for either E1 or E2 has been changed or if any of the OS IP parameters (OS port, TCP keep alive, etc.) have changed, the PME will make these changes, save them to EEPROM, and reboot once.

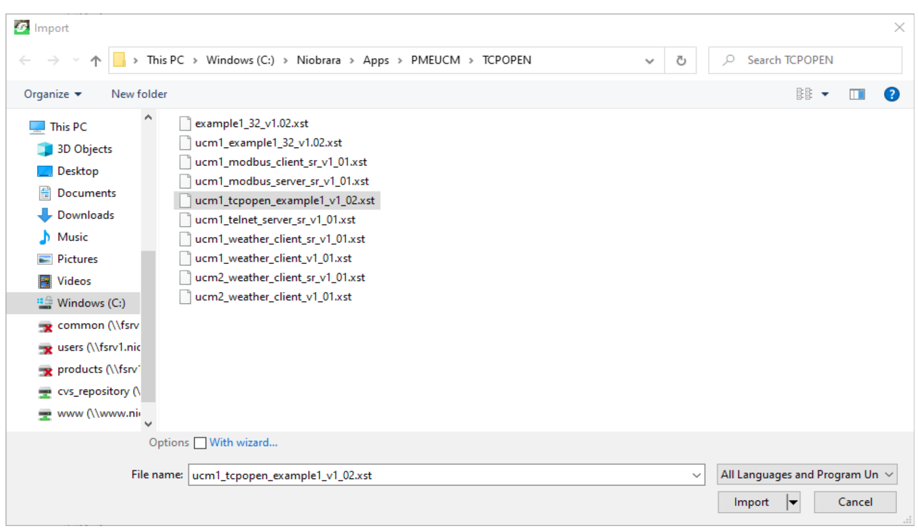
Import FBD Code and Sample Logic

The TCPOPEN application requires two DFBs to be installed in the Unity Project. The two sections are included in the PMEUCM_TCPOPEN_Setup. The very easiest method to get both of these DFBs (as well as some example logic) is to import the TCPOPEN example. It contains both DFBs, and some ST logic that employs them both in the proper manner.

Open the Project Browser. Expand the "Program" tree, and expand "Tasks." Expand "MAST," right-click on "Logic" and select "Import..."

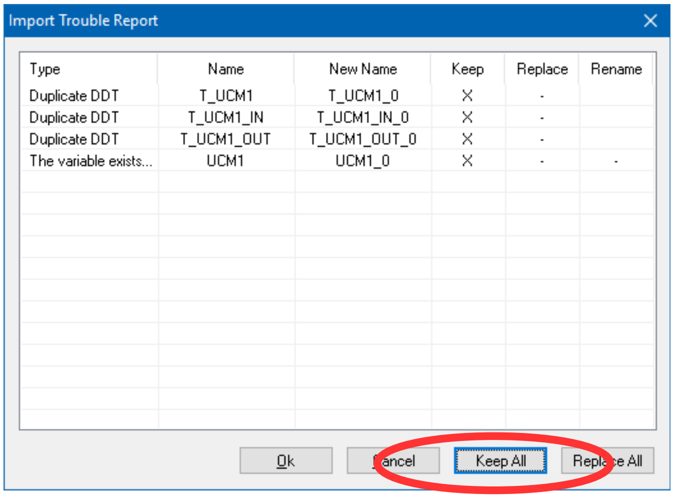


Browse to the “C:\Niobara\apps\PMEUCM\TCPOPEN\” folder. There should be a list of .xst files.



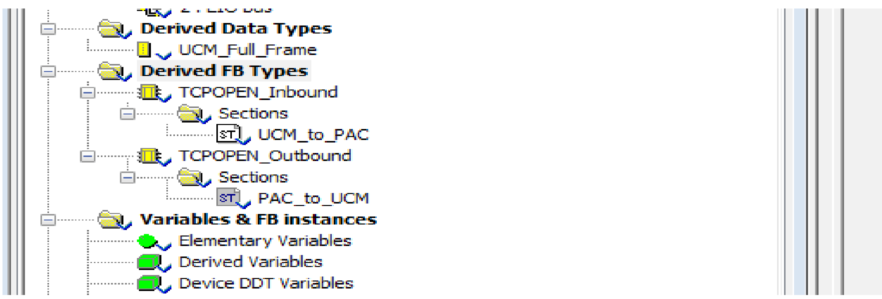
Select ‘ucm1_tcpopen_example1_v1_02.xst’ and press ‘Import’.

Unity Pro should prompt with a message box:



Select ‘Keep All’ and then ‘Ok’.

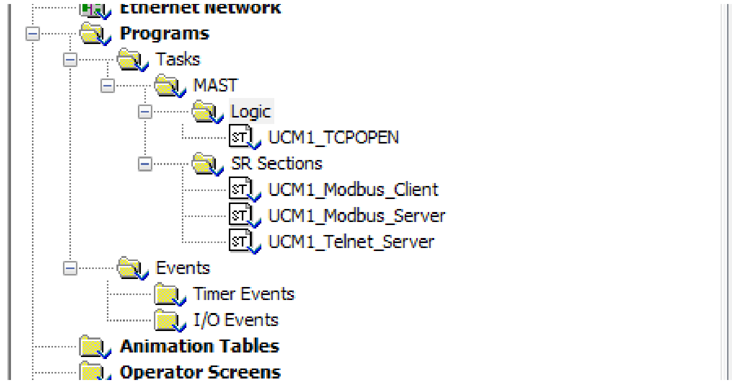
Two new Derived FB will be added:



Communications ST sections

Along with the two Derived Fbs, there was also code that will actually do something with sockets!

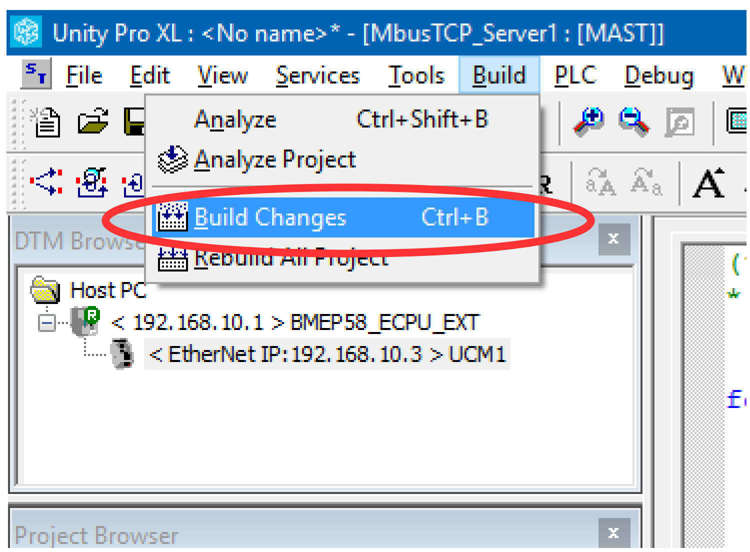
There should now be a new MAST section called ‘UCM1_TCPOPEN’ and three subroutines: UCM1_Modbus_Client, UCM1_Modbus_Server, and UCM1_Telnet_Server.



Build All or Build Changes

After importing the wanted segments, it is time to do a 'Build All' or 'Build Changes' for the Project.

Select "Build > Rebuild All Project" or "Build" > "Build Changes"



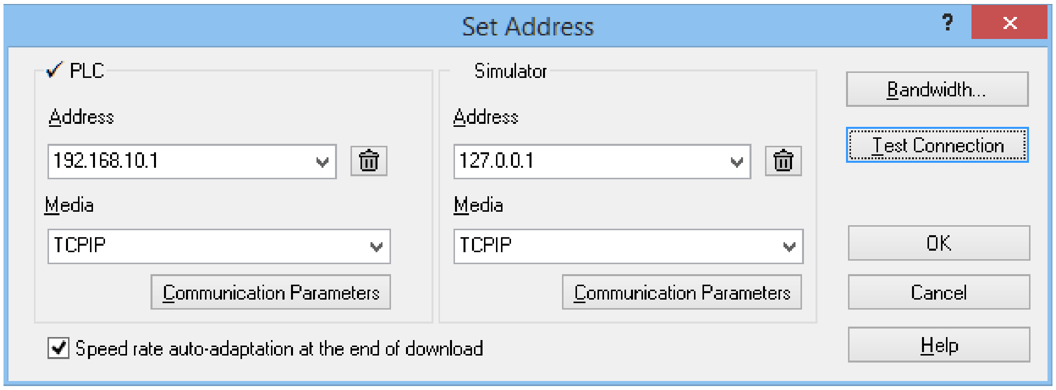
Transfer Project to PLC

After a successful Build, it is time to transfer the project to the M580. This may be done through USB or over Ethernet. Since an Ethernet port is required to transfer the DTM PRM file to the FDR server, the Ethernet connection will be shown.

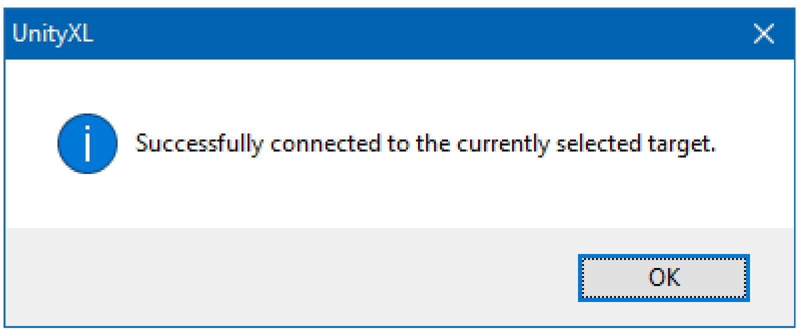
Connect the Ethernet port of the PC to the Service Port of the M580.

PLC Set Address

Select PLC > Set Address and choose TCPIP for the Media and set the Address of the M580 (192.168.10.1).



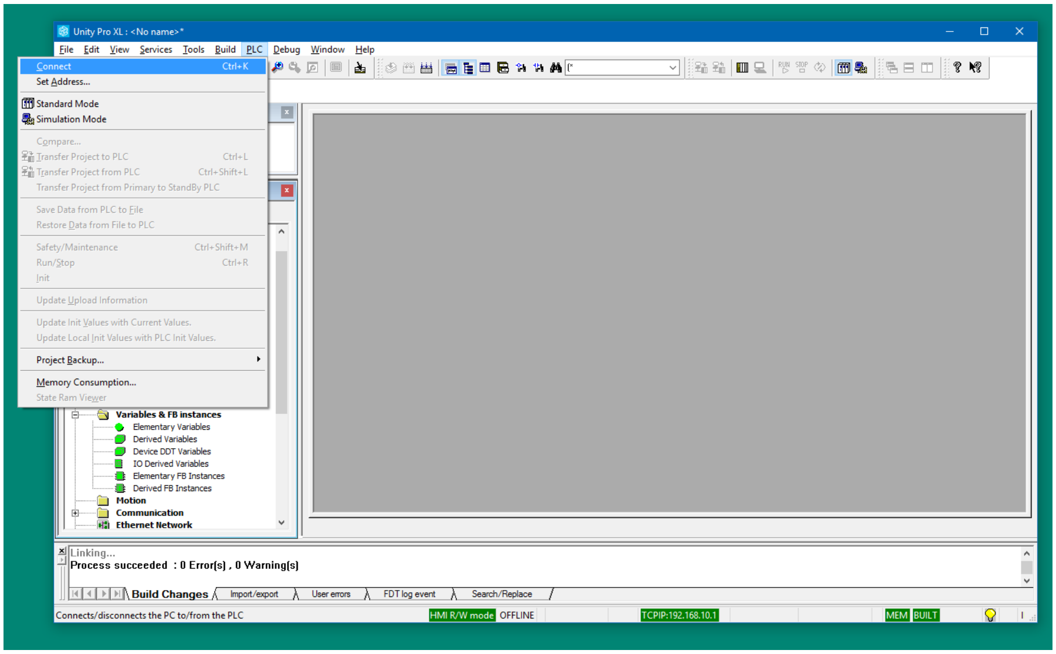
It is usually a good idea to try the "Test Connection" button to make sure that the PC can connect with the M580.



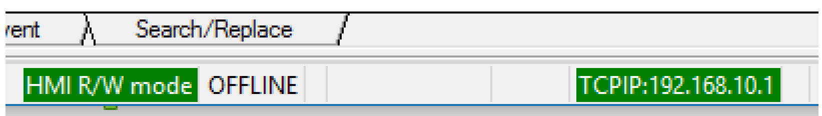
If successful, press the OK button to close the 'Set Address' window.

PLC Connect

Now select PLC > Connect to open a connection to the M580 CPU.



