PMEUCM DTM

Installation Manual

This manual covers the PMEUCM DTM development and installation procedures.

Effective: August 11, 2017

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

All trademarks and registered trademarks are the property of their respective owners. Subject to change without notice.

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 exchange of data between the M580 PAC and the PMEUCM across the Ethernet bacplane is controlled by a DTM. This document provides on overview of the UCM DTM development process and installation procedure.

DTM stands for Device Type Manager. DTMs are special files that provide information concerning the configuration and operation of some sort of 'field device'. Typical field devices may be motor drives, electric meters, or the PMEUCM.

DTMs are used with a software package called a FDT (Field Device Tool). Unity Pro includes a built-in FDT. DTMs are installed into the FDT (Unity Pro) and then used to configure the remote device. The DTMs conform to a standard so field devices from different vendors may all be configured within a single FDT.

The M580 PAC relies heavily on DTMs. Many of the Schneider-Electic x80 cards use DTMs for their configuration. All CAPP member x80 cards (like the PMEUCM) use a DTM to tell the M580 which variables are exchanged with the card, static and dynamic configuration, etc.

The PMEUCM is different from other x80 cards because it does not use just a single DTM. Products like the SCAIME PMESWT0100 have a single DTM that uniquely defines all configuration aspects of the card. The PMEUCM is 'user programmable' which means that the data structure that needs to be exchanged with the M580 will be dependent upon the 'user program'.

A customer may have a PMEUCM application that communicates with an inkjet printer. This program may need to exchange data like a string for the label, commands to start/stop printing, feedback about ink levels. A specific DTM for this application will be needed.

A different customer may have a PMEUCM application that communicates with an automotive dynamometer. This application would require variables like wheel speed,

power, start/stop/coast, etc. Obviously, this application needs a different DTM than the printer app.

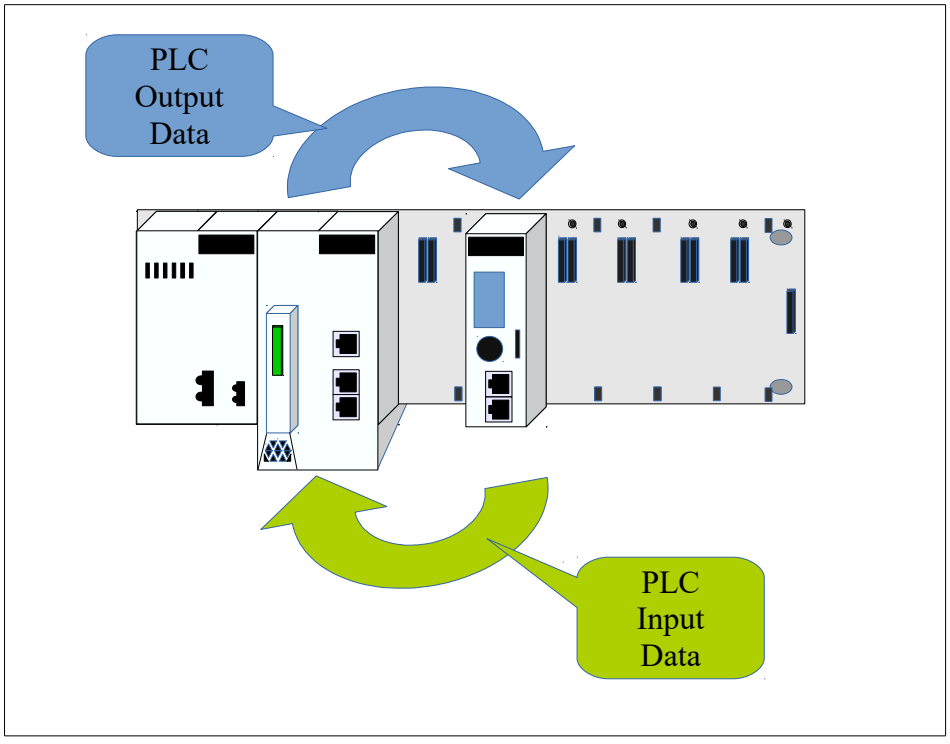
Developing a custom DTM for each PMEUCM application would be cost prohibitive. Fortunately, Unity Pro includes a 'Generic PTK DTM' which allows a specific type of XML (DDXML) file to be imported into the Unity Pro package and be used as a DTM.

Building one of these DDXML files by hand is tedious, so Niobrara has developed a utility that will generate a proper DDXML file from a simple comma separated text file (.txt) and automatically installs it into the location required by Unity Pro. A simple update of the DTM Hardware Catalog inside Unity Pro is all that is needed to bring a custom PMEUCM DTM to the M580.

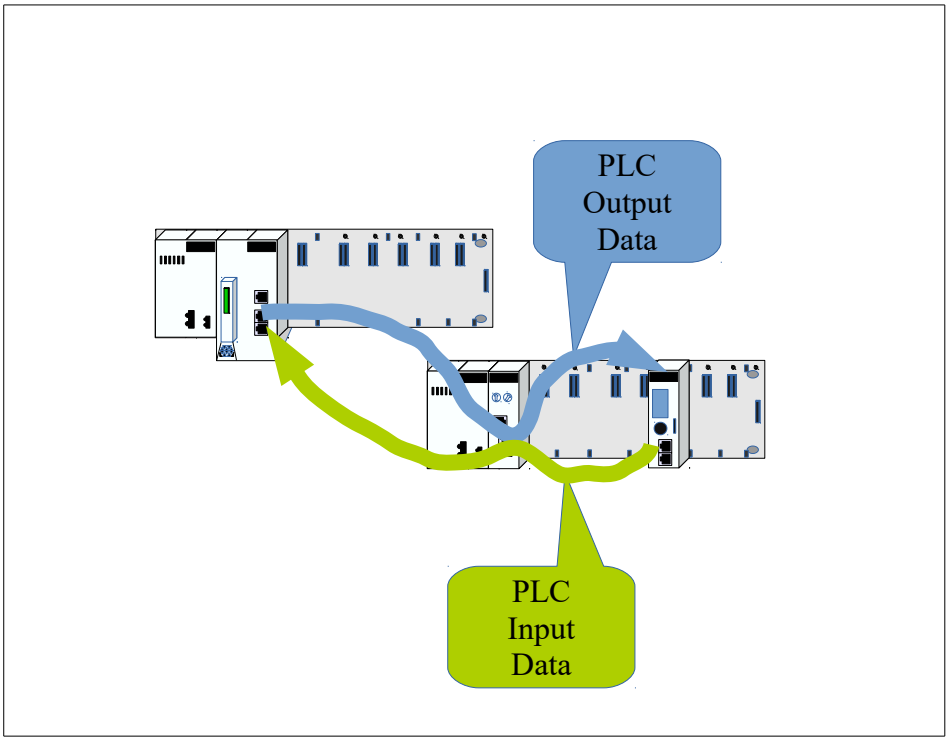
2 DIO Overview

All PME devices operate as an Ethernet/IP DIO device to the M580 CPU. This means that the I/O data communication between the M580 CPU and the PMEUCM is done by means of an Ethernet/IP data exchange across the backplane.

Wben the PMEUCM located in the CPU rack, this data exchange happens completely across the Ethernet backplane between the CPU and the PMEUCM backplane interface.



In the case of the PMEUCM located in an eRIO rack, the exchange happens across the RIO ring from the CPU through the CRA and across the Ethernet backplane to the PMEUCM backplane interface.



Each Ethernet/IP message from the CPU is transported via the RIO ring, through the CRA, and across the Ethernet backplane to the PMEUCM.

The update time of the I/O Data between the CPU and PMEUCM is controlled by the Request Packet Interval (RPI) The RPI is typically 10mS and may be adjusted using the DTM of the PMEUCM.

CPU DIO Limitations

There are several limiting factors concerning DIO use with the M580 PLC. The two most common limits are the number of device connections and the number of bytes exchanged.

DIO Connection Limits

The different M580 CPU models feature limits on the number of Ethernet/IP (EIP) connections, Modbus/TCP (MB) connections and combined EIP+MB connections. Table 2.1 shows the connection limits for M580 CPUs.

DIO Byte Count Limits

Along with limitations on the number of possible DIO devices that may be connected to a single M580 CPU, there are limits on the total amount of bytes of data exchanged.

The actual limiting factor will usually be the total number of PLC Input or Output

bytes used by the DIO. Exceeding the PLC's limit on any of these items will result in a failure to build the proposed system inside Unity Pro.

Table 2.1: CPU DIO Limits

CPU	Max EIP DIO	Max Modbus DIO	Max DIO EIP+MB	Total Bytes DIO INPUT	Total Bytes DIO OUTPUT
BMEP581020	64	64	64	2048	2048
BMEP582020	128	128	128	4096	4096
BMEP582040	64	64	64	2048	2048
BMEP583020	128	128	128	4096	4096
BMEP583040	64	64	64	2048	2048
BMEP584020	128	128	128	8192	8192
BMEP584040	64	64	64	2048	2048
BMEP585040	64	64	64	2048	2048
BMEP586040	64	64	64	2048	2048

For example, the SCAIME PMESWT0100 Weighing module's DTM configures 118 bytes of PLC input data and 32 bytes of PLC output data per card. The maximum number of PMESWT0100 cards that may be installed in a 581020 CPU is limited to 17 cards. This is because having 18 cards would exceed the maximum allowed number of DIO input bytes for the CPU.

18 modules x 118 bytes/module = 2124 bytes

The PMEUCM setup includes several simple example applications with DTM files. EXAMPLE6 includes 1960 bytes of PLC inputs and 1960 bytes of PLC outputs. A 581020 CPU would not be able to include a both the SCAIME PMESWT0100 card and a PMEUCM running EXAMPLE6.

118 (SWT bytes) + 1960 (UCM bytes) = 2078 bytes total

A 582020 CPU would need to be used because it has 4096 bytes available for DIO.

Notice that the byte limitation includes all DIO connections from the CPU.

PME modules + Modbus/TCP slaves + Ethernet/IP slaves

Consider a 584040 CPU that is scanning 30 Holding Registers from each of 15 Modbus/TCP electric meters. This action uses 900 bytes of the 2048 PLC inputs available.

(15 slaves) x (30 register/slave) x (2 bytes/register) = 900 bytes total

This PLC would then be able to include 9 PMESWT0100 weighing modules.

2048 – 900 = (1148 available bytes) / (118 byes/module) = 9.73 modules

NOC DIO

One option to consider when running into DIO byte count restrictions would be to move some of the DIO to an BMENOC3*1 module. The NOC is capable of controlling its own DIO on its own Ethernet ports. The eNOC Ethernet backplane must be enabled for the DIO to function.

Hot Standby Implications

The PMEUCM may be used in M580 Hot Standby systems (HSBY) remote Ethernet rack.

Inputs Not Bumpless

Like most DIO connections, the PLC Input data from a PME module is not "bumpless" during a transfer from Secondary to the new Primary. The Ethernet/IP connections will be closed and the module "Freshness" will drop to 0 during during the transfer. The data may not be available for several PLC scans after the transfer as the Ethernet/IP connections are re-established.

Outputs Not Bumpless

The PMEUCM will mark the PLC offline when the transfer occurs.

Remote Rack Mounting Only

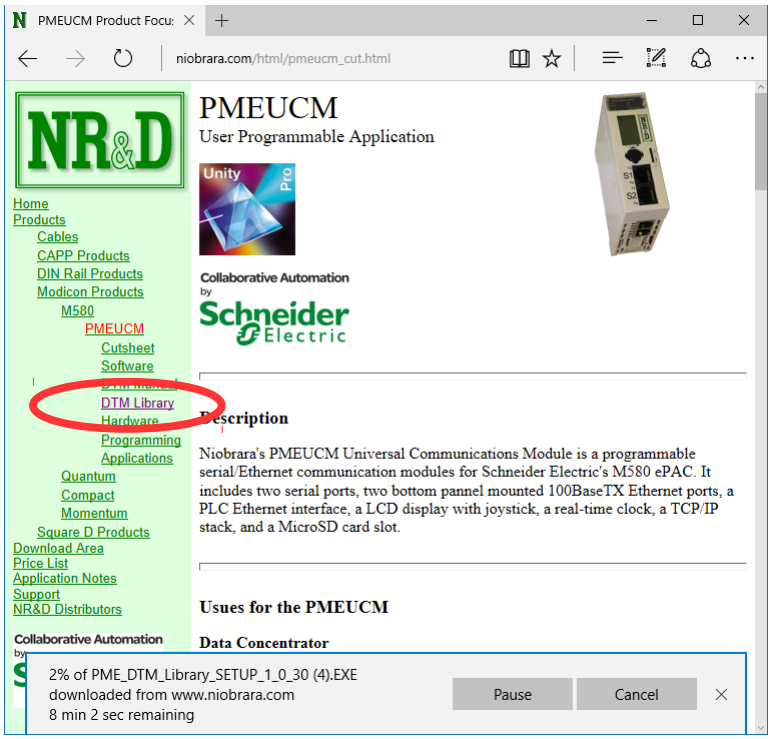
The PMEUCM may only be used in a remote Ethernet rack in a M580 HSBY system. The PMEUCM may not be mounted in either HSBY CPU rack.

3 SE PME DTM Library

The latest version of Schneider Electric's PME DTM Library must be installed before attempting to use the PMEUCM. The PMEUCM requires many newly added features to the PME DTM.

The latest version is available at Niobrara's web site:

http://www.niobrara.com/programs/PME_DTM_Library_SETUP_1_0_30.EXE



◦ *Installation*

NOTE: Unity Pro must be closed before installing the PME_DTM_Library.