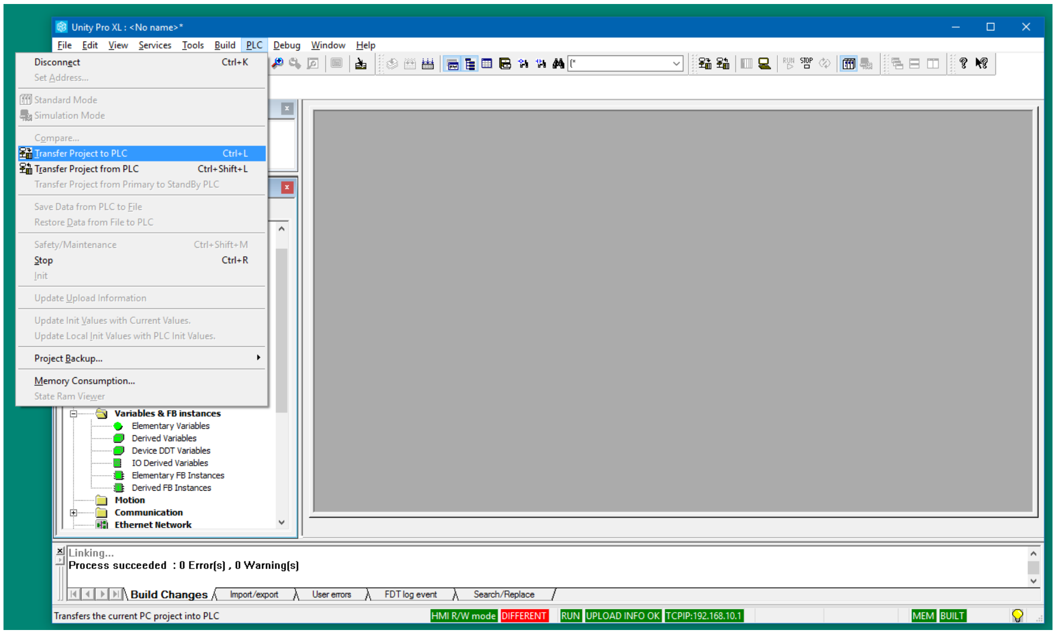
The bottom display will change from ‘OFFLINE’ to ‘ONLINE’. It should also show ‘DIFFERENT’ to indicate that the PLC is not the same as Control Expert.

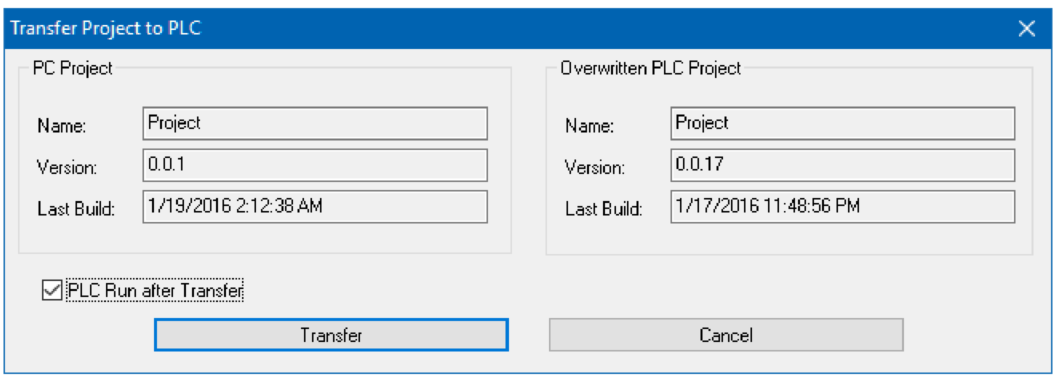


Transfer Project to PLC

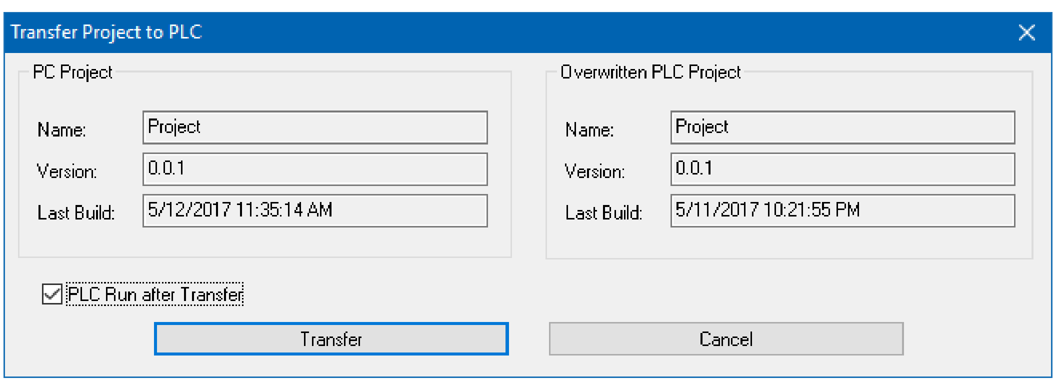
After connecting, transfer the project to the PLC.

The Transfer Project to PLC window should look something like this:



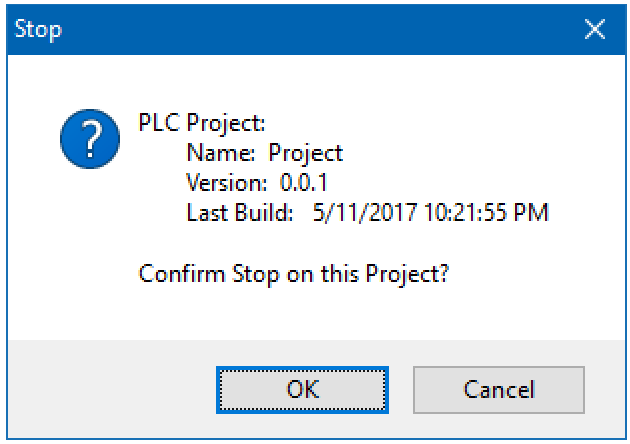


It is usually convenient to check the PLC Run after Transfer box.

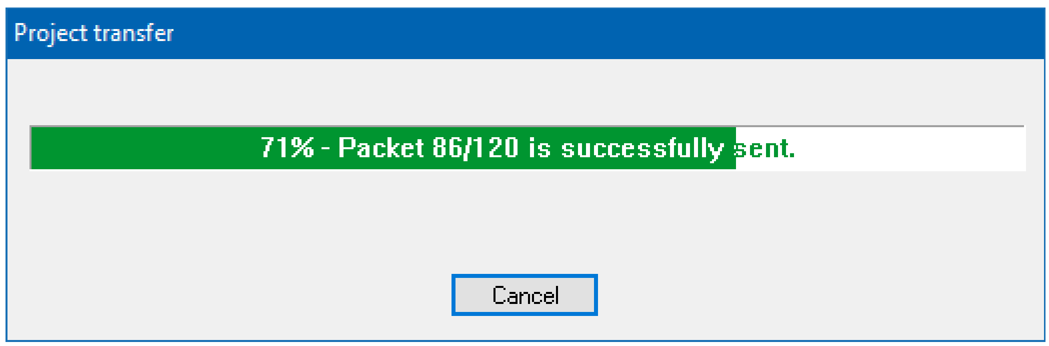


If the PLC is in RUN, you will be prompted to Stop the M580.

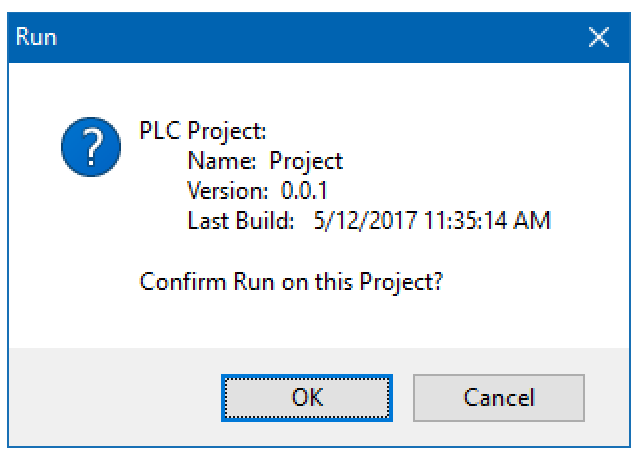
WARNING: Stopping a running PLC may result in injury or death. Make sure that you understand the consequences of halting a running program.



The transfer should look like this:



The Run confirmation screen will be shown if the "Run after Transfer" was selected.



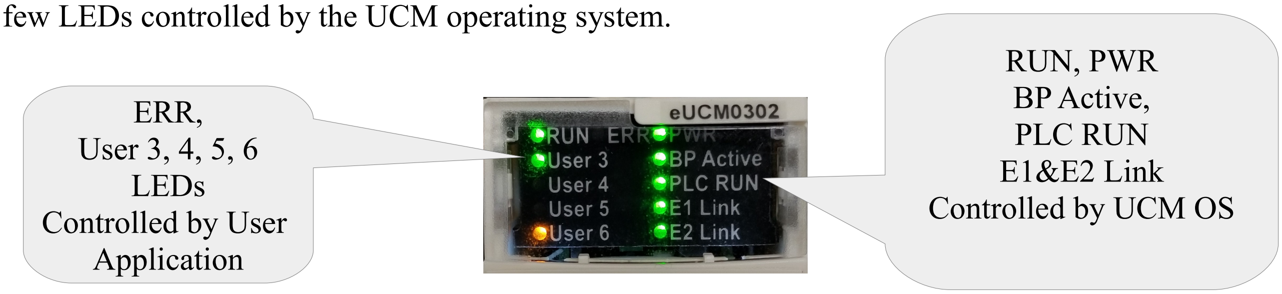
Selecting "OK" will start the PLC.

The UCM should make some beeping sounds as the PLC transitions from STOP > RUN and RUN>STOP. It also beeps when booted, and when the setup is complete and the application is properly communicating with the M580 DIO.

8 Front Panel Operation

LED Panel

Most of the LED indicators on the top panel are controlled by the user application with a few LEDs controlled by the UCM operating system.



Top Panel Lights

The meaning of these lights is described in the following table.

Label	Color	Description
RUN	Green	ON – The TCPOPEN application is running. NOTE: This is NOT an indication of the run/halt state of the PLC.
ERR	Red	ON – There is a configuration issue. The front panel LCD will contain more information about the error condition.
PWR	Green	ON - The PMEUCM has proper 24Vdc power from the Ethernet backplane.
User 3	Green	The DTM configuration from the Primary CPU is correct.
User 4	Amber	The DTM configuration is not verified.
User 5	Green	The PRM file in the HSBY Secondary FDR server is being checked.
User 6	Amber	The PRM file in the HSBY Secondary FDR server is not verified.
BP Active	Green	DIO Operation with the Primary CPU is active.

PLC Run	Green	M580 CPU is in RUN and DIO is active.
E1 Link	Green	Ethernet port E1 has active LINK.
E2 Link	Green	Ethernet port E2 has active LINK.