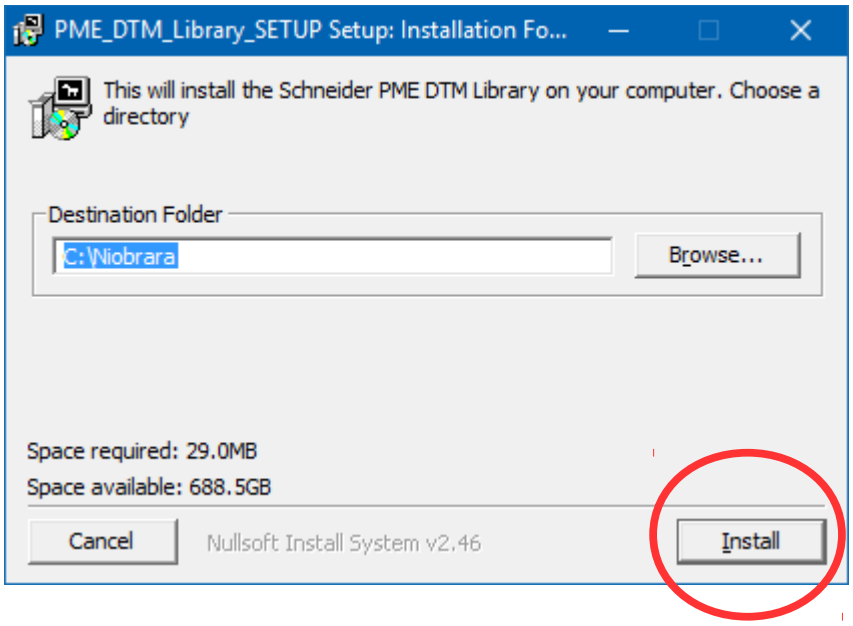
NOTE: Unity Pro V11.1 and V12.0 are shipped with an older version of the PME_DTM_Library already installed. The installed version must be updated to allow the PMEUCM0302 to operate properly.

NOTE: Installing a new version of the PME_DTM_LIBRARY will remove all currently installed DDXML files from the repository.

Use the File>Restore feature of the NRDPTKDDXMLUTIL program to recover the previously installed DTMs.

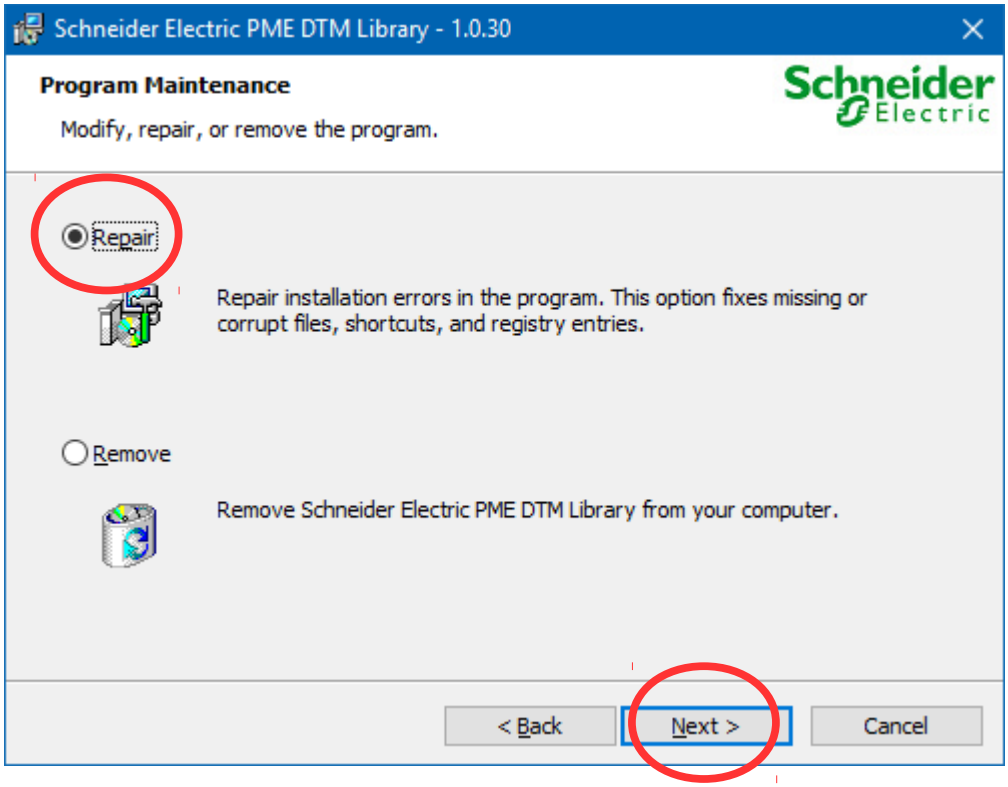
NOTE: This example shows version 1.0.30. The actual file downloaded may not be this version but the procedure is the same.

After downloading the PME_DTM_Library_SETUP_1_0_30.EXE file, run it to begin the installation.

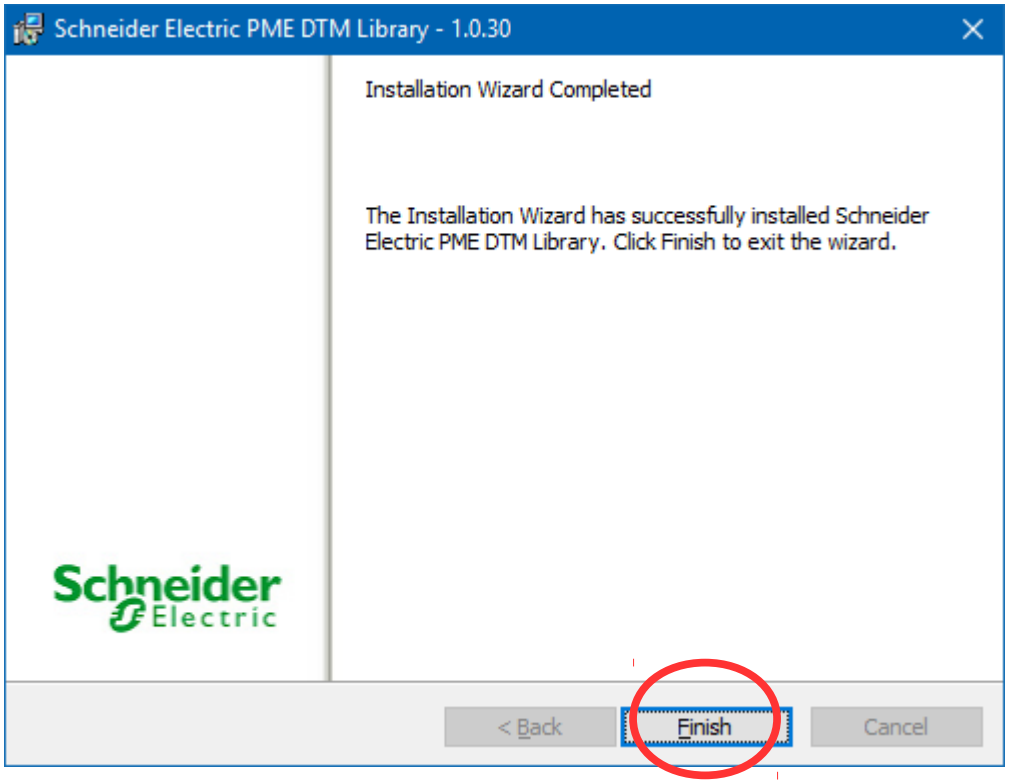


The self extractor will install the setup files into the c:\Niobrara\ folder and automatically start the S-E installation wizard.

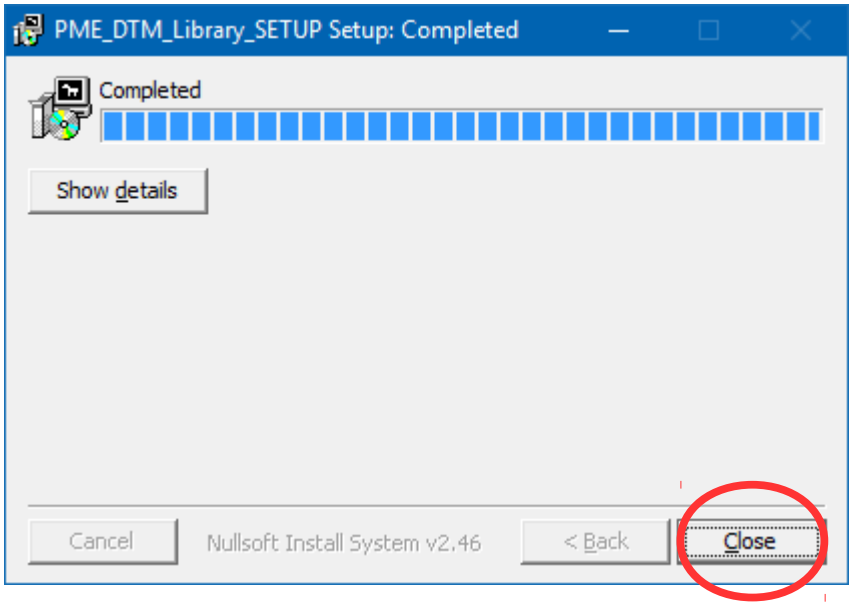
If you have a previous version of the PME DTM Libraray installed, you may be prompted to Repair or Remove the previous installation. Select Repair.



When finished a screen like the following should be displayed.



After selecting "Finish", the S-E Wizard will close. Now close the PME_TCM_Libraray_Setup window to complete the setup.

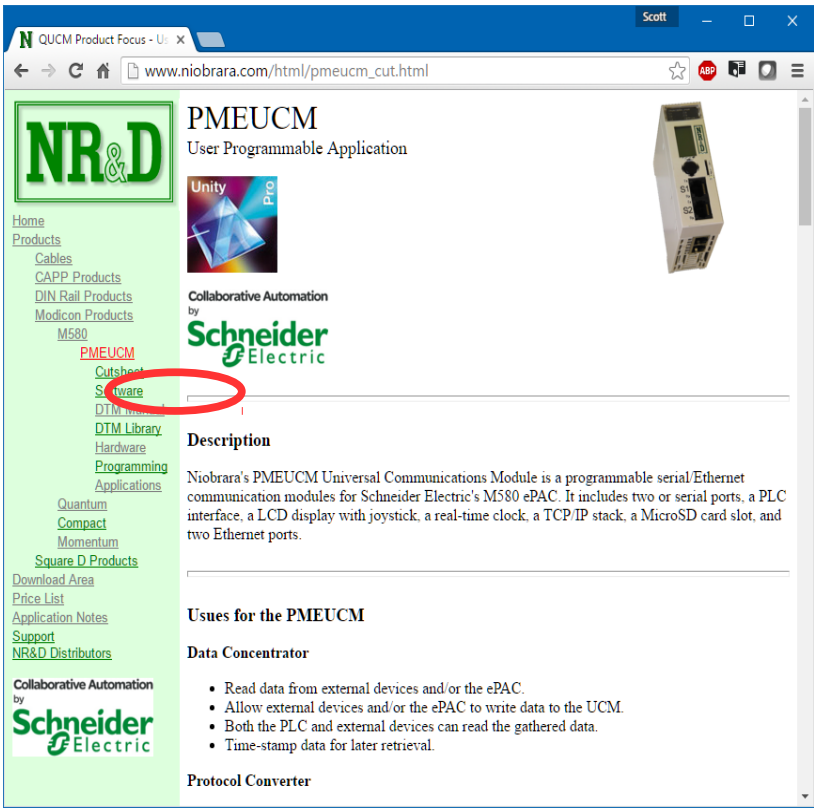


If you have not installed the PMEUCM_SETUP.EXE program and added the PMEUCM0302 to the Unity Hardware Catalog, proceed immediately to the next chapter before starting Unity Pro.