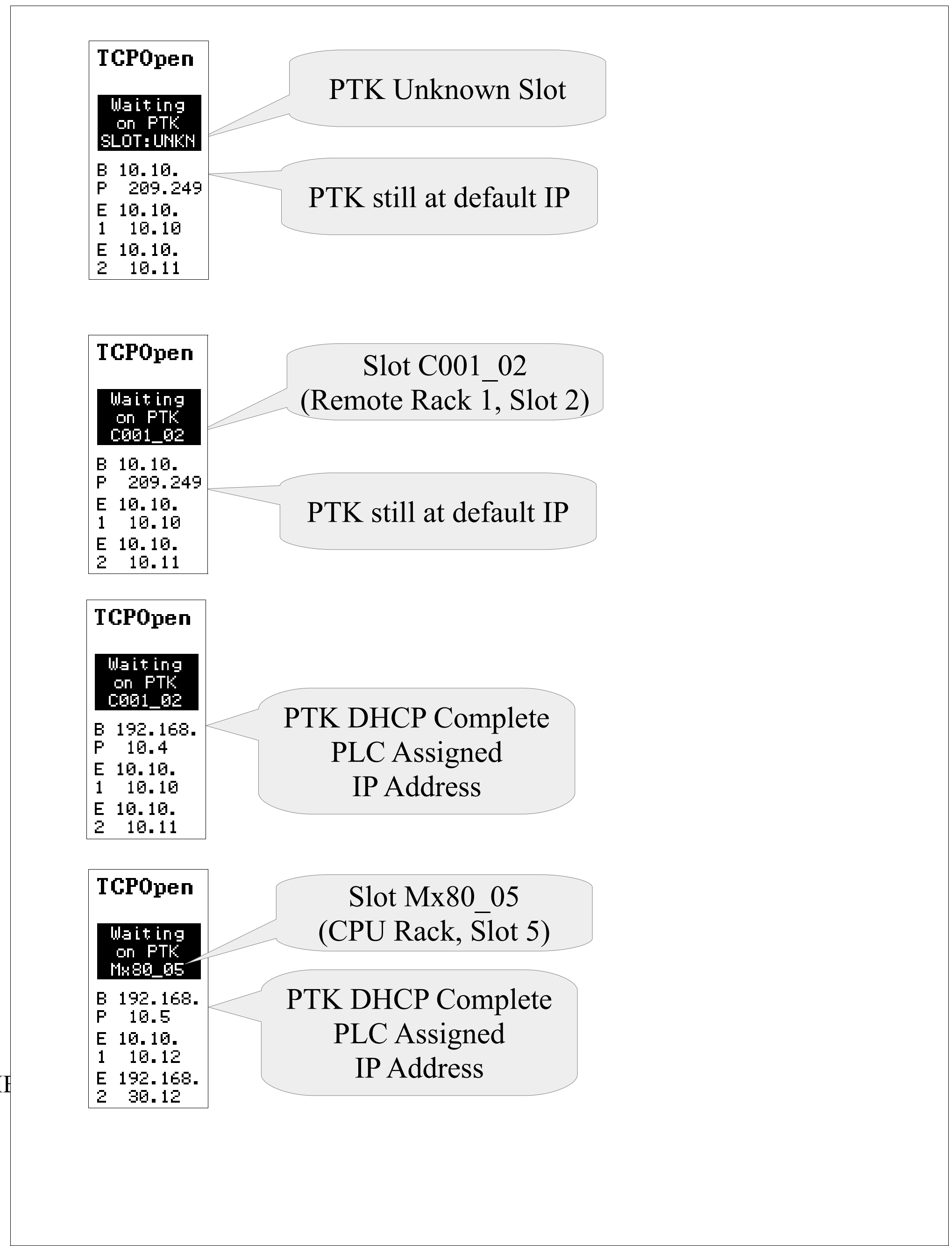
Additionally, there are two RED lights behind the LCD. One red light will only be on when the backlight is OFF to help visually indicate a problem.

The second red light is controlled by UCM_COMMAND.3 from the PLC code.

LCD and Joystick Operation

The front panel LCD provides status information about the PMEUCM and user interaction with the setup and operation of the card/application.

The information displayed on the "splash" screen varies depending on the



configuration and state of the module.

Fault Indication

If the application is in a fault condition, the screen will show the fault in an inverted text box. The PTK IP Address is shown as BP (backplane). The E1 and E2 IP Addresses are also shown.

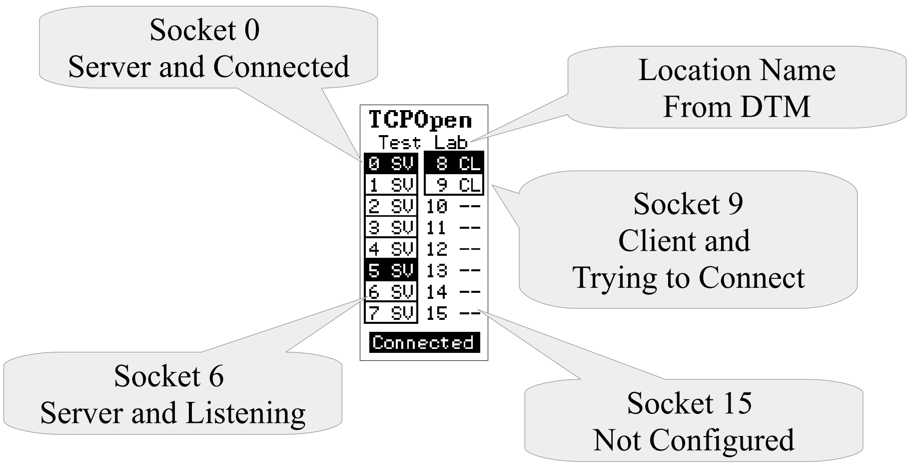
Many fault display screens change between multiple views showing the found condition and the required condition.

Fault	Condition	Solution
Duplicate IP Address	E1 or E2 in Conflict with another device	Change IP Address
PTK FW Too Old	PTK FW needs updating	Update PTK with Unity Loader
Waiting on PTK slot unknown BP = 0.0.0.0	Unit just booted	Wait for PTK to establish comms with M580
Waiting on PTK slot unknown BP = 10.10.x.y	Unit up for a while, BP still at factory IP Address	Check rack addressing in PLC
Waiting on PTK slot known BP = 192.168.0.x	PTK unable to load prm file	Wait, Check “Identifier” DTM setting, or “Transfer to FDR server”
Wrong DTM File	DTM File name must be 'PME UCM 0302_TCPOPEN'	Install correct DTM in DTM browser
Wrong DTM Version	DTM version must match application requirement	Update installed DTM with DTM Utility
Wrong DTM Byte Count	DTM configuration is wrong	Contact Niobrara Tech Support
Wrong Util Version	Nrdptkddxmlutil.exe too old	Install new version and update DTM
Wrong DLL Version	Generic PME DTM too old	Install new Generic PME DTM to Unity

Normal Operation

The screen shows an overview of the 16 possible socket connections.

The screen shows all 16 sockets. SV = Server, CL= Client, --- = Not Config
Frame indicates listening (Server) or attempting to connect (Client)
Inverted box indicates socket is connected.



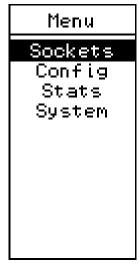
Backlight

The backlight time is controlled by user code. In this case there is a timer that keeps the backlight on for 180000 mS (3 minutes) when there is no activity of the joystick. At the end of this timer, the UCM code changes the screen back to the splash screen.

Menus

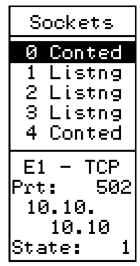
Moving the joystick will cause the application to show various menus to access status or setup screens. Move the highlighted cursor around with the joystick. Typically a right press will act as “Enter” while a left press will act as “Escape”. Sometimes a push in “Enter” is needed (Factory Default for example).

Menu Screen



Moving the joystick to the right while on the splash screen shows the main menu screen. Move the joystick up and down to highlight the items and to the right or push in for Enter to select. Move left to exit.

Sockets Screens



These data screens allow quick viewing of the online/offline status of the 16 possible sockets. The top part of the screen shows four of the sockets with the left column the entry number (0-15) and the right column the status of the connection.

Possible states are:

- “Conted” = Connected
- “Listng” = Server Listening
- “Trying” = Client trying to connect
- “Idle” = Idle, not active

The bottom portion of the screen shows:

- Target UCM Ethernet port E1 or E2
- TCP or UDP
- local TCP port number for servers or the remote port for clients
- IP Address of the remote device
- State condition of the PLC program

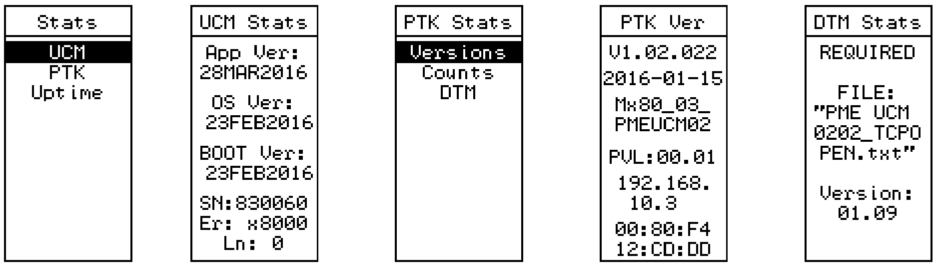
Config Menu

The Config menu shows the current IP Address of the PTK card and allows the modification of the UCM's E1 and E2 IP Address.

NOTE: The UCM's E1 and E2 may only be changed if the backplane interface is inactive.

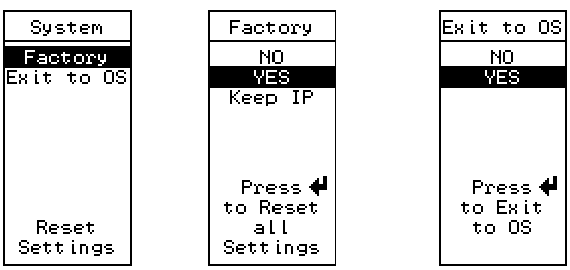
Stats Menu

Statistical counters are provided for a variety of fields.



System Menu

The system items allow the user to reset the application to factory default settings or exit to the UCM operating system.

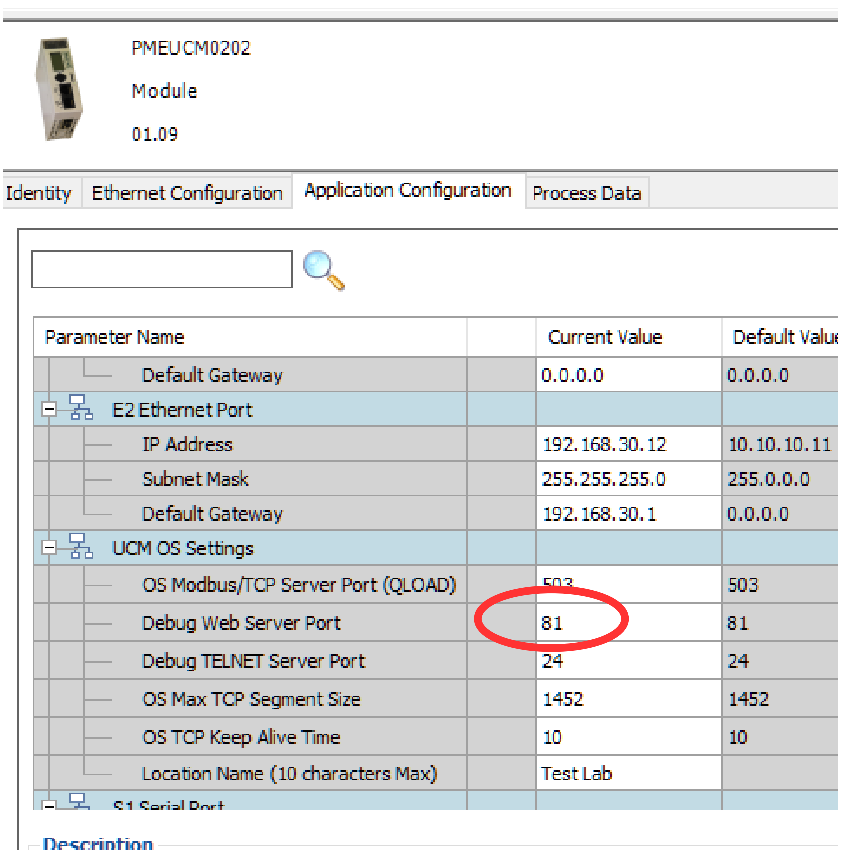


Press Enter means to push in on the joystick.