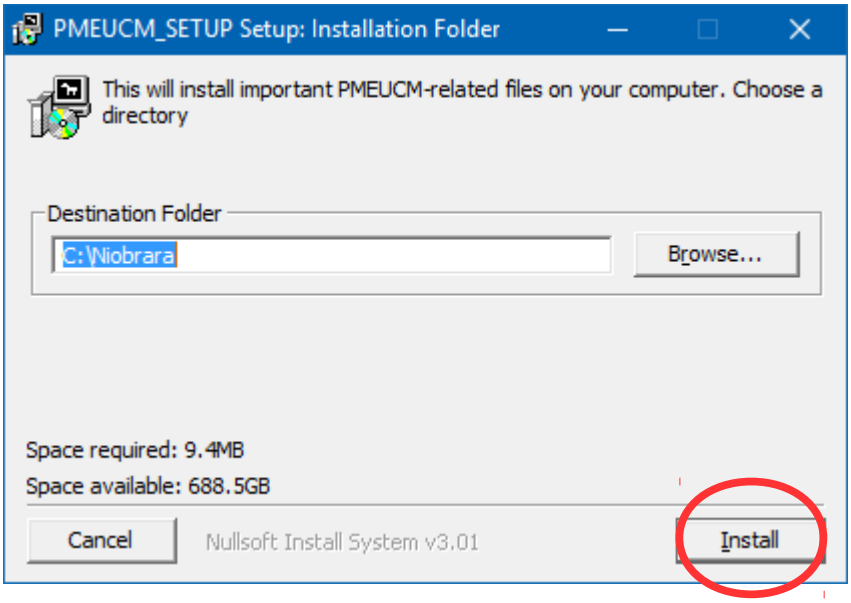
PMEUCM_SETUP.EXE

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



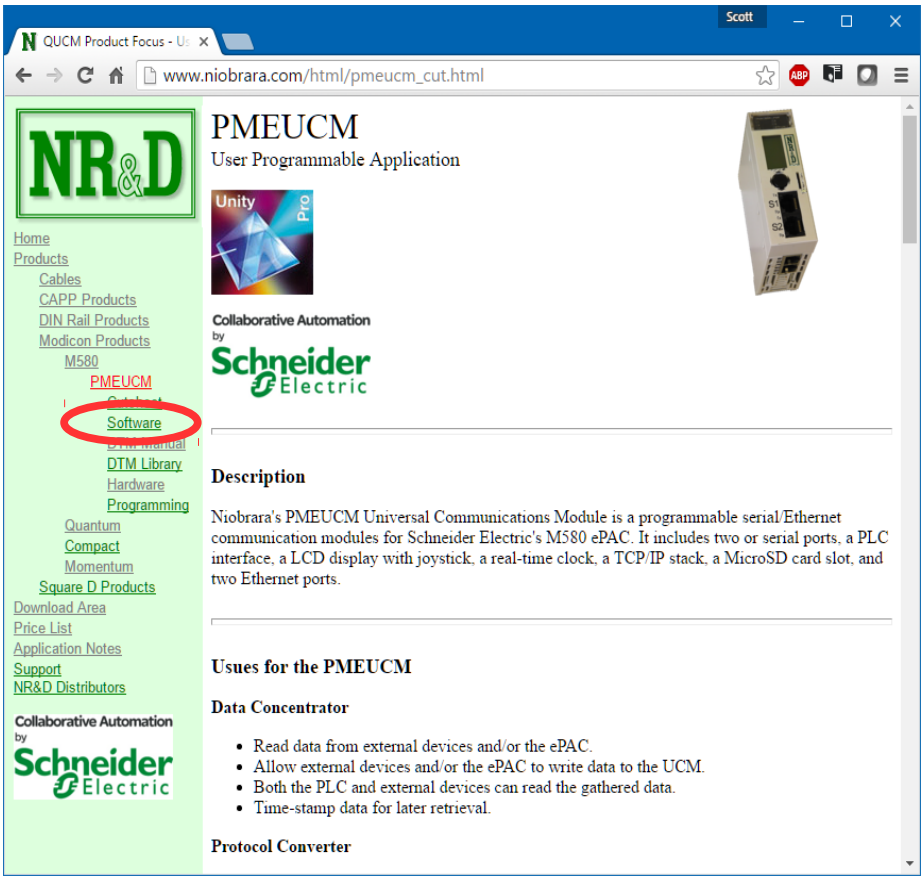
Download and run PMEUCM_SETUP.EXE. A box will appear prompting the user to choose a directory in which to install. The default is C:\Niobrara, as shown below.


PMEUCM_SETUP Setup: Installation Folder
This will install important PMEUCM-related files on your computer. Choose a directory
Destination Folder
C:\Niobrara
Browse...
Space required: 9.4MB
Space available: 688.5GB
Cancel
Nullsoft Install System v3.01
Install

4 Adding the PMEUCM to Unity Pro

Unity Pro versions 8.1 and higher provide a method for adding third party modules to their hardware catalog. Niobrara provides the necessary .cpx file as part of PMEUCM_SETUP.EXE. The user may access this file at:

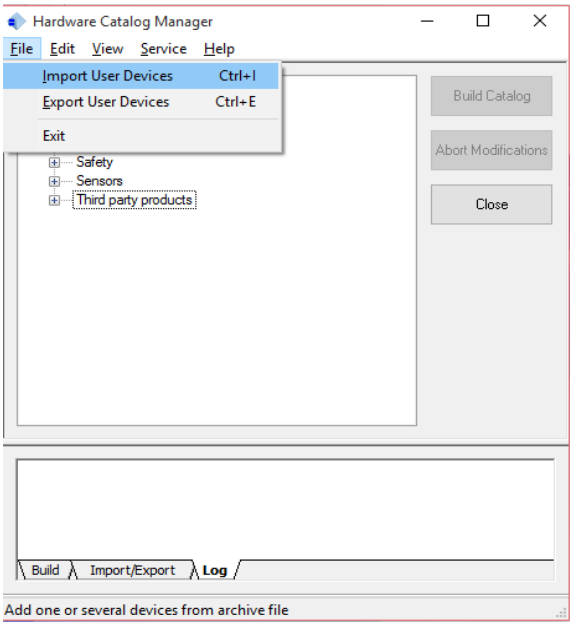
http://www.niobrara.com/html/pmeucm_cut.html



Adding the PMEUCM to Unity Hardware Catalog Manager

NOTE: Unity Pro must be not be running to access the Hardware Catalog Manager.

After the setup program is finished, start the Hardware Catalog Manager, located at Start>All Programs>Schneider Electric. In the File menu, click on Import User Devices, as shown below.



Choose the folder where PMEUCM_SETUP.EXE installed the .cpx file.

This is normally the 'c:\Niobrara\PMEUCM\DTM\' folder.

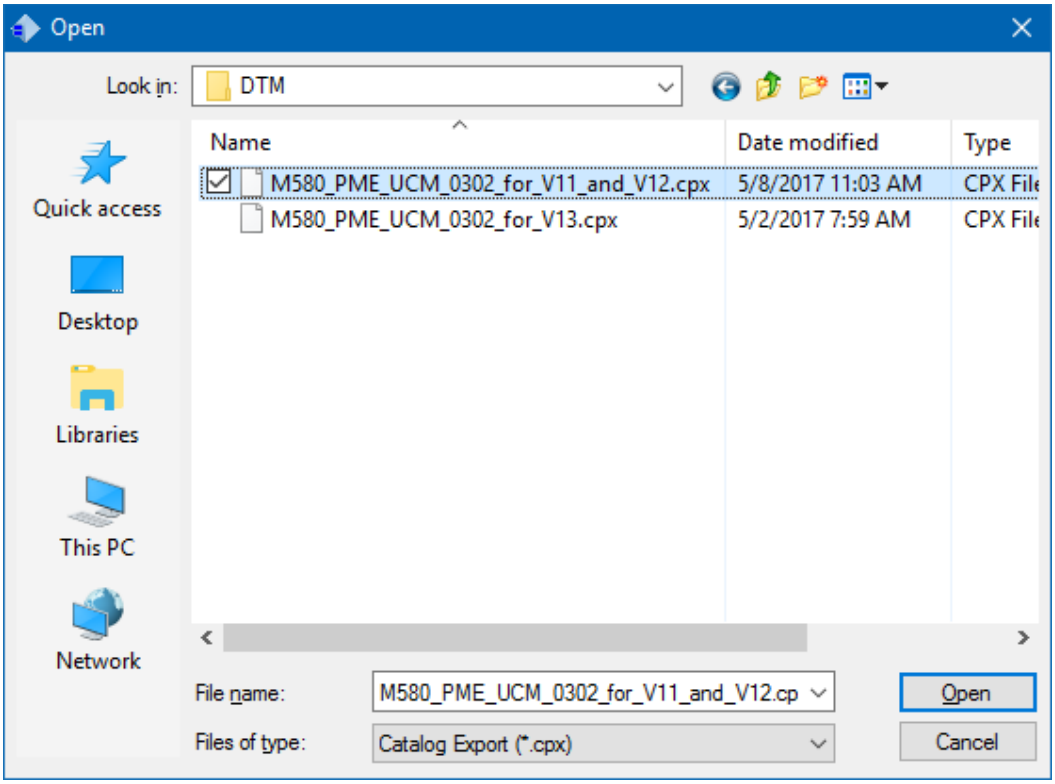
Inside the DTM folder is a file for Unity V11, V11.1, and V12:

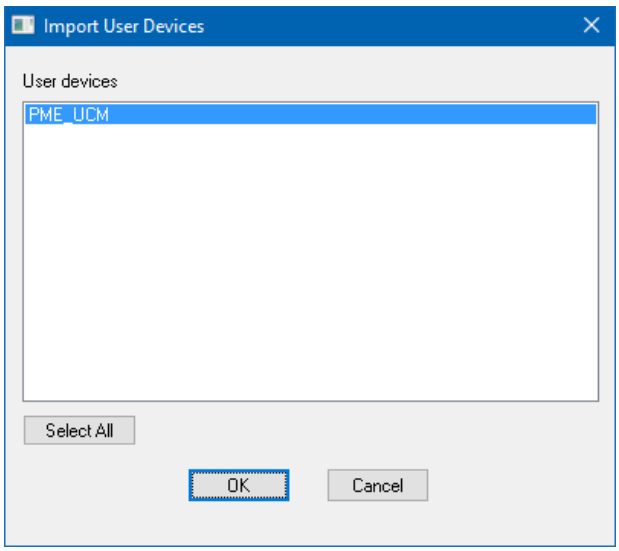
M580_PME_UCM_0302_for_V11_and_V12.cpx

Also present is a cpx file for Unity V13:

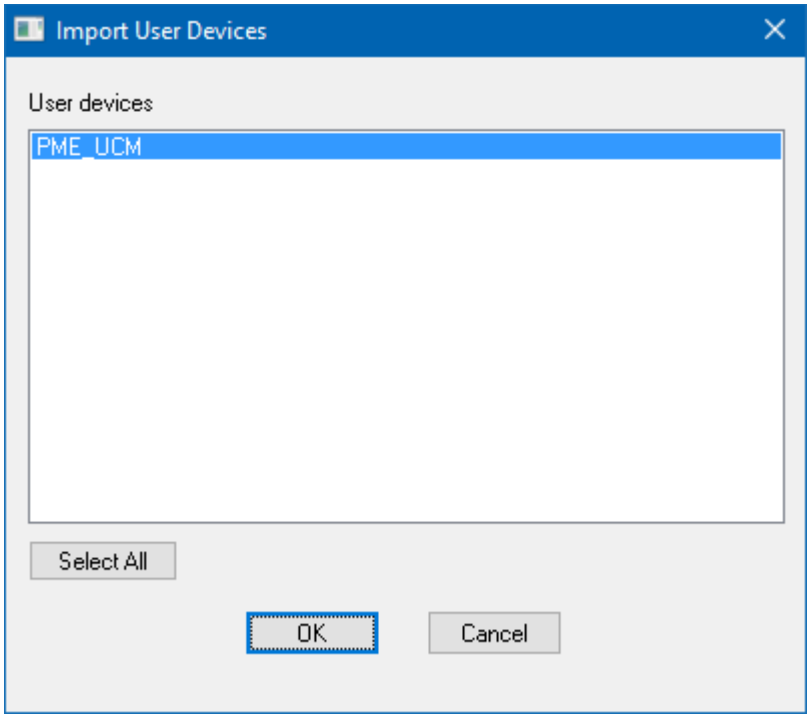
M580_PME_UCM_0302_for_V13.cpx

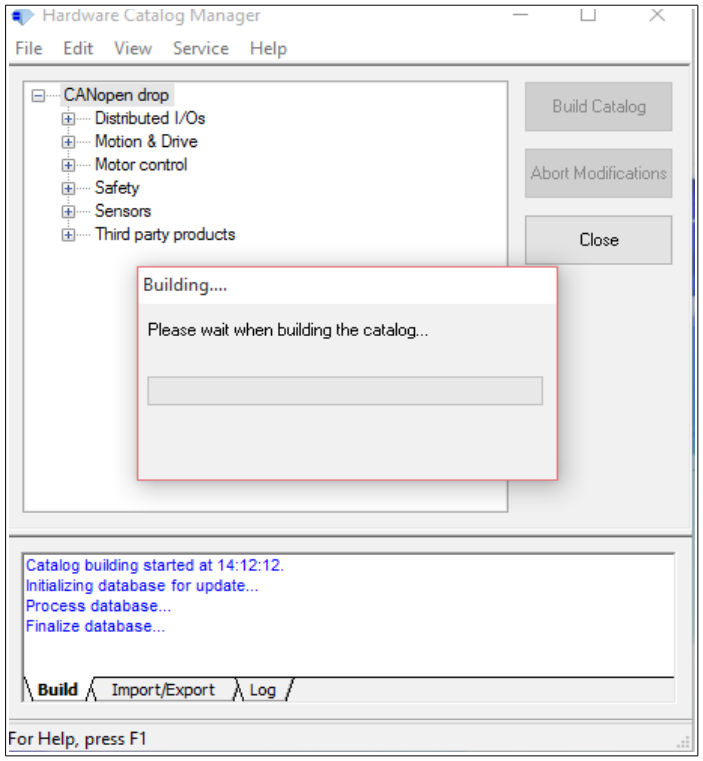
Choose the appropriate file for the installed version of Unity Pro, then click ‘Open’.


Open
Look in:
DTM
Name
Date modified
Type
M580_PME_UCM_0302_for_V11_and_V12.cpx
5/8/2017 11:03 AM
CPX File
M580_PME_UCM_0302_for_V13.cpx
5/2/2017 7:59 AM
CPX File
Quick access
Desktop
Libraries
This PC
Network
File name:
M580_PME_UCM_0302_for_V11_and_V12.cp
Open
Files of type:
Catalog Export (*.cpx)
Cancel

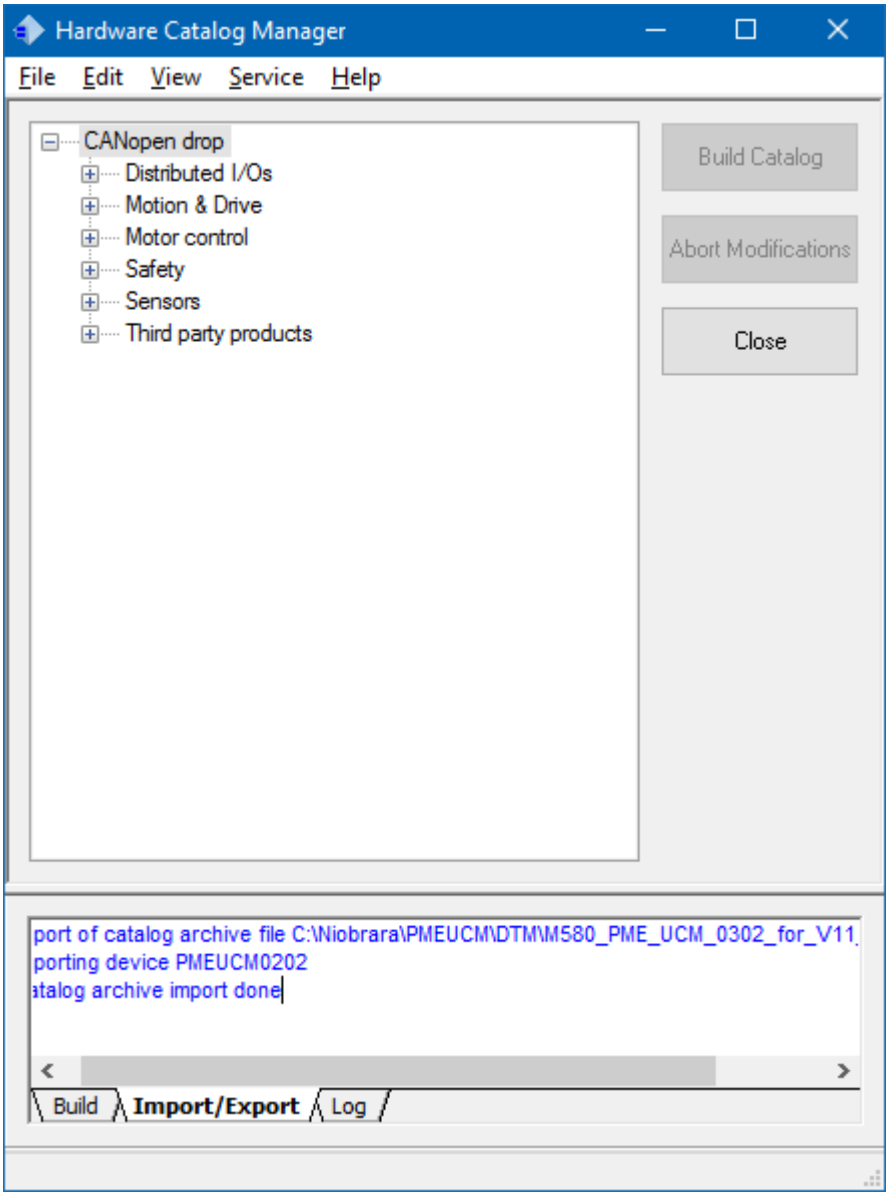

Import User Devices
User devices
PME_UCM
Select All
OK
Cancel

The Hardware Catalog Manager will show a dialog box displaying its progress.

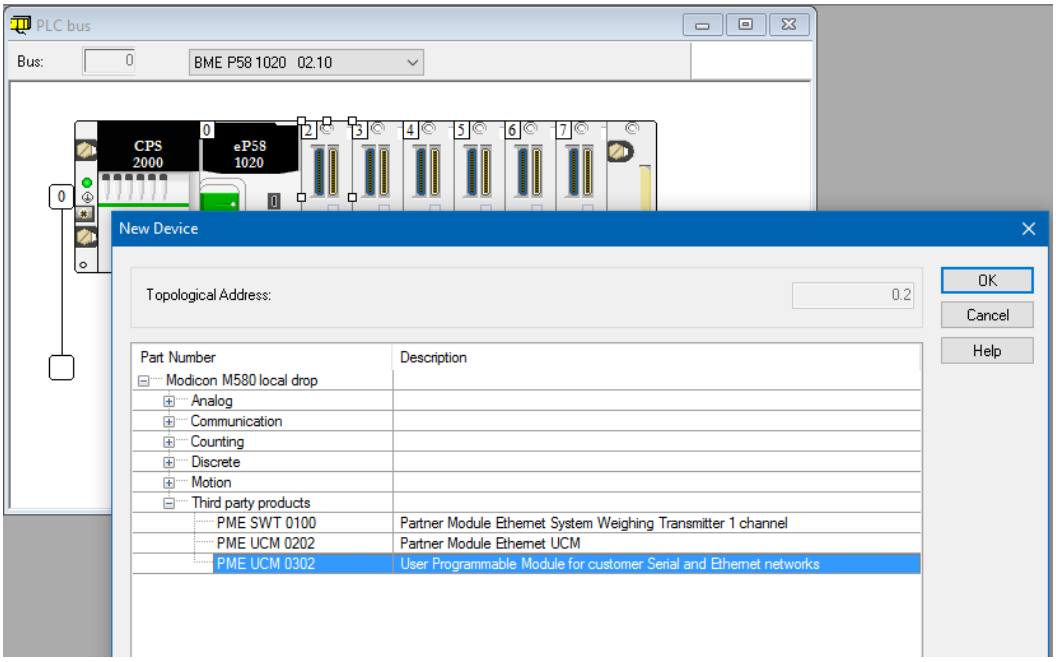




When it is finished, it will appear as below.



Close the Hardware Catalog Manager, and start Unity Pro. The PMEUCM0302 can now be chosen from the Hardware Catalog under the “Third party products” section.


PLC bus
Bus:
0
BME P58 1020 02.10
CPS 2000
eP58 1020
New Device
Topological Address:
0.2
OK
Cancel
Help
Part Number
Description
Modicon M580 local drop
Analog
Communication
Counting
Discrete
Motion
Third party products
PME SWT 0100
Partner Module Ethernet System Weighing Transmitter 1 channel
PME UCM 0202
Partner Module Ethernet UCM
PME UCM 0302
User Programmable Module for customer Serial and Ethernet networks

5 Designing the TXT File

A comma separated variable text file (TXT) is used by the NRDPTKDTMUTIL.EXE program to generate the custom DDXML file used by Unity Pro's Generic PTK DTM. This txt file is simple to configure using a standard text editor like MS Notepad.

The following rules are enforced by the compiler utility:

- The filename of the txt file must have a single underscore "_".
  - The filename should be of the format "PME UCM 0202_name.txt" where "name" is the project name.
  - More than one underscore is not allowed.
  - Decimal points and commas are not allowed.
  - Spaces are allowed.
- The format of each line is tagname, tagdata followed by a new line.
- Only one tagname is allowed per line.
- Leading spaces and Tab characters before the tagname are ignored.
- Tab characters between the comma and the tagdata are ignored.
- Leading spaces between the comma and the tagdata are ignored.
- Trailing spaces between the tagdata and new line are ignored.
- Empty lines are ignored.
- Some tagnames include modifiers that are separated by commas.
- Commas are not allowed inside tagdata. Use a semi-colon instead.
- Tagnames are not case sensitive.
- Tagdata may be case sensitive.
- assemblyID and assemblyPath tagnames are required as a pair. These tagnames must be adjacent to each other.
- There must be at least two assemblyID and assemblyPath pairs.
- The varType 'STRING' must be preceded by the varLength tagname.

- Language fields must be presented in the same order throughout the document. For example, if 'En-En' is listed first, followed by 'Fr-Fr' then this order must be followed whenever both EN and FR are referenced by a description.
- If the very first character of a line is #, the line is ignored. This may be used to add comments.
- The 'Header' SW is the only required header value.

Header Information

The first block of information includes data about the application.

Tagname	Description	Default Value if Ommited
fileCreator	Name of the person or company building the file.	NRDDTMUTIL
fileModifiedBy	Name of person or company modifying the file.	NRDDTMUTIL
fileCreationDate	Hardcoded date or 'now' to use the PC's time/date.	'now'
fileModifiedDate	Hardcoded date or 'now' to use the PC's time/date.	'now'
MinCompiler	Datecode of required DTM Utility	Current version of the DTM Utility
productID	Must start with 'PME UCM 0202' followed by a unique identifier for this DTM.	'PME UCM 0202 name' where 'name' is the text following the underscore in the source file name.
productText	Usually the text following the 0202 in the productID	'name' i.e. the text following the underscore in the source file name.
SW	Number of the software version. This is the value shown in the Unity Pro DTM Browser as the 'version'.	Required. Should be of the form xx.xx normally 01.00
FW	Set to match the Firmware Version of the UCM board. (Ignored at this time)	01.00
HW	Set to match the Hardware Version of the UCM board. (Ignored at this time)	01.00

Assembly Information

The data is transported between the M580 CPU and the PMEUCM using an Ethernet/IP explicit data connection. Each of these connections is limited to a total of around 1400 bytes each in and out. This 1400 byte limit includes the header information tacked on by the DTM. Sometimes it is desirable to have more than one Ethernet/IP data connection between the M580 and the PMEUCM. The assemblyID and assemblyPath tagnames are used to define these connections. The update timing of these assemblies may be set using the assemblyDefaultRPI tagname.

Rules for assemblyID

- The first group of InputVars must be assemblyID = 1.
- The first group of OutputVars must be assemblyID = 2.
- assemblyIDs must be sequential with InputVars EVEN numbers while OutputVars must be ODD numbers
  - InputVars follow the sequence 1, 3, 5, 7
  - OutputVars follow the sequence 2, 4, 6, 8
  - The maximum assemblyID is 8.
- assemblyID settings are valid in a top-down fashion. Variables defined after an assemblyID declaration are included in that assembly until the next assemblyID statement is reached.

Rules for assemblyPath

- The first InputVars assemblyPath must be 101.
- The first OutputVars assemblyPath must be 101.
- assemblyPath settings are valid in a top-down fashion. Variables defined after an assemblyPath declaration are included in that assembly until the next assemblyPath statement is reached.

Rules for RPI

- The RPI for the assembly group may be optionally set using assemblyDefaultRPI.
- The minimum supported RPI is 4mS.
- The maximum supported RPI is 500mS.
- The tagname assemblyMinRPI can be used to limit the lowest settable RPI on a connection. This may be useful for restricting the bandwidth used by a secondary connection.
- The tagname assemblyMaxRPI may be used to limit the highest settable RPI for a connection. This may be useful for restricting the update time of the Primary Connection.
- RPI settings apply to both input and output channels. It is only necessary to set RPI commands on the INPUT assemblies. The DTM Utility will automatically apply RPI settings to the OUTPUT assembly.
- The following default settings are used if the RPI tagnames are omitted:
  - Assembly 1,2
    - Default = 10mS
    - Min = 4 mS
    - Max = 500mS
  - Assembly 3,4
    - Default = 50mS
    - Min = 4mS
    - Max = 500mS
  - Assembly 5,6
    - Default = 100mS
    - Min = 20mS
    - Max = 500mS
  - Assembly 7,8
    - Default = 300mS
    - Min = 40mS
    - Max = 500mS

Example assemblyPath for single Ethernet/IP half-duplex

```
SW,                         1.00

assemblyID,                 1
assemblyPath,               101
varHeading,                 InputVars
       varType,             WORD
       varName,             UCM_Runtime_Status
       varLabel,en-us,      .15=Run; LSB=Runtime Error
       v
varHeading,                 InputVars
       varType,             UINT
       varName,             UCM_Halt_Line_Number
       varLabel,en-us,      UCM Runtime Error Halt Line Number
       v
varHeading,                 InputVars
       varType,             INT
       varName,             In_01
       varLabel,en-us,      In_01

assemblyID,                 2
assemblyPath,               102
varHeading,                 outputVars
       varType,             REAL
       varName,             out_01
       varLabel,en-us,      out_01

varHeading,                 outputVars
       varType,             DINT
       varName,             out_02
       varLabel,en-us,      out_02

varHeading,                 outputVars
       varType,             UINT
       varName,             out_03
       varLabel,en-us,      out_03

varHeading,                 outputVars
       varType,             UINT
       varName,             out_04
       varLabel,en-us,      out_04
```

Example assemblyPath for single Ethernet/IP full-duplex

```
SW,                         1.00

assemblyID,                 1
assemblyPath,               101
varHeading,                 InputVars
       varType,             WORD
       varName,             UCM_Runtime_Status
       varLabel,en-us,      .15=Run; LSB=Runtime Error

varHeading,                 InputVars
       varType,             UINT
       varName,             UCM_Halt_Line_Number
       varLabel,en-us,      UCM Runtime Error Halt Line Number

varHeading,                 InputVars
       varType,             INT
       varName,             In_01
       varLabel,en-us,      In_01

assemblyID,                 2
assemblyPath,               101
varHeading,                 outputVars
       varType,             REAL
       varName,             out_01
       varLabel,en-us,      out_01

varHeading,                 outputVars
       varType,             DINT
       varName,             out_02
       varLabel,en-us,      out_02

varHeading,                 outputVars
       varType,             UINT
       varName,             out_03
       varLabel,en-us,      out_03

varHeading,                 outputVars
       varType,             UINT
       varName,             out_04
       varLabel,en-us,      out_04
```

Notice that the both the first and second assemblyPaths are set to 101.