9 Debug Web Server

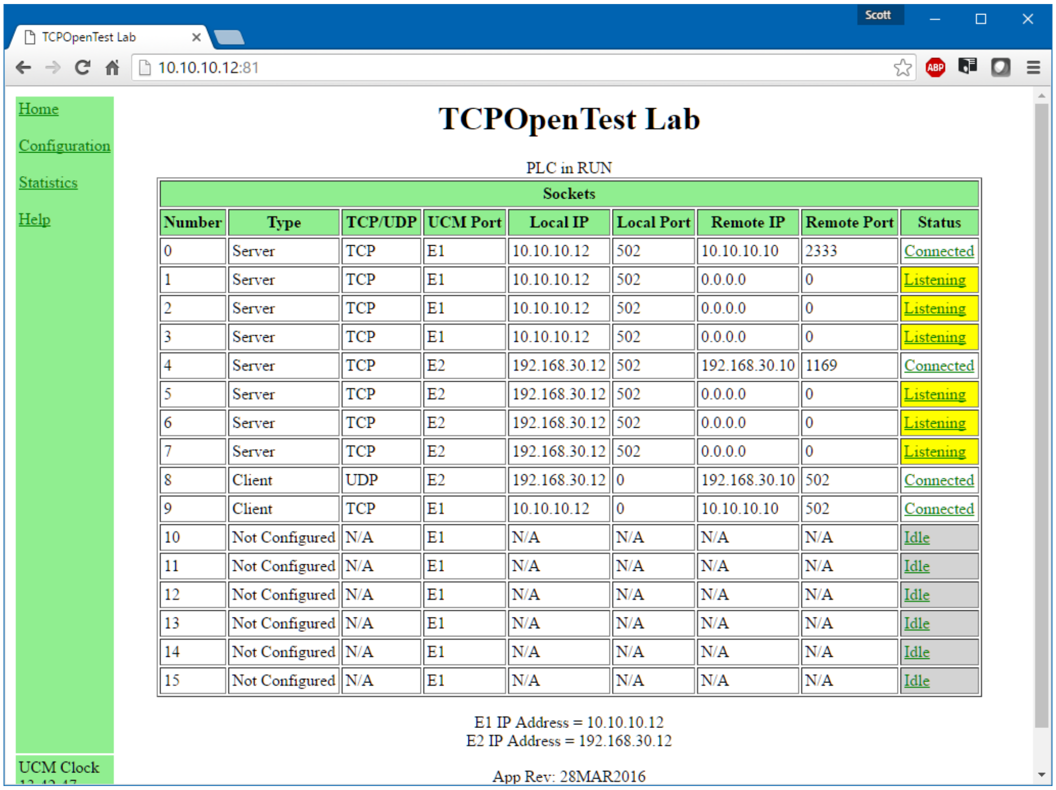
A simple web server is included in the TCPOPEN application. The TCP port number that the web server listens on is controlled by the setting in the DTM editor. Possible values are:

- 0 = Disabled
- 80 = Standard TCP port for web servers
- 81 = Default (in case the TCPOPEN PLC code is running a web server)



Note: Remember to do a 'Build Changes' and then 'Transfer to FDR server' after modifications to the DTM. The UCM will need to be rebooted after the update.

Home Page

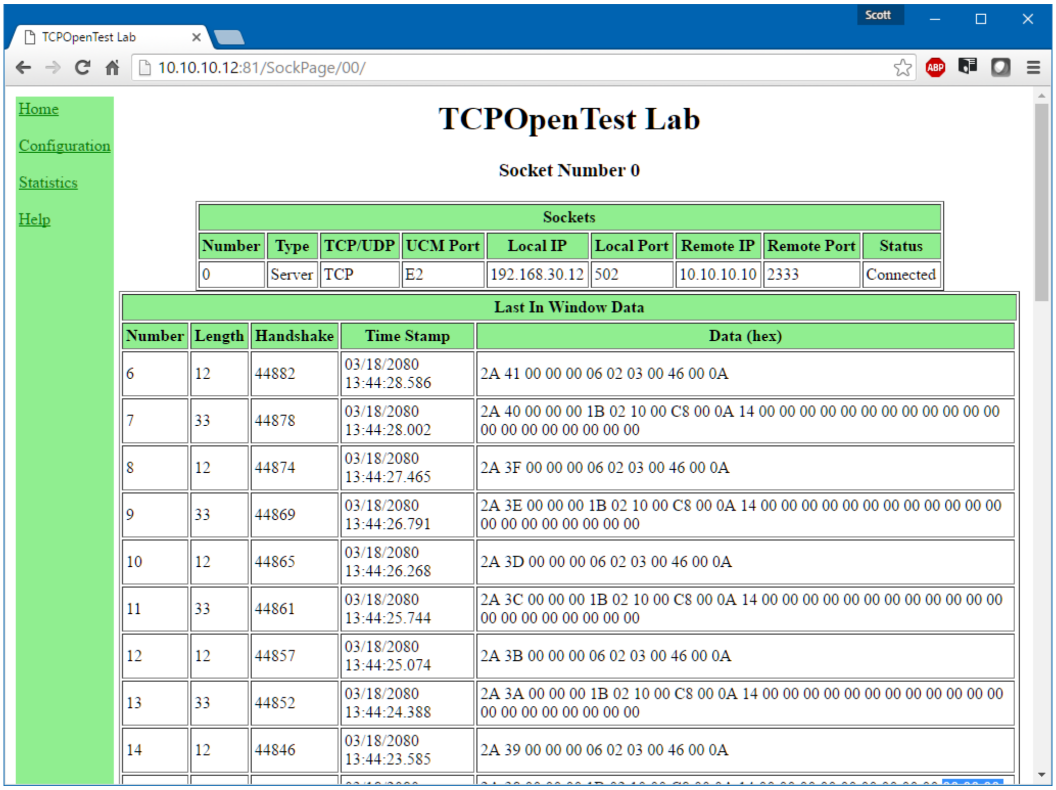


Sockets								
Number	Type	TCP/UDP	UCM Port	Local IP	Local Port	Remote IP	Remote Port	Status
0	Server	TCP	E1	10.10.10.12	502	10.10.10.10	2333	Connected
1	Server	TCP	E1	10.10.10.12	502	0.0.0.0	0	Listening
2	Server	TCP	E1	10.10.10.12	502	0.0.0.0	0	Listening
3	Server	TCP	E1	10.10.10.12	502	0.0.0.0	0	Listening
4	Server	TCP	E2	192.168.30.12	502	192.168.30.10	1169	Connected
5	Server	TCP	E2	192.168.30.12	502	0.0.0.0	0	Listening
6	Server	TCP	E2	192.168.30.12	502	0.0.0.0	0	Listening
7	Server	TCP	E2	192.168.30.12	502	0.0.0.0	0	Listening
8	Client	UDP	E2	192.168.30.12	0	192.168.30.10	502	Connected
9	Client	TCP	E1	10.10.10.12	0	10.10.10.10	502	Connected
10	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
11	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
12	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
13	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
14	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
15	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle

The Home page shows an overview of the 16 sockets. Clicking on a 'Status' link moves to a detail page for that socket.

Socket Status Page

Each socket includes a page that shows the last 20 'window' messages for both PLC Inbound and Outbound.



TCPOpenTest Lab

10.10.10.12:81/SockPage/00/

Home

Configuration

Statistics

Help

TCPOpenTest Lab

Socket Number 0

Sockets								
Number	Type	TCP/UDP	UCM Port	Local IP	Local Port	Remote IP	Remote Port	Status
0	Server	TCP	E2	192.168.30.12	502	10.10.10.10	2333	Connected

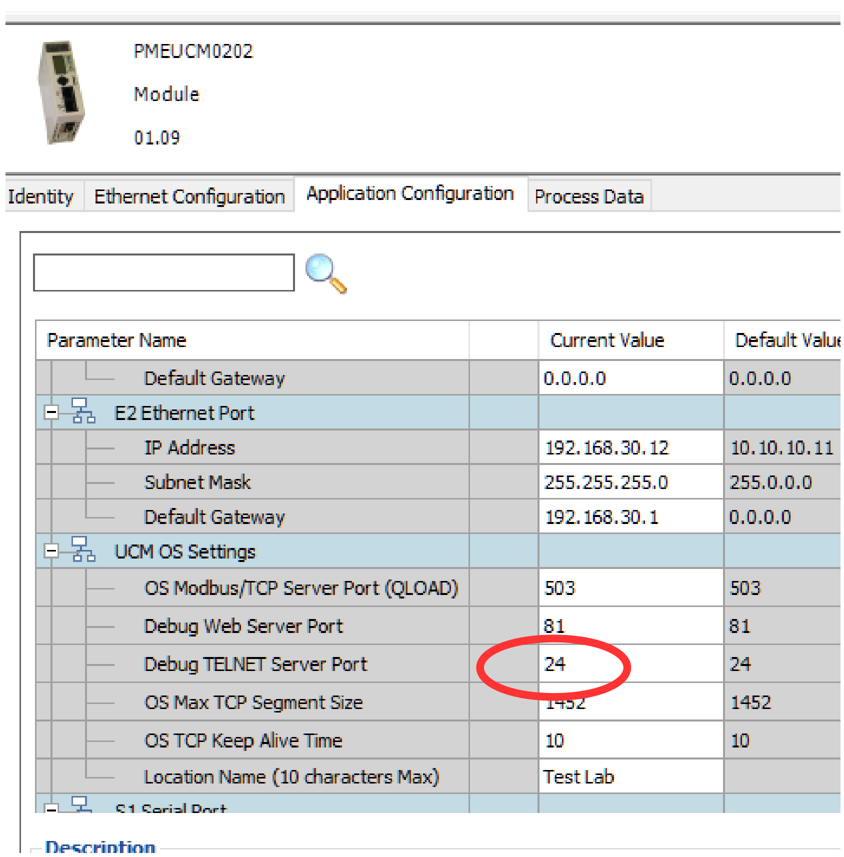
Last In Window Data				
Number	Length	Handshake	Time Stamp	Data (hex)
6	12	44882	03/18/2080 13:44:28.586	2A 41 00 00 00 06 02 03 00 46 00 0A
7	33	44878	03/18/2080 13:44:28.002	2A 40 00 00 00 1B 02 10 00 C8 00 0A 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
8	12	44874	03/18/2080 13:44:27.465	2A 3F 00 00 00 06 02 03 00 46 00 0A
9	33	44869	03/18/2080 13:44:26.791	2A 3E 00 00 00 1B 02 10 00 C8 00 0A 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10	12	44865	03/18/2080 13:44:26.268	2A 3D 00 00 00 06 02 03 00 46 00 0A
11	33	44861	03/18/2080 13:44:25.744	2A 3C 00 00 00 1B 02 10 00 C8 00 0A 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
12	12	44857	03/18/2080 13:44:25.074	2A 3B 00 00 00 06 02 03 00 46 00 0A
13	33	44852	03/18/2080 13:44:24.388	2A 3A 00 00 00 1B 02 10 00 C8 00 0A 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
14	12	44846	03/18/2080 13:44:23.585	2A 39 00 00 00 06 02 03 00 46 00 0A

The data is shown in hexadecimal. The timestamp is based on the UCM's RTC.

10 Debug Telnet Server

A simple telnet server is included in the TCPOPEN application. The TCP port number that the telnet server listens on is controlled by the setting in the DTM editor. Possible values are:

- 0 = Disabled
- 23 = Standard TCP port for telnet servers
- 24 = Default (in case the TCPOPEN PLC code is running a telnet server)



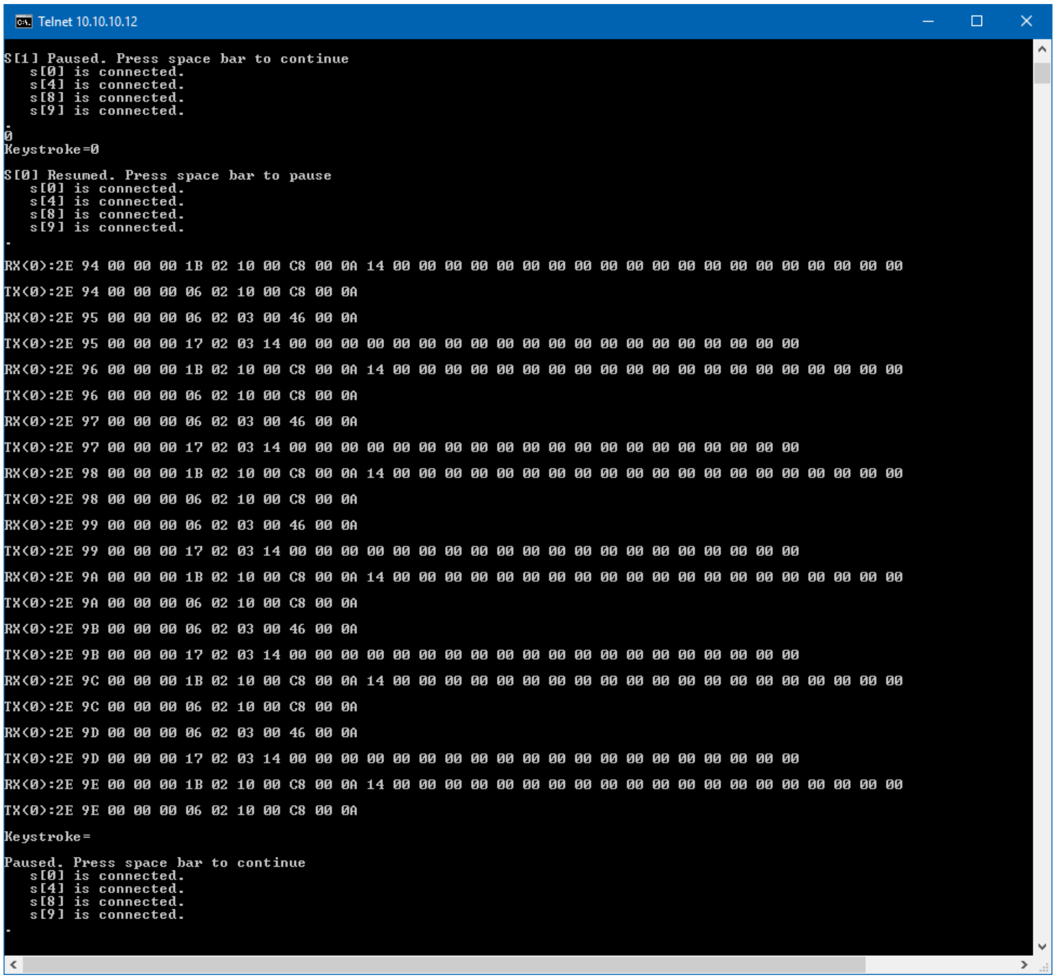
Note: Remember to do a 'Build Changes' and then 'Transfer to FDR server' after modifications to the DTM. The UCM will need to be rebooted after the update.