Example assemblyPath for dual Ethernet/IP full-duplex

```
SW,                         1.00

assemblyID,                 1
assemblyPath,               101
varHeading,                 InputVars
       varType,             WORD
       varName,             UCM_Runtime_Status
       varLabel,en-us,      .15=Run; LSB=Runtime Error

varHeading,                 InputVars
       varType,             UINT
       varName,             UCM_Halt_Line_Number
       varLabel,en-us,      UCM Runtime Error Halt Line Number

varHeading,                 InputVars
       varType,             INT
       varName,             In_01
       varLabel,en-us,      In_01


assemblyID,                 3
assemblyPath,               102
varHeading,                 InputVars
       varType,             INT
       varName,             In_02
       varLabel,en-us,      In_02


assemblyID,                 2
assemblyPath,               101
varHeading,                 outputVars
       varType,             REAL
       varName,             out_01
       varLabel,en-us,      out_01

varHeading,                 outputVars
       varType,             DINT
       varName,             out_02
       varLabel,en-us,      out_02


assemblyID,                 4
assemblyPath,               102
varHeading,                 outputVars
       varType,             UINT
       varName,             out_03
       varLabel,en-us,      out_03

varHeading,                 outputVars
       varType,             UINT
       varName,             out_04
       varLabel,en-us,      out_04
```

Language Tags Rules

- Tagnames that require language tags must include a single language tag.
- Tagnames requiring language tags must be inserted in the same order throughout the txt file.
- Language tags are not case sensitive.
- Language tags follow the tagname with a comma.
- The following Language Tags are valid:

Tag	Language
En or en-us	English (United States)
Fr or fr-fr	French (standard)
De or de-de	German (standard)
It or it-it	Italian (standard)
pt-br	Portuguese (Brazil)
Es or es-es	Spanish (standard)

Examples of Language tagnames

varLabel,en-us, Gross Measurement

varLabel,fr-fr, Mesure en Brut

Input Data

Rules for Input Data

- VarHeading must be the first tagname
- varType STRING must be preceded by varLength
- varTypes REAL, DINT, UDINT, and DWORD must be aligned on a 4-byte boundary starting from the first byte assigned.
- VarTypes INT, UINT, and WORD must be aligned on a 2-byte boundary starting from the first byte assigned.
- A maximum of 428 bytes of Input data may be defined in the Primary (first) Ethernet/IP assembly. The 72 byte PTK header is always included in the first input assembly.
- A maximum of 500 bytes of Input data may be define for assemblies 2, 4, 6, and 8.
- The first variable must be UCM_Runtime_Status. The UCM operating system always places this WORD variable into the PLC INPUT data

structure.

- The second variable must be UCM_Halt_Line_Number. Again, the OS places this UINT value into the PLC INPUT data structure.

Input Data Tagnames (Required)

<table>
<tr><th>Tagname</th><th>Value</th><th>Notes</th></tr>
<tr><td>varHeading</td><td>InputVars</td><td>Must be the first field for a variable.</td></tr>
<tr><td>varLength</td><td>Length of string that follows</td><td>Required for varType=STRING</td></tr>
<tr><td>varType</td><td><table><tr><th>Type</th><th>Byte Length</th></tr><tr><td>BOOL</td><td>1</td></tr><tr><td>BYTE</td><td>1</td></tr><tr><td>WORD</td><td>2</td></tr><tr><td>DWORD</td><td>4</td></tr><tr><td>REAL</td><td>4</td></tr><tr><td>STRING</td><td>variable</td></tr><tr><td>INT</td><td>2</td></tr><tr><td>DINT</td><td>4</td></tr><tr><td>UINT</td><td>2</td></tr><tr><td>UDINT</td><td>4</td></tr></table></td><td></td></tr>
<tr><td>varName</td><td>Text Field</td><td></td></tr>
<tr><td>varLabel</td><td>Text Field</td><td>Requires Language tag</td></tr>
<tr><td>varDescription</td><td>Text Field</td><td>Requires Language tag
Defaults to varLabel if ommited.</td></tr>
<tr><td>varAccess</td><td><table><tr><th>Type</th></tr><tr><td>Read</td></tr><tr><td>Write</td></tr><tr><td>ReadWrite</td></tr><tr><td>ConditionalReadWrite</td></tr><tr><td>noAcccess</td></tr></table></td><td>Defaults to 'Read' for Input, 'Read/Write' for Output if ommited.</td></tr>
<tr><td>varPersistent</td><td><table><tr><th>Type</th></tr><tr><td>True</td></tr><tr><td>False</td></tr></table></td><td>Defaults to 'True' for Input and 'False for Output if ommited.</td></tr>
</table>

Input Data Tagnames (Optional)

Tagname	Value	Notes
varDefault	Signed Decimal number	Sets the default value for the variable.
varLowLimit	Signed Decimal number	Sets the lowest allowed value.
varHiLimit	Signed Decimal number	Sets the highest allowed value.
varWarning	Text Field	Requires Language tag
varDescription	Text Field	Requires Language tag
varUnit	Text Field	

Output Data

Output Data Tagnames (Required)

<table>
<tr><th>Tagname</th><th>Value</th><th>Notes</th></tr>
<tr><td>varHeading</td><td>InputVars</td><td>Must be the first field for a variable.</td></tr>
<tr><td>varLength</td><td>Length of string that follows</td><td>Required for varType=STRING</td></tr>
<tr><td>varType</td><td><table><tr><th>Type</th><th>Byte Length</th></tr><tr><td>BOOL</td><td>1</td></tr><tr><td>BYTE</td><td>1</td></tr><tr><td>WORD</td><td>2</td></tr><tr><td>DWORD</td><td>4</td></tr><tr><td>REAL</td><td>4</td></tr><tr><td>STRING</td><td>variable</td></tr><tr><td>INT</td><td>2</td></tr><tr><td>DINT</td><td>4</td></tr><tr><td>UINT</td><td>2</td></tr><tr><td>UDINT</td><td>4</td></tr></table></td><td></td></tr>
<tr><td>varName</td><td>Text Field</td><td></td></tr>
<tr><td>varLabel</td><td>Text Field</td><td>Requires Language tag</td></tr>
<tr><td>varDescription</td><td>Text Field</td><td>Requires Language tag
Defaults to varLabel if ommited.</td></tr>
<tr><td>varAccess</td><td><table><tr><th>Type</th></tr><tr><td>Read</td></tr><tr><td>Write</td></tr><tr><td>ReadWrite</td></tr><tr><td>ConditionalReadWrite</td></tr><tr><td>noAcccess</td></tr></table></td><td>Defaults to 'Read' for Input, 'Read/Write' for Output if ommited.</td></tr>
<tr><td>varPersistent</td><td><table><tr><th>Type</th></tr><tr><td>True</td></tr><tr><td>False</td></tr></table></td><td>Defaults to 'True' for Input and 'False for Output if ommited.</td></tr>
</table>

Output Data Tagnames (Optional)

Tagname	Value	Notes
varDefault	Signed Decimal number	Sets the default value for the variable.
varLowLimit	Signed Decimal number	Sets the lowest allowed value.
varHiLimit	Signed Decimal number	Sets the highest allowed value.
varWarning	Text Field	Requires Language tag
varDescription	Text Field	Requires Language tag
varUnit	Text Field	

Configuration Data

Variable Arrays

Certain variable types may be defined as arrays. To declare a variable as an array, simply add a ‘,’ followed by the starting index ‘0’, then another ‘,’ followed by the ending index.

NOTE: The current version of the DTM only supports a starting index of 0.

Array Variable Type	Example	Result
BYTE	VarType, BYTE,0,99	Array of BYTE[0..99]
WORD	VarType, WORD,0,9	Array of WORD[0..9]
DWORD	VarType, DWORD,0,4	Array of DWORD[0..4]
INT	VarType, INT,0,9	Array of INT[0..9]
DINT	VarType, DINT,0,9	Array of DINT[0..9]
UINT	VarType, UINT,0,3	Array of UINT[0..3]
UDINT	VarType, UDINT,0,9	Array of UDINT[0..9]
REAL	VarType, REAL,0,10	Array of REAL[0..10]

Automatic Incremental Variables

It can become tedious to define a number of variables that share the same parameters and only differ by variable name. The DTM Utility supports automatic incremental naming of a variable where the name includes a three character number that varies from a starting point to a finish point.

```
# The following loop creates PLC_OUT_B000 through PLC_OUT_B009
# 0=Start, 9=End

varLoopStart, 0, 9

varHeading,           outputVars
       varType,       BYTE
       varName,       PLC_OUT_B%
       varLabel,en-us,Process data from PLC

varLoopEnd
```

VarLoopStart, x, y

VarLoopStart marks the beginning of an automatic variable generation. It includes two parameters. The first parameter is the starting value and the second is the ending value. The first value must be smaller than the second value. The maximum number of variables generated is 500.

VarName

The varName must include a '%' character at the end of the string. This % character is replaced by the 3 digit number generated by the varLoop. The number is always 3 digits and includes leading zeros.

VarLoopEnd

VarLoopEnd denotes the end of a VarLoopStart segment. The trailing comma and any parameter are ignored and may be ommited.

6 NRD PTK DTM UTIL

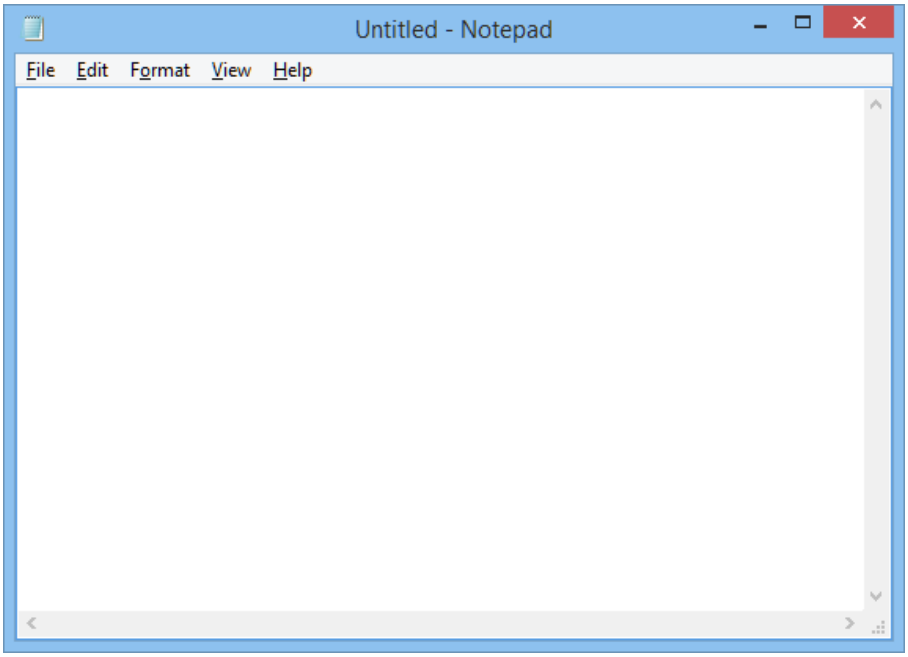
Editing the .TXT file

Any text editor (like MS Notepad) may be used to edit the .TXT file used to generate the DDXML for the PMEUCM.

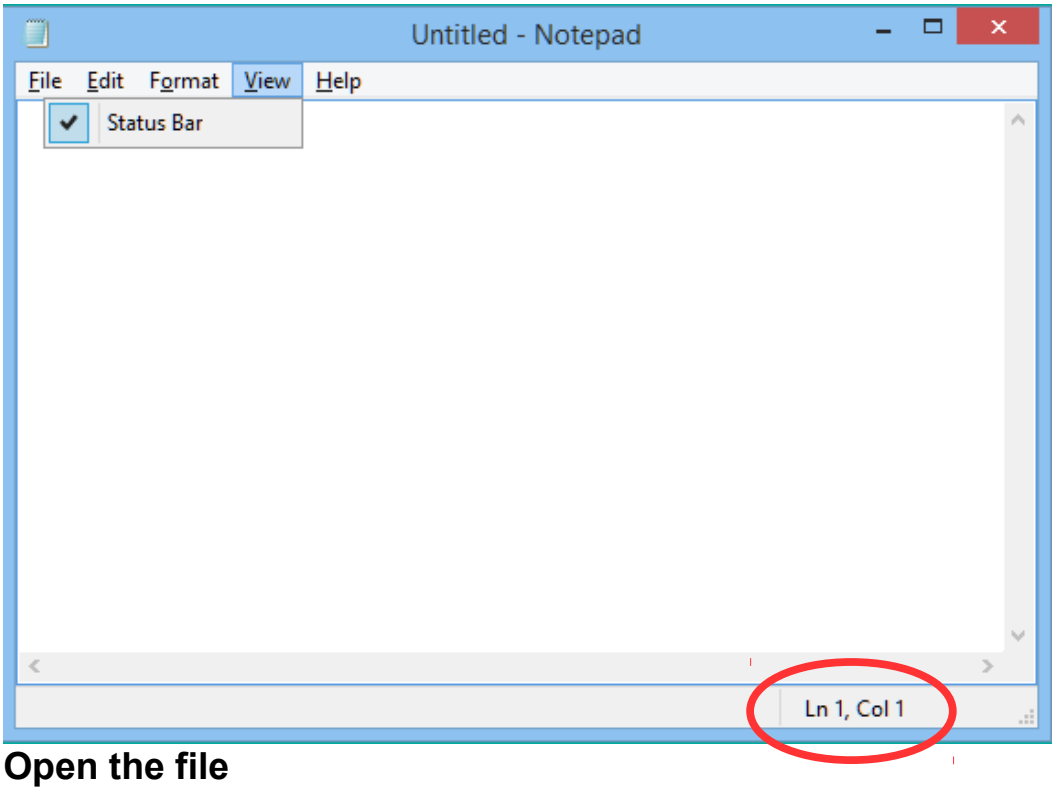
For this exmple, the file c:\Niobrara\DTM\PME UCM 0202_Example1.txt is used.

Start Notepad

The first step is to open Notepad. It is usually found under Start > Programs > Accessories > Notepad.



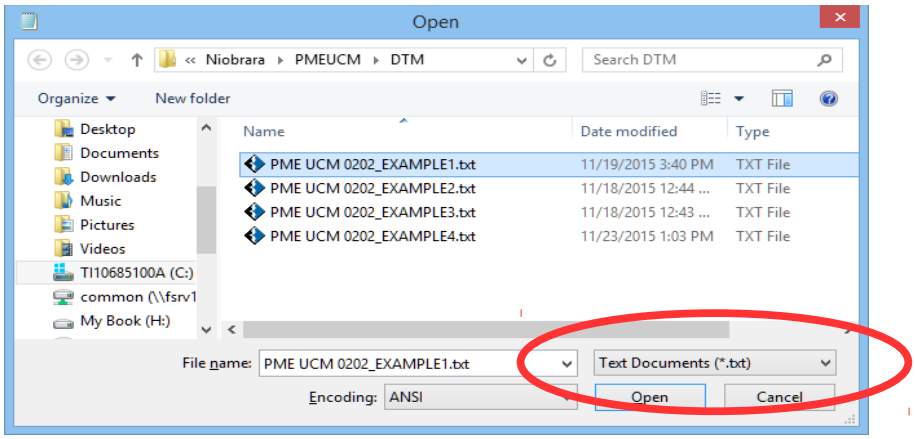
Next, it is a good idea to turn on the "Status Bar". This setting is found under "View, Status Bar". The Status Bar shows the line number and column location of the cursor. The line number is helpful if there is an error compiling the .txt file.

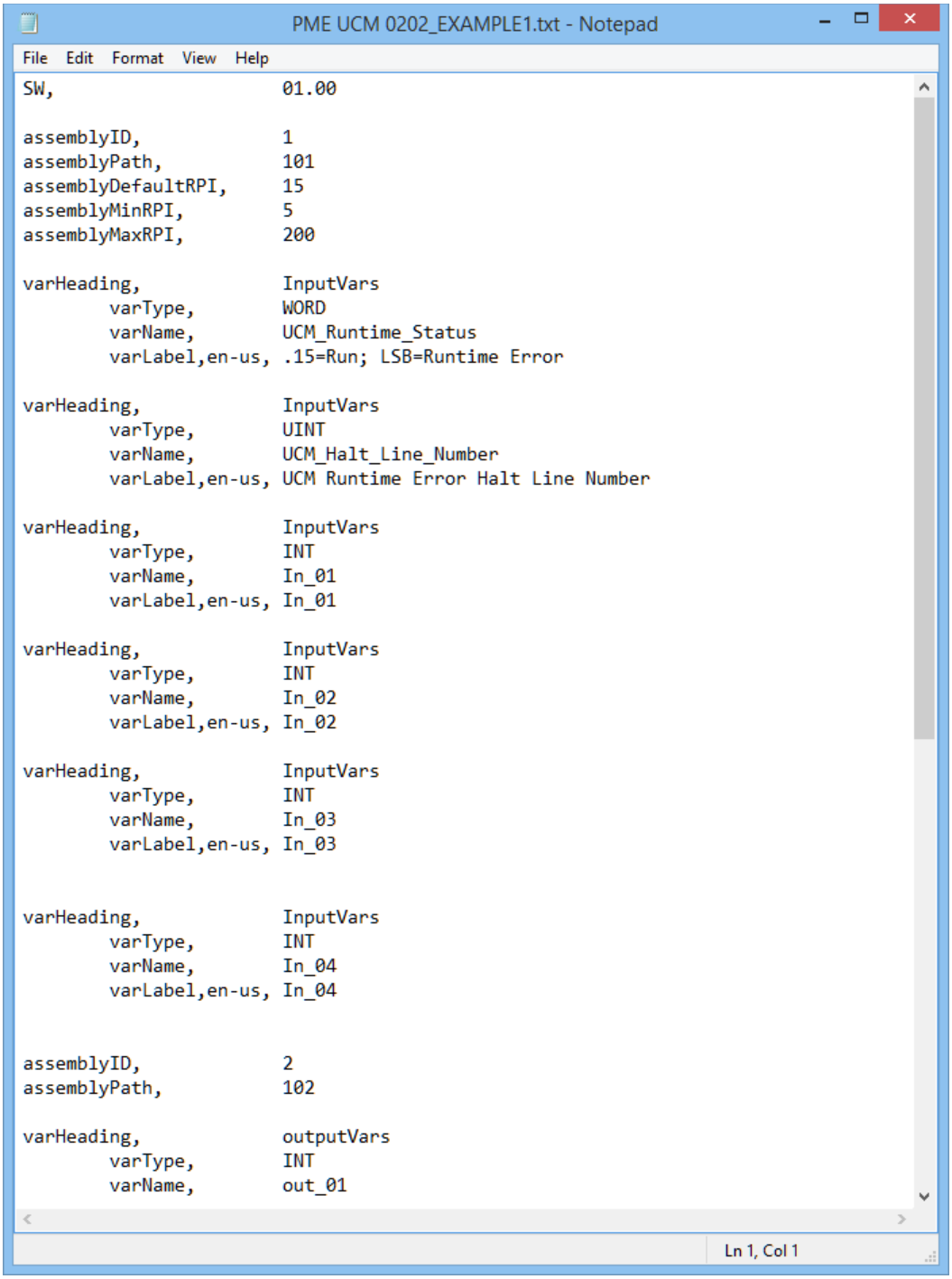


Open the file

Next select File, Open and then browse to c:\Niobrara\DTM\ and open the file:

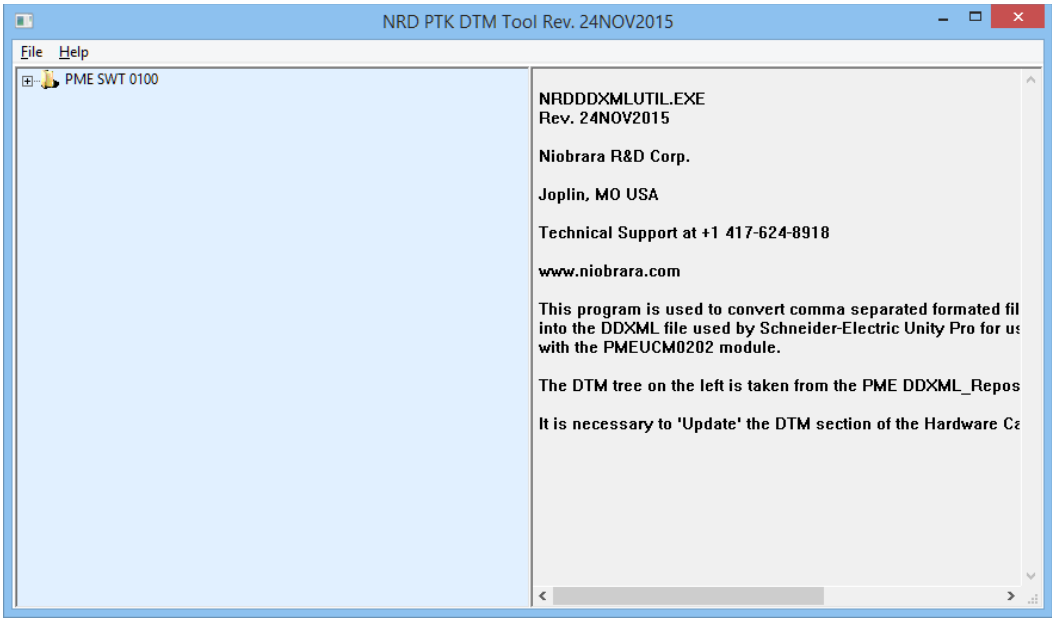
"PME UCM 0202_Example1.txt"




PME UCM 0202_EXAMPLE1.txt - Notepad
File Edit Format View Help
SW, 01.00
assemblyID, 1
assemblyPath, 101
assemblyDefaultRPI, 15
assemblyMinRPI, 5
assemblyMaxRPI, 200
varHeading, InputVars
varType, WORD
varName, UCM_Runtime_Status
varLabel,en-us, .15=Run; LSB=Runtime Error
varHeading, InputVars
varType, UINT
varName, UCM_Halt_Line_Number
varLabel,en-us, UCM Runtime Error Halt Line Number
varHeading, InputVars
varType, INT
varName, In_01
varLabel,en-us, In_01
varHeading, InputVars
varType, INT
varName, In_02
varLabel,en-us, In_02
varHeading, InputVars
varType, INT
varName, In_03
varLabel,en-us, In_03
varHeading, InputVars
varType, INT
varName, In_04
varLabel,en-us, In_04
assemblyID, 2
assemblyPath, 102
varHeading, outputVars
varType, INT
varName, out_01
Ln 1, Col 1

Open the NRD DTM Tool

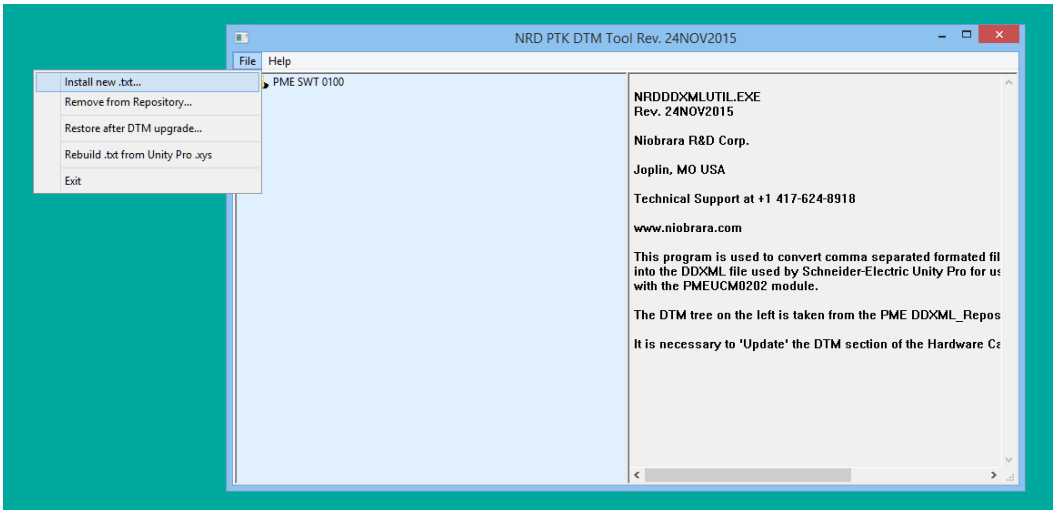
The next step is to open the Niobrara DTM Tool. Select Programs > Niobrara > PMEUCM > DTM > DTM Utility.

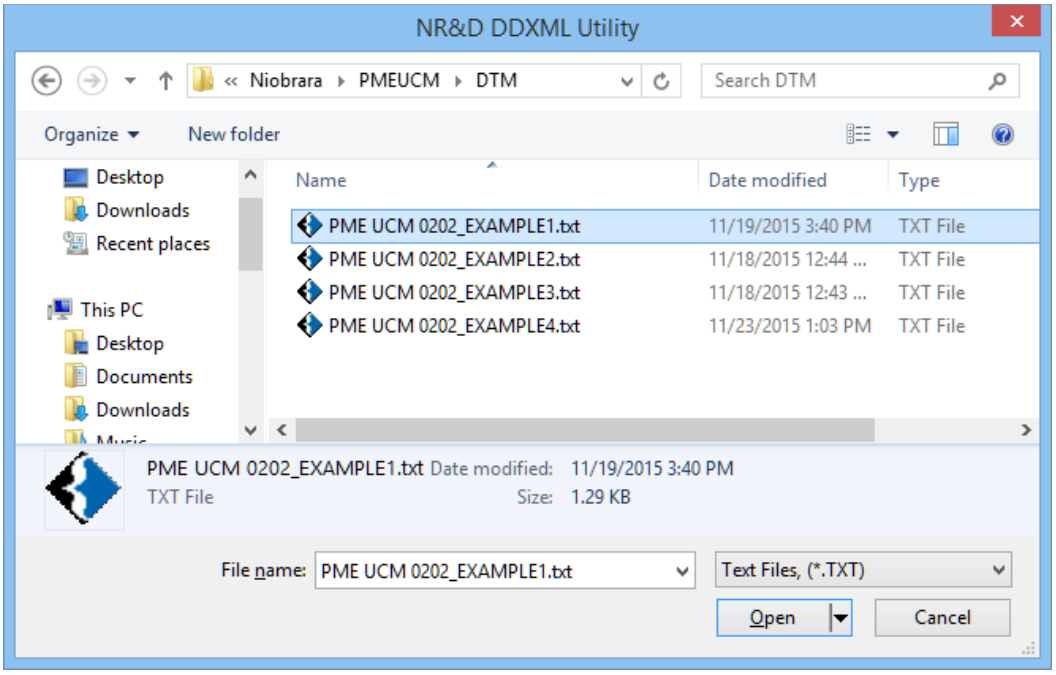


The tree on the left of the screen shows the PTK DTMs installed in Unity Pro. In this case, it shows the PME SWT 0100 Weighing Module from SCAIME.