The Microsoft telnet client is not always installed on PCs. Most systems may enable the telnet client by:

Control Panel > Programs and Features > Turn Windows features on and off.

To use the standard Microsoft telnet client, simply open a command prompt and enter: >telnet 10.10.10.12 24

where 10.10.10.12 is the target IP Address and 24 is the target TCP port number.

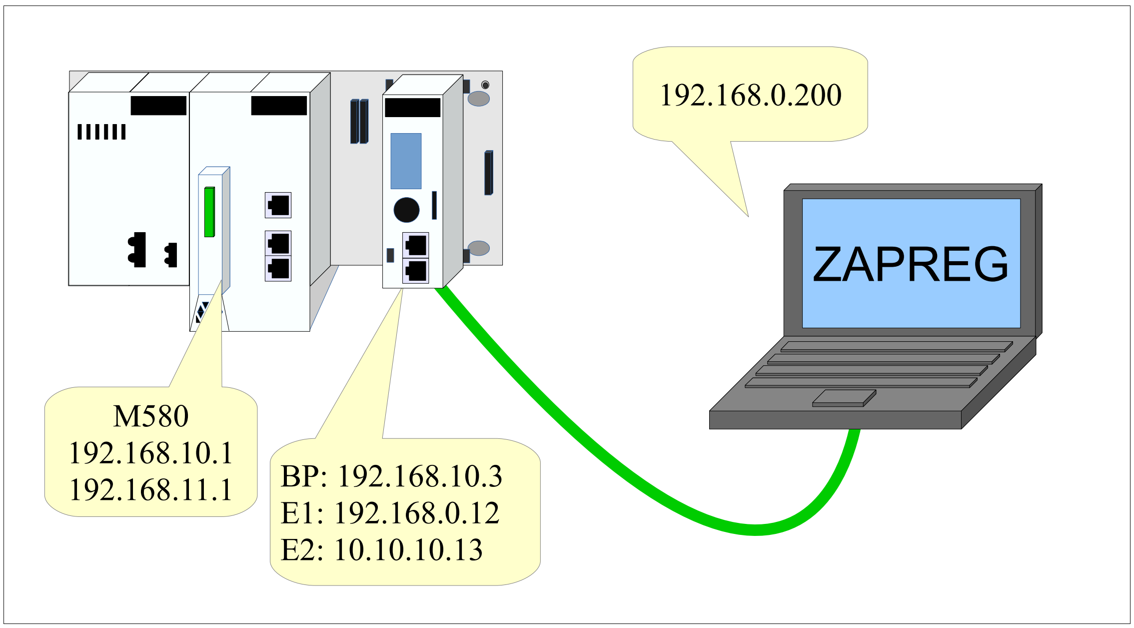


The following keystrokes adjust the operation of the telnet server:

- Space Bar – pauses/starts the display motion. It also shows a list of the connected sockets.
- 0 through 9 – enables/disables sockets 0 through 9 inclusive
- A through F – enables/disables sockets 10 through 15 inclusive. May be either upper or lower case.

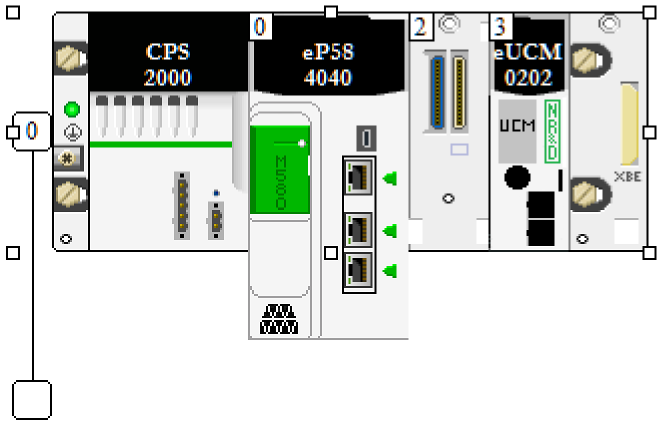
- P – forces a read of the prm file from the PTK board.

11 Modbus/TCP Server Example

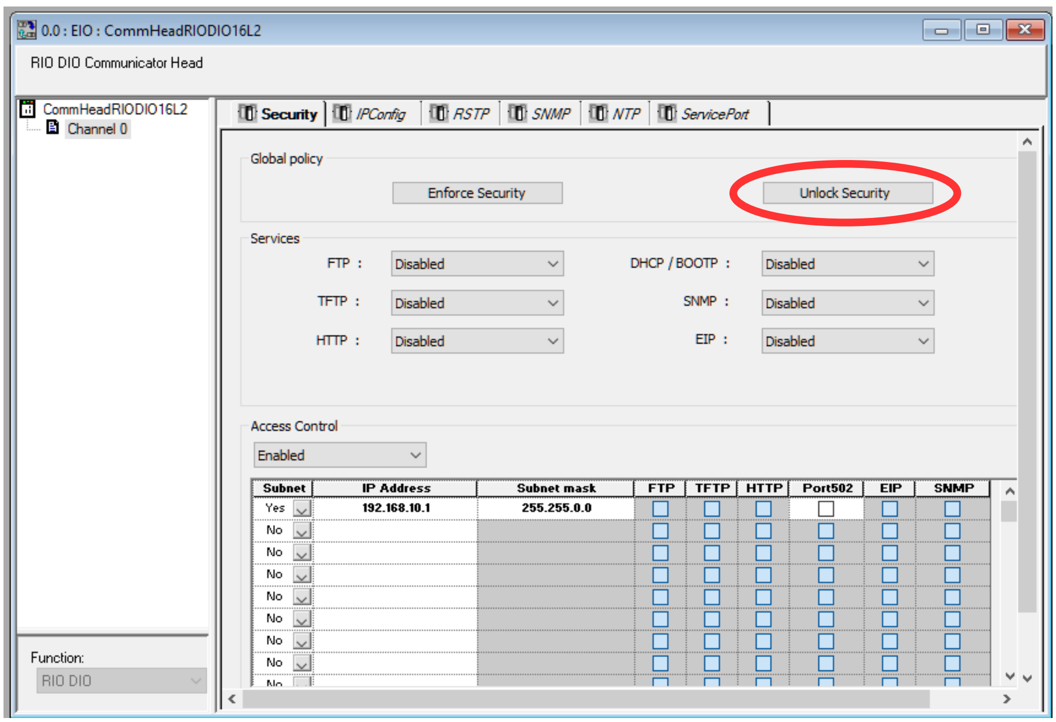


This example uses the ST segment tcpopen_mbtcp_server_w0.xst code to allow external Modbus/TCP clients (zapreg32) to read+write holding registers in a virtual slave inside the M580 CPU.

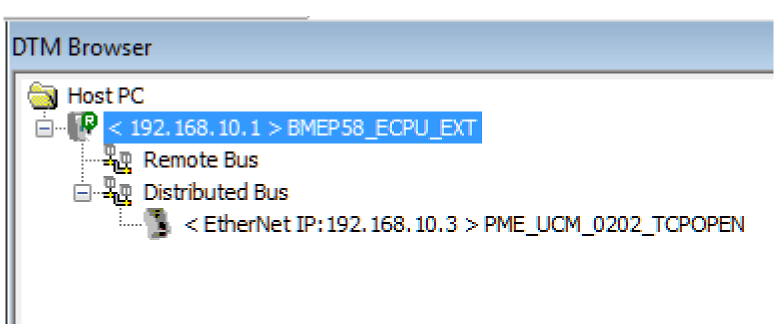
Build a new project with the CPU and the PMEUCM in the local rack, slot 3.



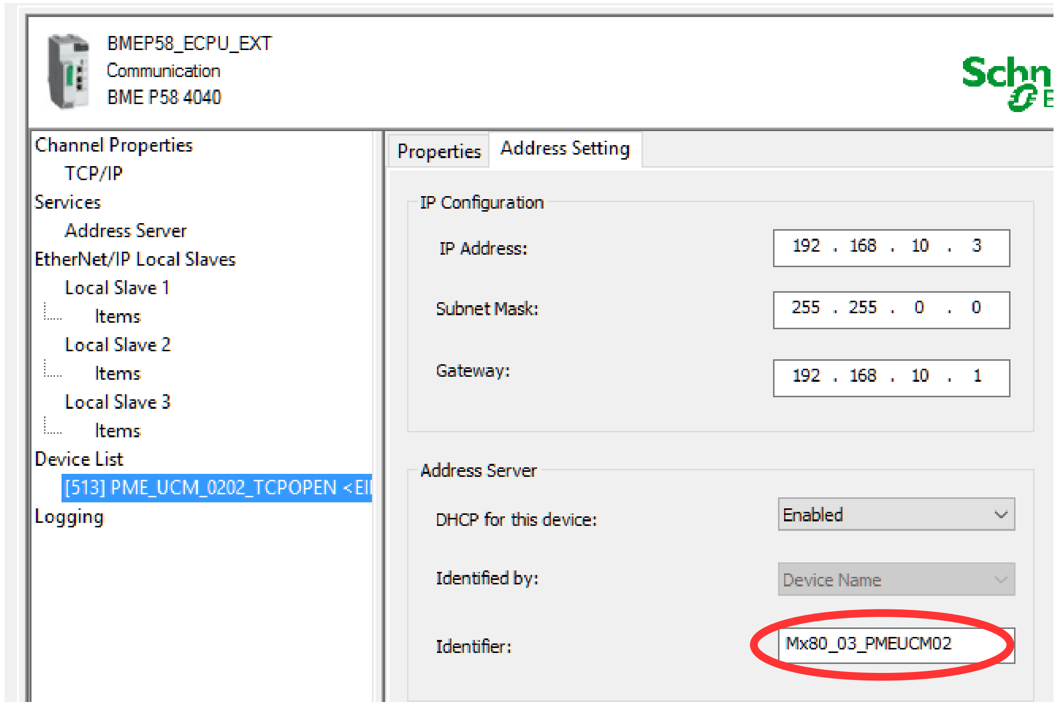
Unlock the Security in the CPU.



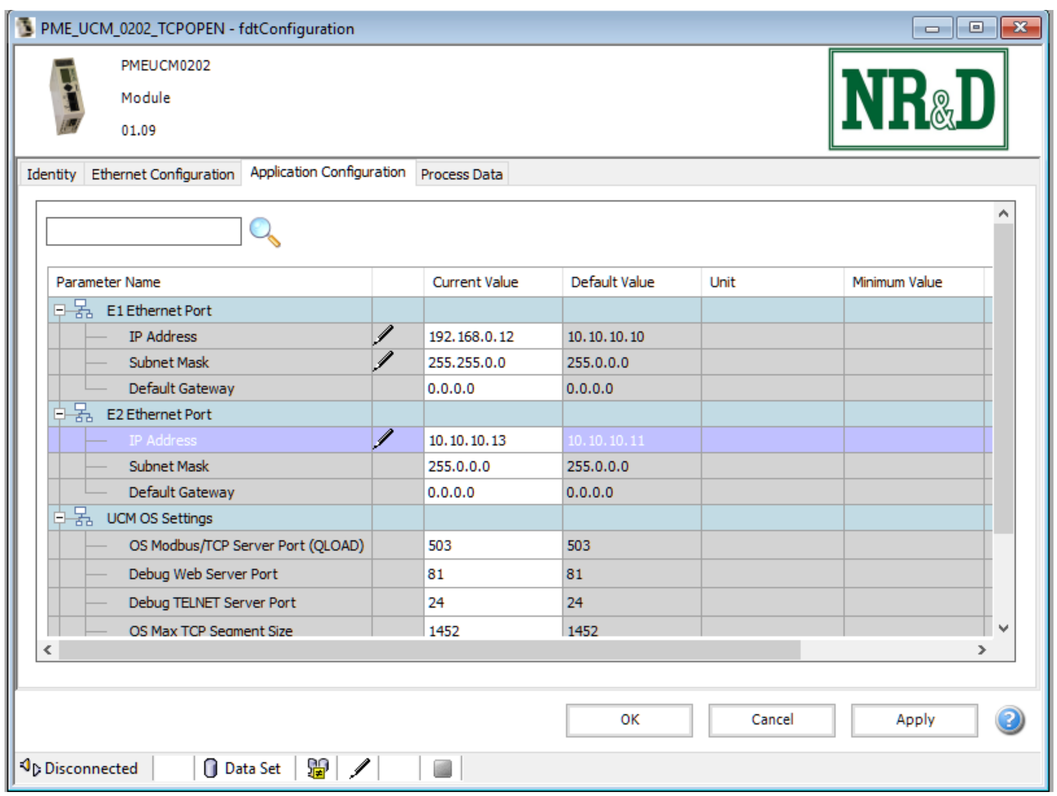
Add the TCPOPEN DTM.



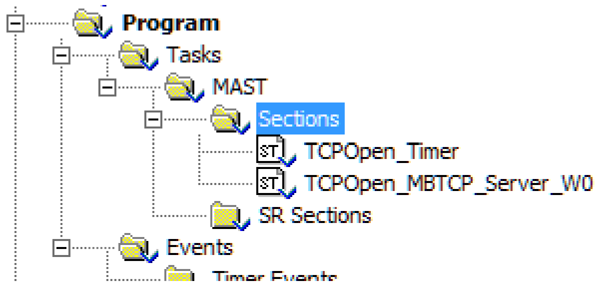
Set the "Identifier" to match the Rack+Slot number of the PMEUCM.



Edit the DTM Application Configuration to set the new IP Address values for E1 and E2. E1 is now 192.168.0.12 while E2 is 10.10.10.13.



Import the ST sections for the Timer and MBTCP_Server_W0.



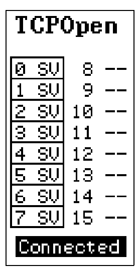
Build the Application and download it into the CPU.

Set the CPU to RUN.

Transfer the prm file to the FDR server by right clicking the UCM tree element in the DTM browser and choosing Device Menu > Additional Features > Transfer to FDR Server.

Reboot the PMEUCM card.

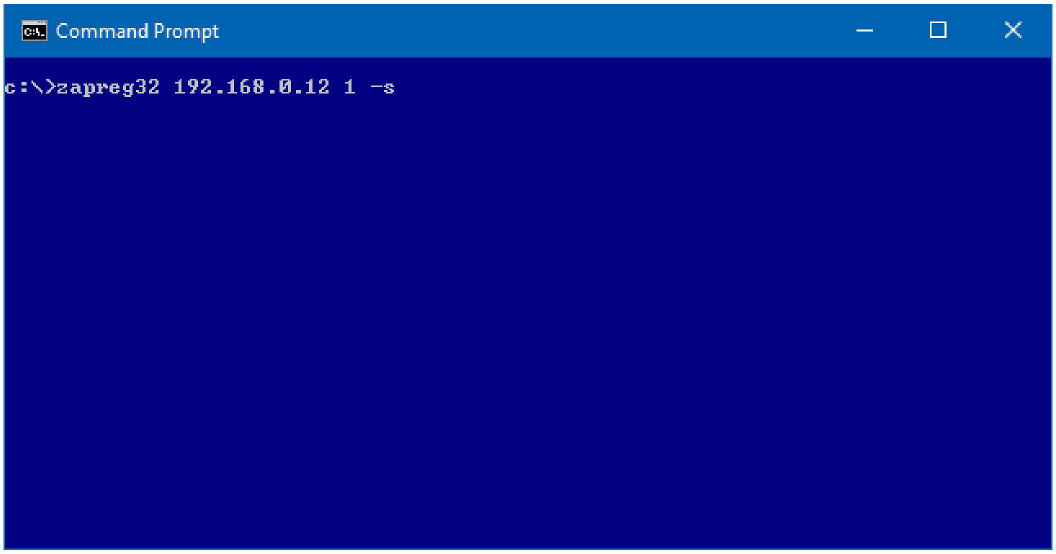
After the card boots and attaches to the M580, the screen should look like this:



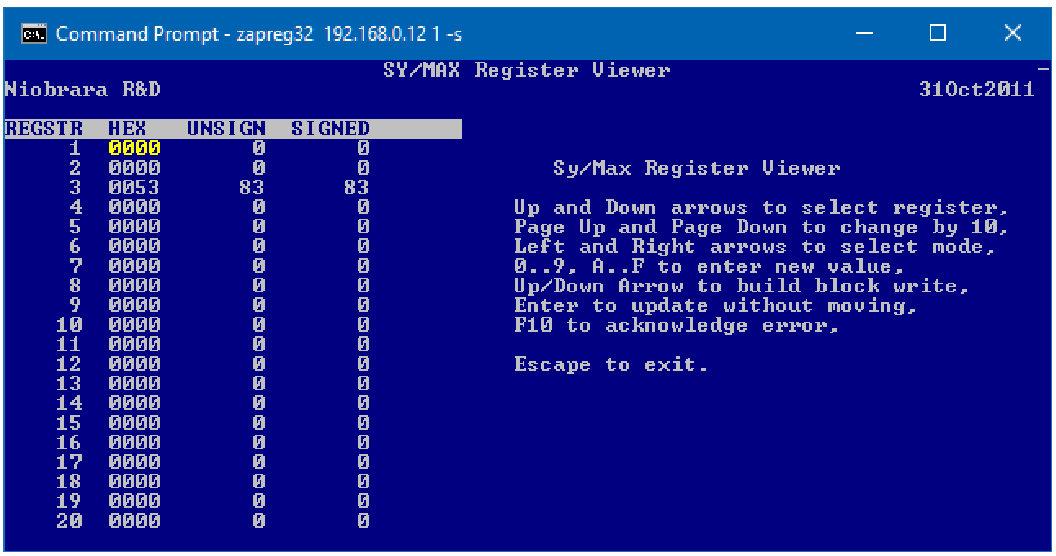
We see sockets 0 through 7 are listening but not connected. Sockets 8 through 15 are not configured.

Now, open a command prompt and enter the following command:

Zapreg32 is a simple wind32 console application that can be a Modbus/TCP client. The command line above tells it to connect to IP Address 10.10.10.10 using TCP port 502 and poll slave 1 and suppress status register polling.



Press Enter and the screen should look like this:



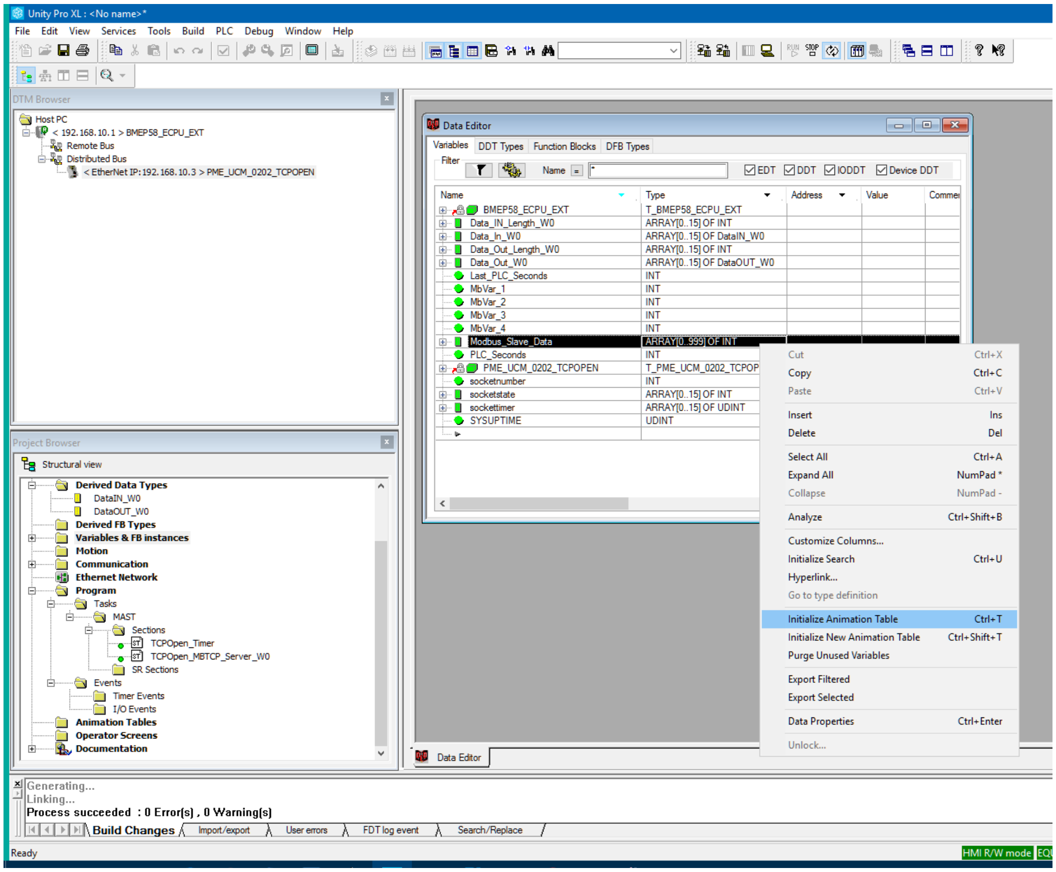
This shows the first 20 Holding registers in Slave 1. The data is shown in Hex, Unsigned, and Signed values. Register number 3 should be incrementing. This register is being incremented with each Modbus read by the PLC code.

Note, Zapreg32 is a standard Modbus poller. It uses the older 'Modicon' standard of starting at Holding Register 1. The M580 code is written to use the newer 'Tele' standard of starting at register 0. A look at the ST code in State 20 shows:

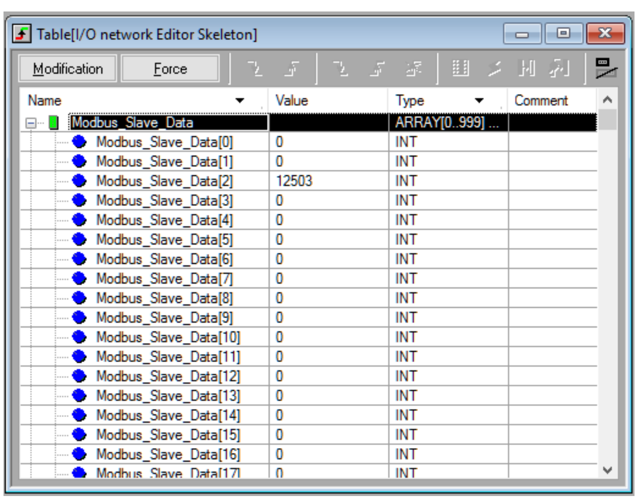
```
Modbus_Slave_Data[2] := Modbus_Slave_Data[2] + 1 ;
```

which explains why Zapreg32 is showing register 3 incrementing.

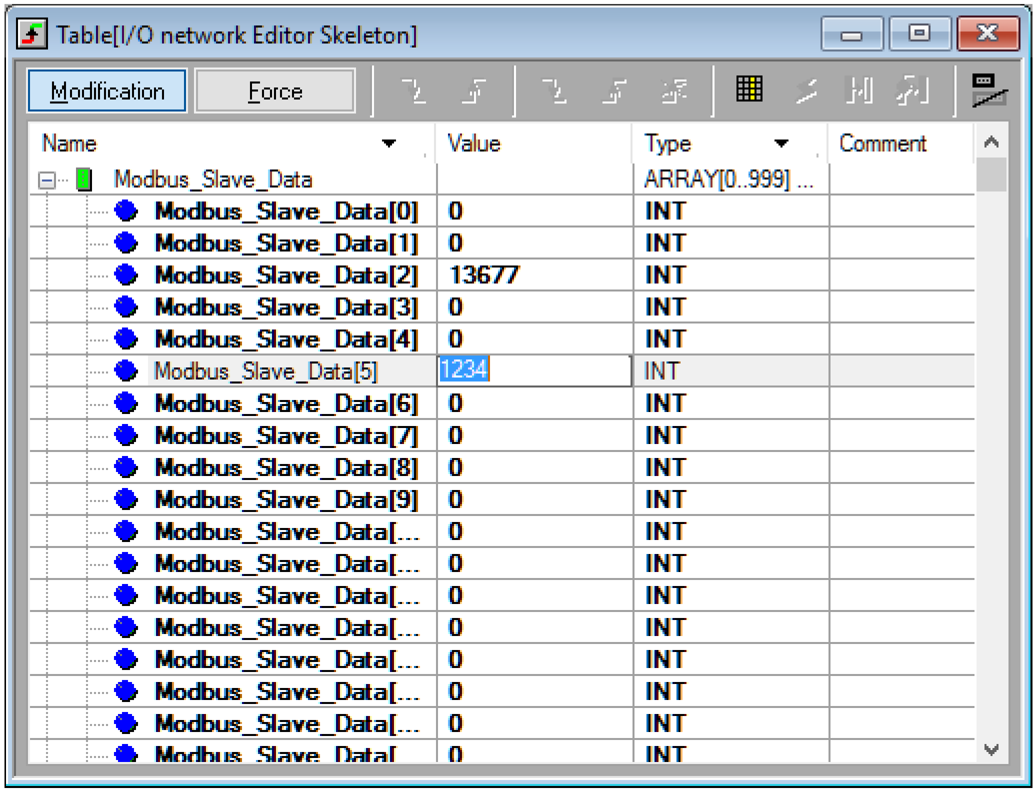
Back in Unity Pro, open the Variables & FB Instances and right click on the Modbus_Slave_Data structure and Initialize Animation Table.