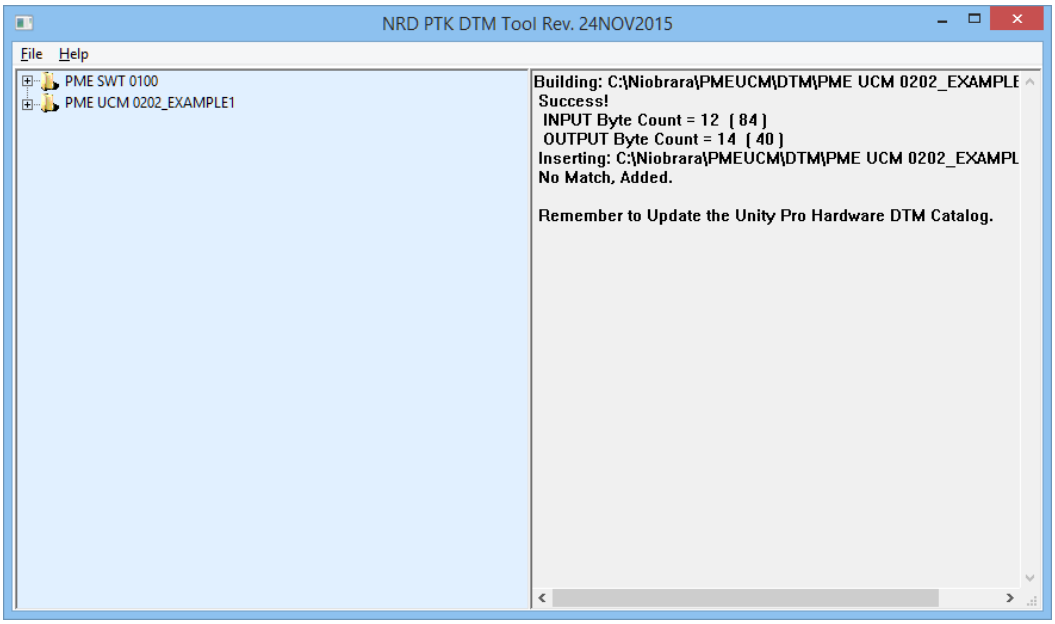
Installing a new file

Select File > "Install new .txt..." and then browse to the c:\Niobara\PMEUCM\DTM\ folder and select the file to install "PME UCM 0202_Example1.txt"

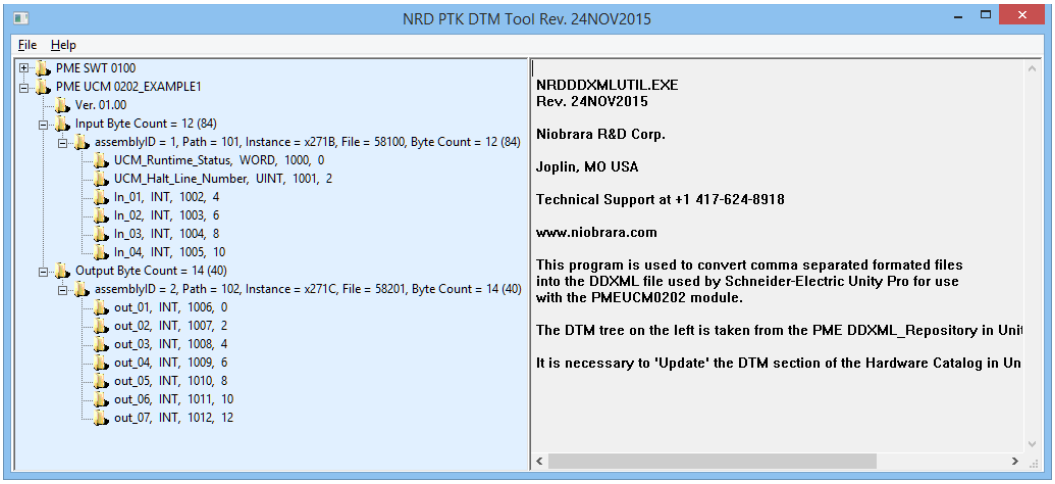




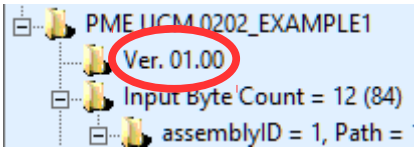
After selecting “Open”, the main screen should now change to show a new entry in the tree.



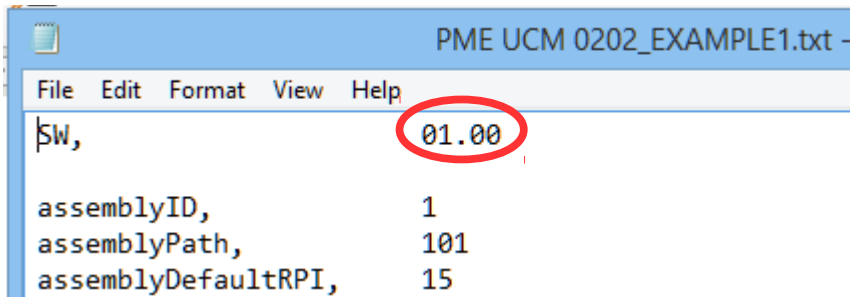
Status information is displayed on the right side of the screen. If there is an error during the compile, the error description and source code line number will be displayed.



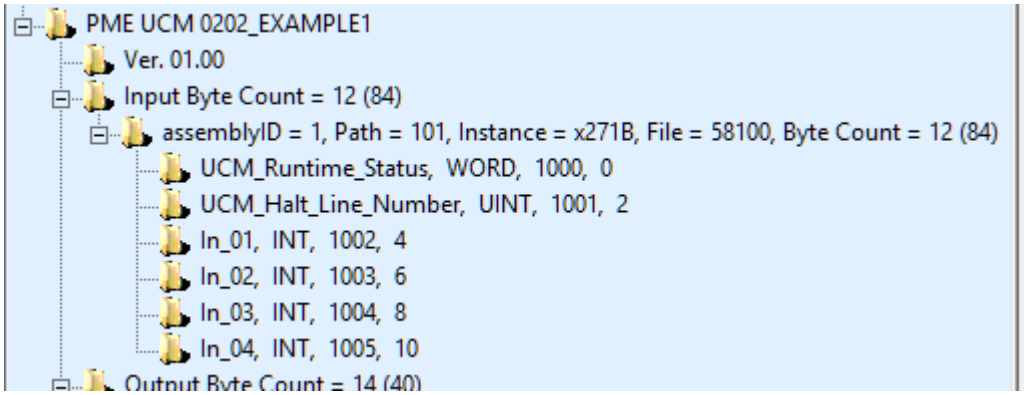
Expanding the tree for the bptest3in4out entry shows quite a few details.
The SW Version number is shown:



This is the value from the txt file:



Next is the structure of the PLC INPUT data.

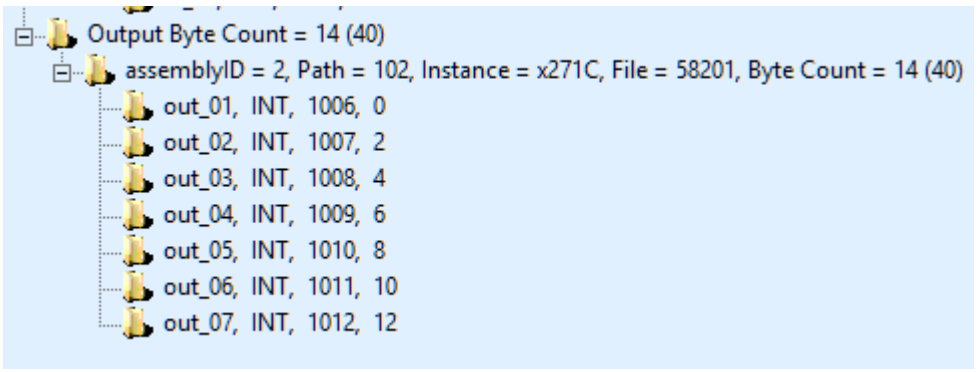


The heading shows the total number of bytes of PLC INPUT data. In this case there are 12 bytes of data. The number following (84) gives the total number of byes including the PTK header information.

The next segment is the assemblyID information. The assemblyID = 1 and Path = 101 are from the .txt file. The Instance = x271B, File = 58100, and Byte Count = 12 values are useful in configuring the UCM application. AssemblyIDs 1 and 2 also include the total byte count with the header information (+72 bytes for inputs and +26 bytes for outputs).

The variables start with the UCM_Runtime_Status. This is a WORD variable. The number 1000 that follows is the Ethernet/IP reference number assigned to this variable. These numbers always start at 1000 and are automatically generated by the DTM Utility. The value 0 that follows the 1000 is the byte offset from the beginning of the structure. The offset value is useful when doing the UCM programming.

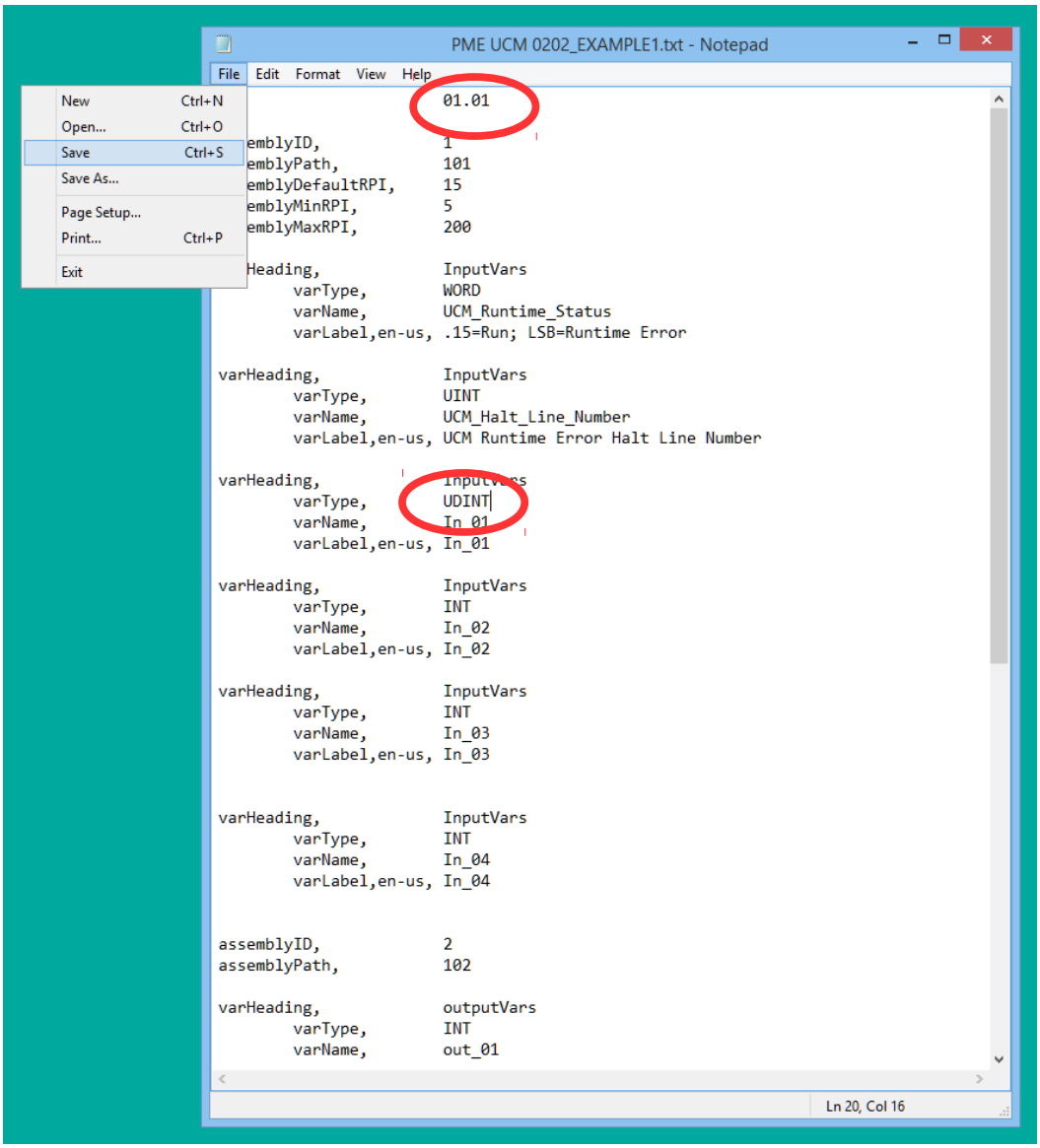
The PLC OUTPUT data structure follows.



Again there is the byte count of outputs 14, followed by the total byte count including the PTK overhead of 26 bytes (40). The assemblyID = 2 and Path = 102 are from the .txt file. The Ethenet/IP index 1006 is next followed by the byte offset from the start of the data structure.