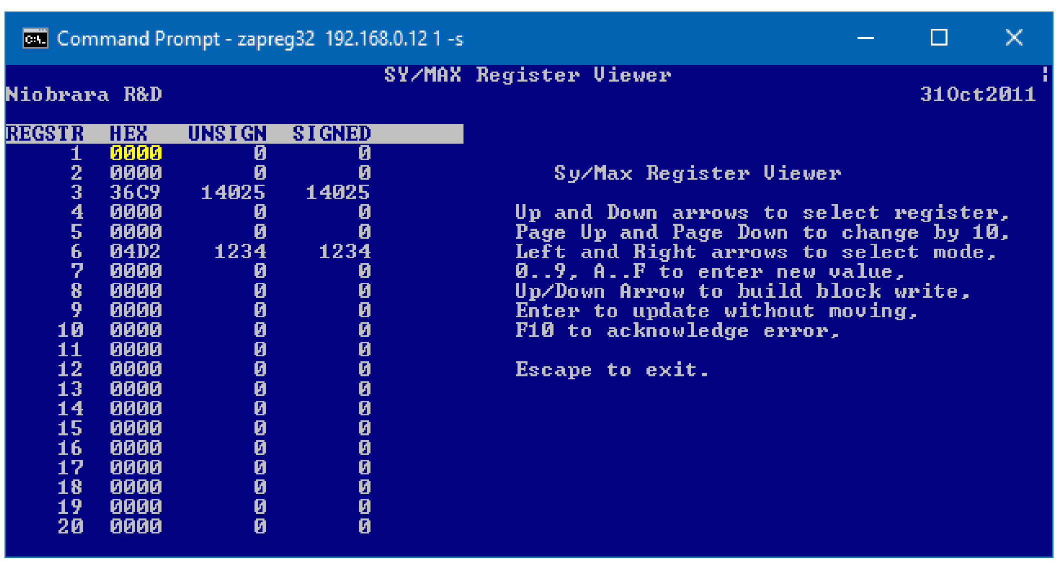
We see that there within this Modbus virtual slave, there are 1000 registers and register [2] is incrementing.



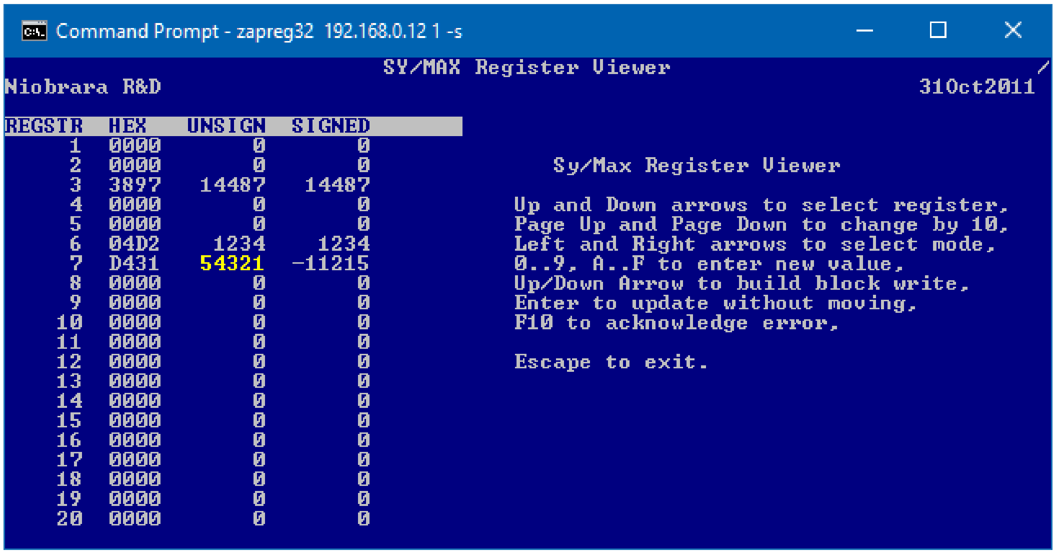
Select “Modification” and change the value of [5] from 0 to 1234.



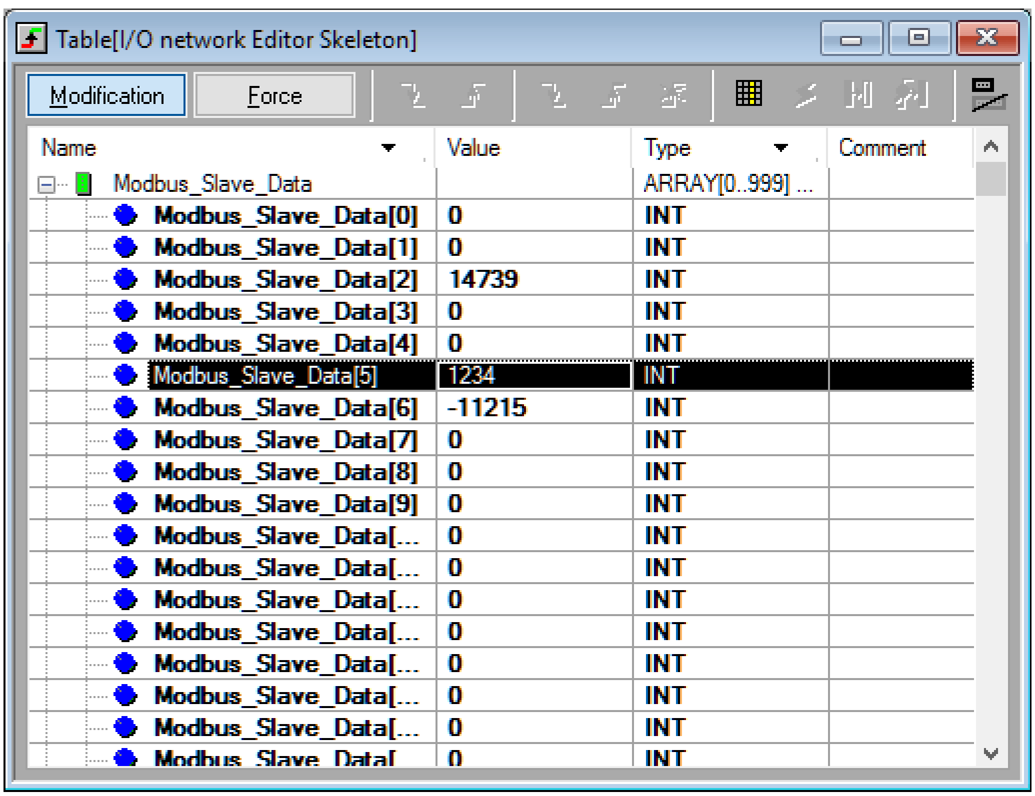
Now go back to the zapreg32 window and we see that register 6 has the value 1234.



Move the cursor in the zapreg window by using the arrow keys. Move the cursor down to register 7 and type in the value 54321 in the unsigned column.

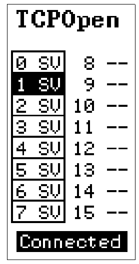


Now go look at the Unity table.

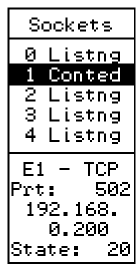


The data -11215 is now in register [6] as expected.

Take a look at the front LCD panel. Socket 1 is now filled in to indicate an active connection. (It could have been any of sockets 0, 1, 2, or 3 that where highlighted. It just depends on which one was connected when Zapreg32 started.)



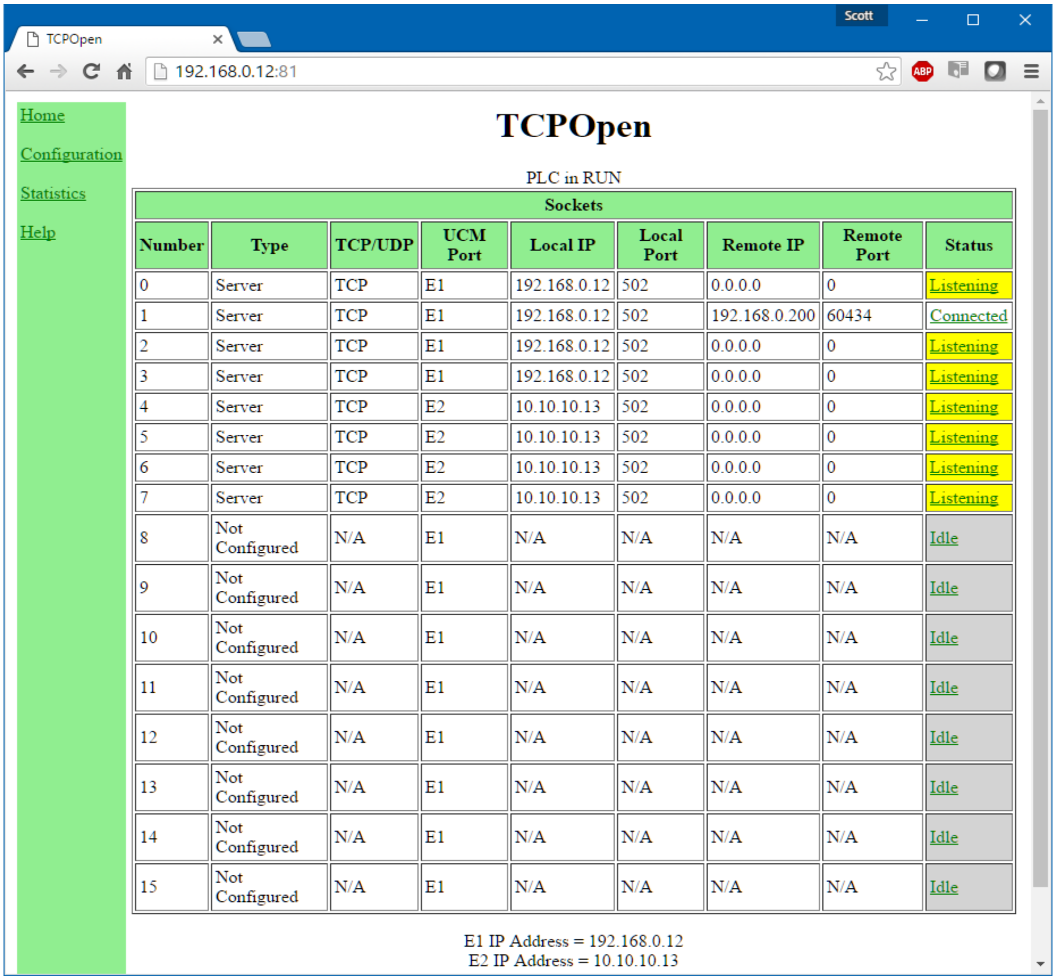
Use the joystick to select “Sockets” and then down arrow to socket 2. This shows connected, E1, TCP, port 502, remote IP 192.168.0.200 and that the server is in State 20 (waiting for Modbus query).



Open a web browser and enter the url of 192.168.0.12:81.

The page shows that socket 2 is connected to the PC. The PC is at IP Address 192.168.0.200.

Clicking on the “Connected” link for socket 2 pulls up the last 20 inbound and outbound messages.



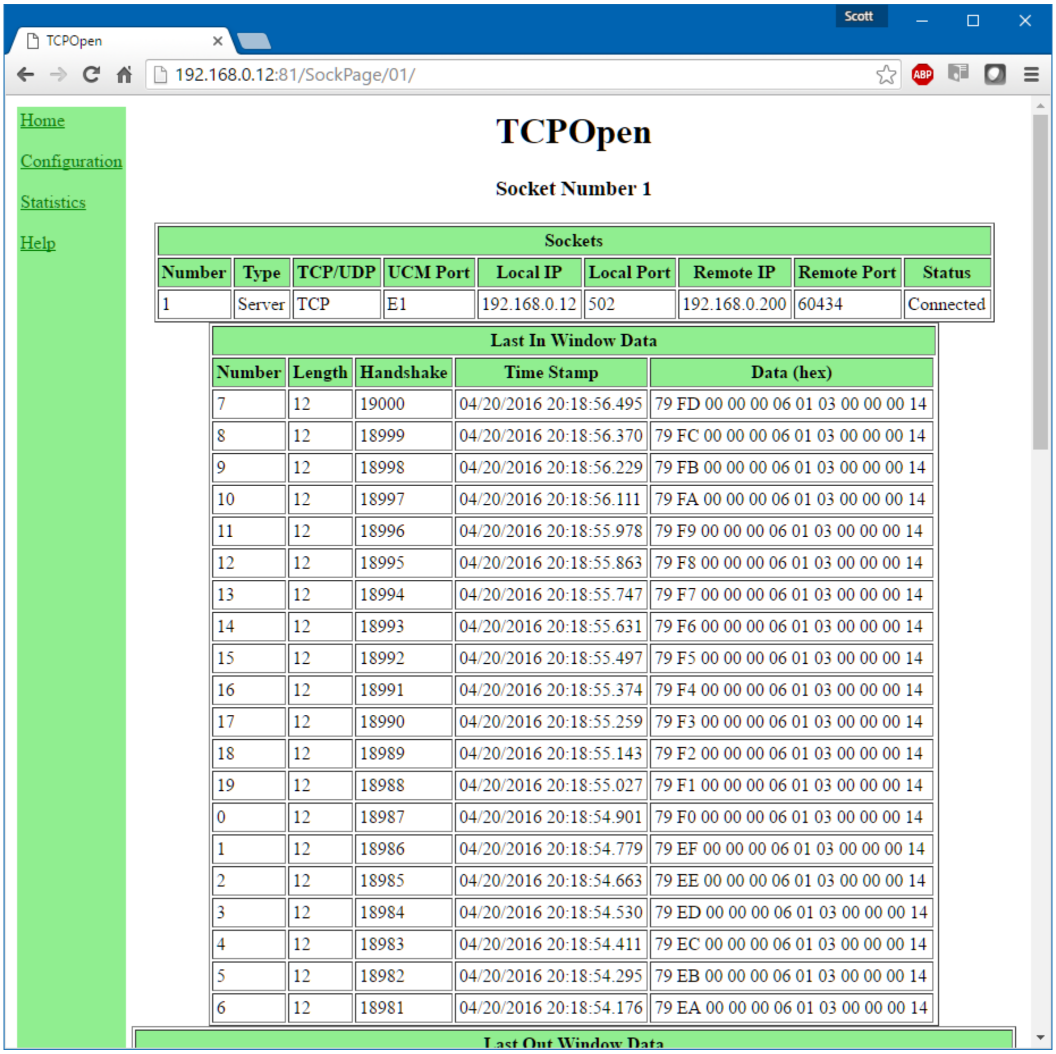
TCPOpen

PLC in RUN

Sockets								
Number	Type	TCP/UDP	UCM Port	Local IP	Local Port	Remote IP	Remote Port	Status
0	Server	TCP	E1	192.168.0.12	502	0.0.0.0	0	Listening
1	Server	TCP	E1	192.168.0.12	502	192.168.0.200	60434	Connected
2	Server	TCP	E1	192.168.0.12	502	0.0.0.0	0	Listening
3	Server	TCP	E1	192.168.0.12	502	0.0.0.0	0	Listening
4	Server	TCP	E2	10.10.10.13	502	0.0.0.0	0	Listening
5	Server	TCP	E2	10.10.10.13	502	0.0.0.0	0	Listening
6	Server	TCP	E2	10.10.10.13	502	0.0.0.0	0	Listening
7	Server	TCP	E2	10.10.10.13	502	0.0.0.0	0	Listening
8	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
9	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
10	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
11	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
12	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
13	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
14	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle
15	Not Configured	N/A	E1	N/A	N/A	N/A	N/A	Idle

E1 IP Address = 192.168.0.12
E2 IP Address = 10.10.10.13

Click on the “Connected” link for Socket 1.



TCPOpen

Socket Number 1

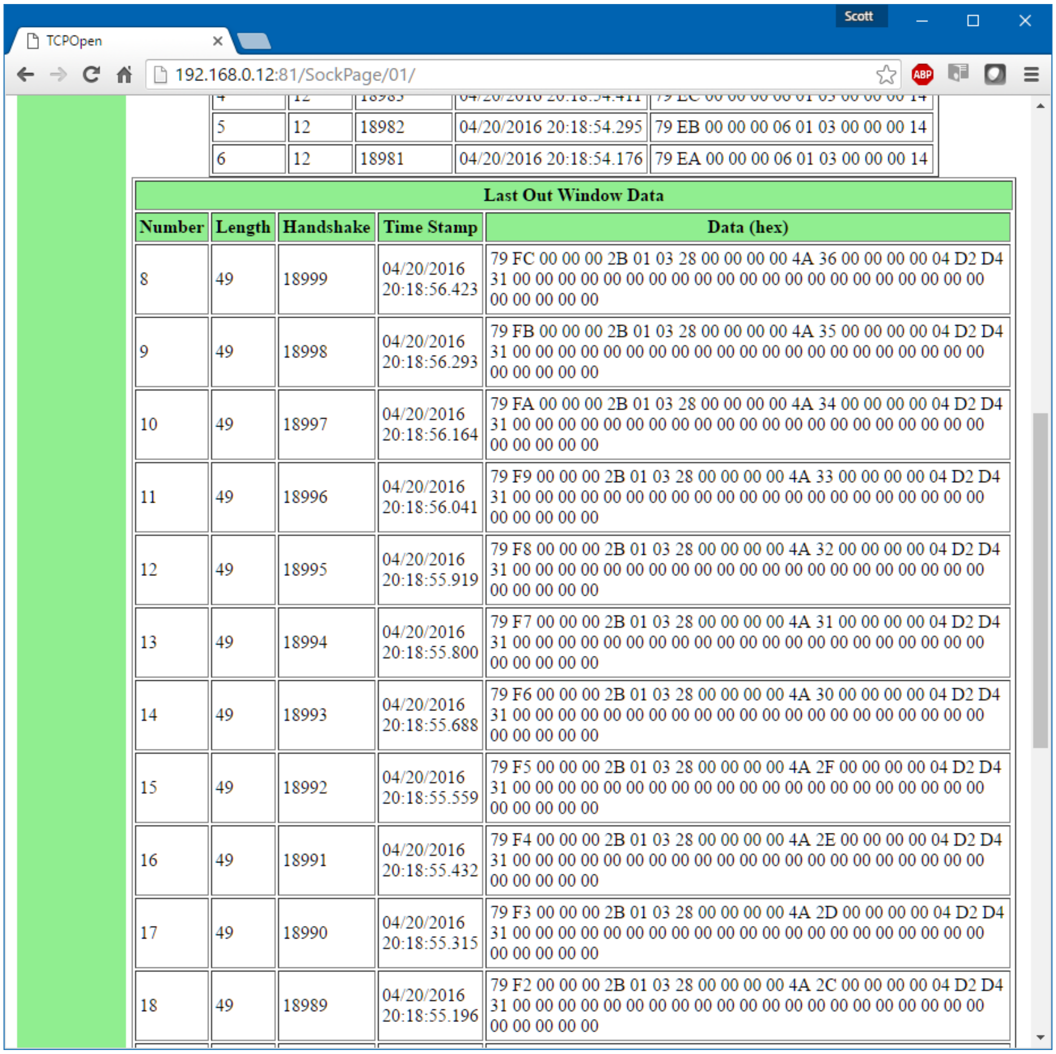
Sockets								
Number	Type	TCP/UDP	UCM Port	Local IP	Local Port	Remote IP	Remote Port	Status
1	Server	TCP	E1	192.168.0.12	502	192.168.0.200	60434	Connected

Last In Window Data				
Number	Length	Handshake	Time Stamp	Data (hex)
7	12	19000	04/20/2016 20:18:56.495	79 FD 00 00 00 06 01 03 00 00 00 14
8	12	18999	04/20/2016 20:18:56.370	79 FC 00 00 00 06 01 03 00 00 00 14
9	12	18998	04/20/2016 20:18:56.229	79 FB 00 00 00 06 01 03 00 00 00 14
10	12	18997	04/20/2016 20:18:56.111	79 FA 00 00 00 06 01 03 00 00 00 14
11	12	18996	04/20/2016 20:18:55.978	79 F9 00 00 00 06 01 03 00 00 00 14
12	12	18995	04/20/2016 20:18:55.863	79 F8 00 00 00 06 01 03 00 00 00 14
13	12	18994	04/20/2016 20:18:55.747	79 F7 00 00 00 06 01 03 00 00 00 14
14	12	18993	04/20/2016 20:18:55.631	79 F6 00 00 00 06 01 03 00 00 00 14
15	12	18992	04/20/2016 20:18:55.497	79 F5 00 00 00 06 01 03 00 00 00 14
16	12	18991	04/20/2016 20:18:55.374	79 F4 00 00 00 06 01 03 00 00 00 14
17	12	18990	04/20/2016 20:18:55.259	79 F3 00 00 00 06 01 03 00 00 00 14
18	12	18989	04/20/2016 20:18:55.143	79 F2 00 00 00 06 01 03 00 00 00 14
19	12	18988	04/20/2016 20:18:55.027	79 F1 00 00 00 06 01 03 00 00 00 14
0	12	18987	04/20/2016 20:18:54.901	79 F0 00 00 00 06 01 03 00 00 00 14
1	12	18986	04/20/2016 20:18:54.779	79 EF 00 00 00 06 01 03 00 00 00 14
2	12	18985	04/20/2016 20:18:54.663	79 EE 00 00 00 06 01 03 00 00 00 14
3	12	18984	04/20/2016 20:18:54.530	79 ED 00 00 00 06 01 03 00 00 00 14
4	12	18983	04/20/2016 20:18:54.411	79 EC 00 00 00 06 01 03 00 00 00 14
5	12	18982	04/20/2016 20:18:54.295	79 EB 00 00 00 06 01 03 00 00 00 14
6	12	18981	04/20/2016 20:18:54.176	79 EA 00 00 00 06 01 03 00 00 00 14

Last Out Window Data

This page shows the time-stamped log of the last 20 messages sent to the PLC on Window W0. The hexadecimal data shows the FC 03 message to slave 1 starting at register 0 with a count of 20.

Scrolling down the window shows the reply data returned from the M580.



4	12	18983	04/20/2016 20:18:54.411	79 EC 00 00 00 06 01 03 00 00 00 14
5	12	18982	04/20/2016 20:18:54.295	79 EB 00 00 00 06 01 03 00 00 00 14
6	12	18981	04/20/2016 20:18:54.176	79 EA 00 00 00 06 01 03 00 00 00 14

Last Out Window Data				
Number	**Length**	**Handshake**	**Time Stamp**	**Data (hex)**
8	49	18999	04/20/2016 20:18:56.423	79 FC 00 00 00 2B 01 03 28 00 00 00 00 4A 36 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
9	49	18998	04/20/2016 20:18:56.293	79 FB 00 00 00 2B 01 03 28 00 00 00 00 4A 35 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10	49	18997	04/20/2016 20:18:56.164	79 FA 00 00 00 2B 01 03 28 00 00 00 00 4A 34 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
11	49	18996	04/20/2016 20:18:56.041	79 F9 00 00 00 2B 01 03 28 00 00 00 00 4A 33 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
12	49	18995	04/20/2016 20:18:55.919	79 F8 00 00 00 2B 01 03 28 00 00 00 00 4A 32 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
13	49	18994	04/20/2016 20:18:55.800	79 F7 00 00 00 2B 01 03 28 00 00 00 00 4A 31 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
14	49	18993	04/20/2016 20:18:55.688	79 F6 00 00 00 2B 01 03 28 00 00 00 00 4A 30 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
15	49	18992	04/20/2016 20:18:55.559	79 F5 00 00 00 2B 01 03 28 00 00 00 00 4A 2F 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
16	49	18991	04/20/2016 20:18:55.432	79 F4 00 00 00 2B 01 03 28 00 00 00 00 4A 2E 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
17	49	18990	04/20/2016 20:18:55.315	79 F3 00 00 00 2B 01 03 28 00 00 00 00 4A 2D 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
18	49	18989	04/20/2016 20:18:55.196	79 F2 00 00 00 2B 01 03 28 00 00 00 00 4A 2C 00 00 00 00 04 D2 D4 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

12 Modbus/TCP Server+Client Example2