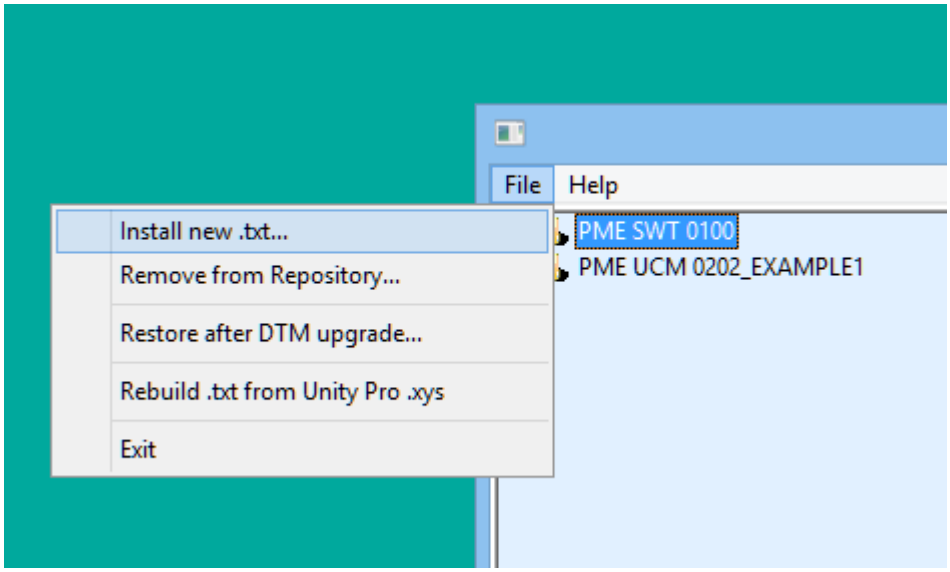
Making Changes to a File

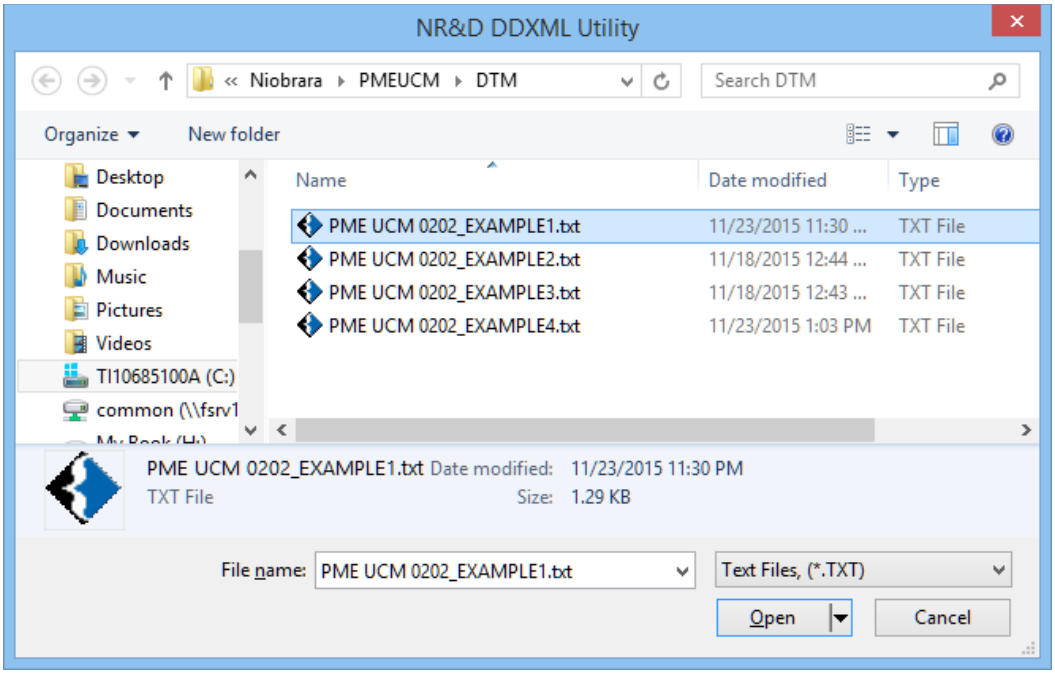
Making a change to the installed DTM is a simple as modifying the txt file, saving it, and then performing the Import function again.



In this example, the SW version is changed from 1.00 to 1.01. Also, the varType of In_01 is changed from an UINT to a UDINT. The new version is saved with the same filename.

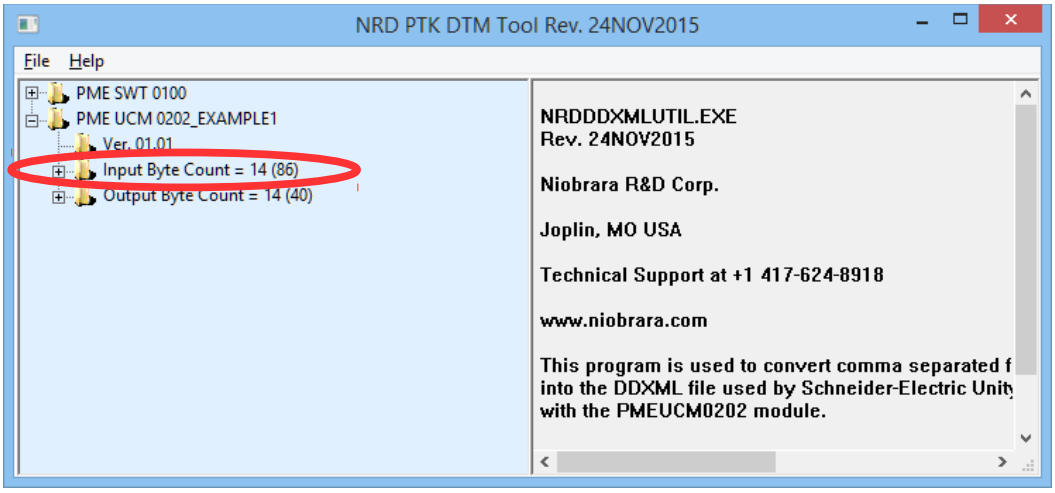
Now, back in the NRD DTM Utility, do a File > “Install new .txt...” and select the same filename.



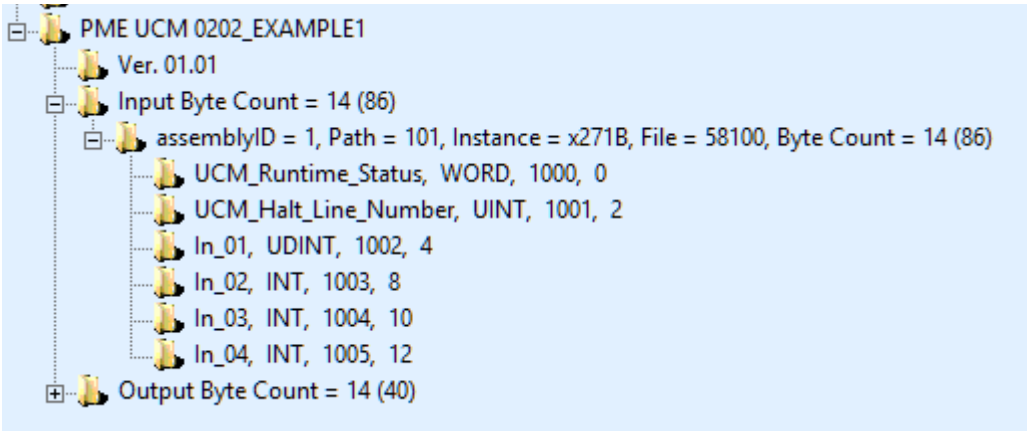


Note: It is not necessary to remove the installed DTM to make modifications. Simply Install the same file again.

Note: The INTPUT Byte Count has changed from 12 to 14 because the INT was changed into a DINT.



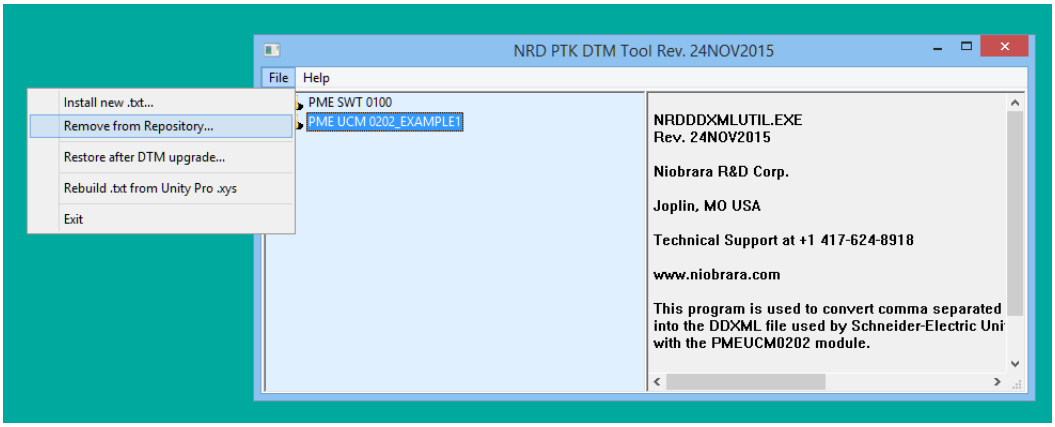
Expanding the tree shows the new structure for the PLC INPUT data:



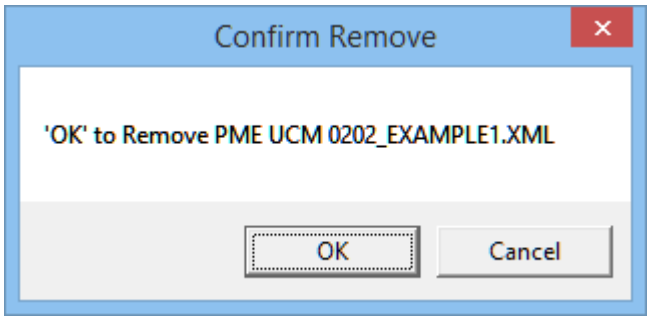
Notice that the bytes offset of In_02 is now 8 because In_01 is a UDINT (4 bytes).

Removing an Entry

Highlight one of the DTM names in the tree and select File > “Remove from Repository”.

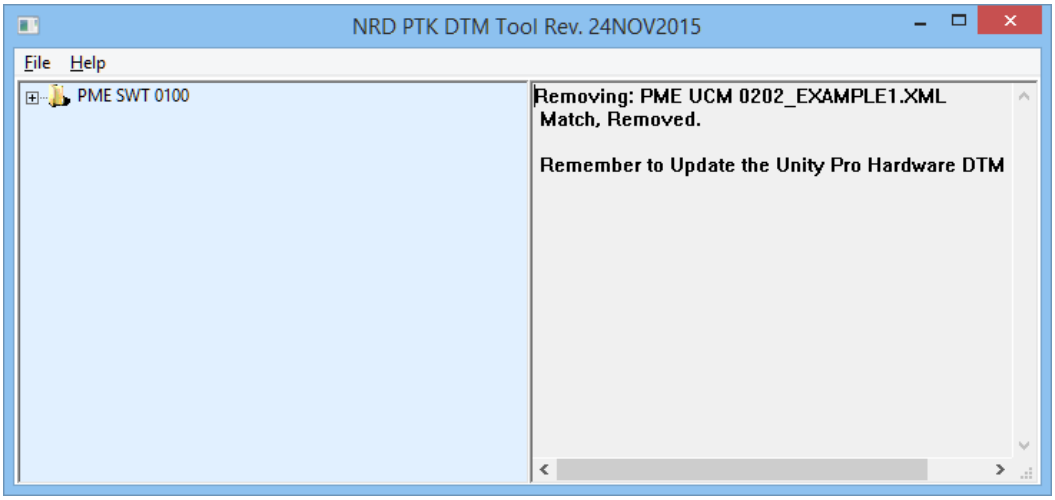


A prompt will appear asking for conformation of the removal.



Select “OK” to remove the DTM entry.

Select “Cancel” to keep the DTM entry.



Restore after DTM Upgrade…

The File > “Restore after DTM Upgrade” menu item is used to recover installed DDXML files from the repository after an update to the DTM has been performed.

Rebuild .txt from Unity Pro .xys…

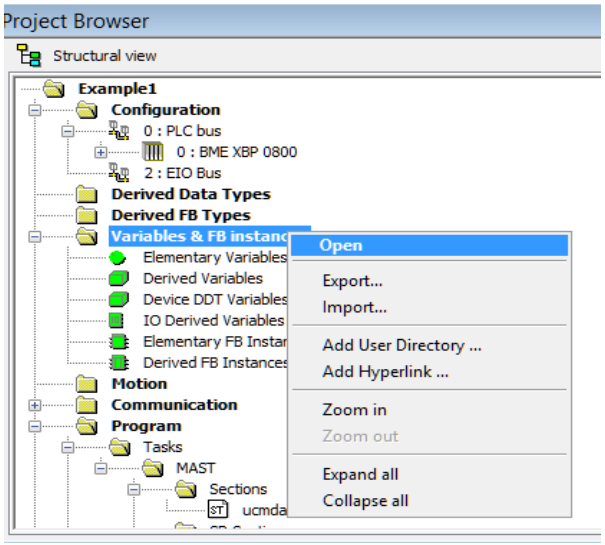
The File > “Rebuild .txt from Unity Pro .xys...” menu item is used to recover a .txt source file from an active Unity Pro project. This may be necessary if the original .txt file is lost and the only information available about the application variable structure is the Unity Pro project.

NOTE: The file rebuilding function makes the following assumptions:

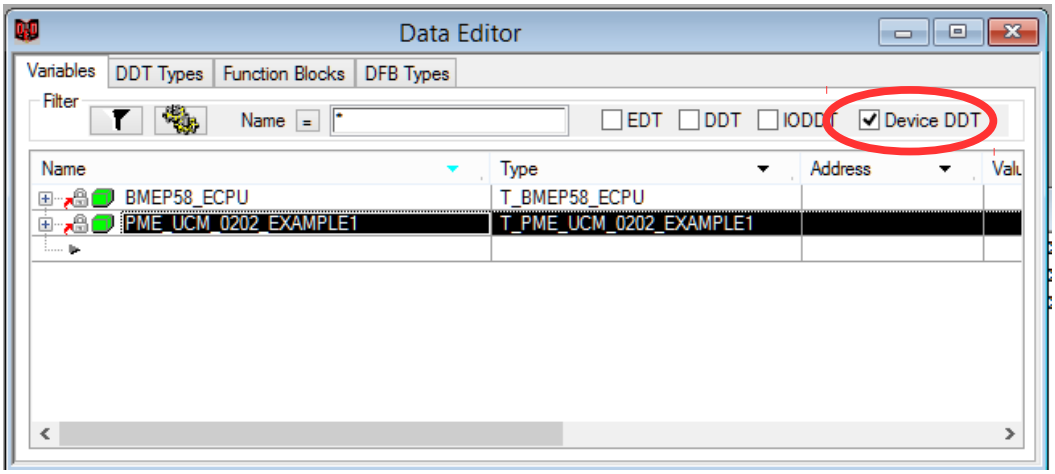
- All PLC_INPUTS are placed in assemblyID = 1 with PATH = 101
- All PLC_OUTPUTS are placed in assemblyID = 2 with PATH = 102
- RPI is set to 10mS

NOTE: It is necessary to inspect the PMEUCM source file to determine if more assemblyID definitions or PATH adjustements are required.

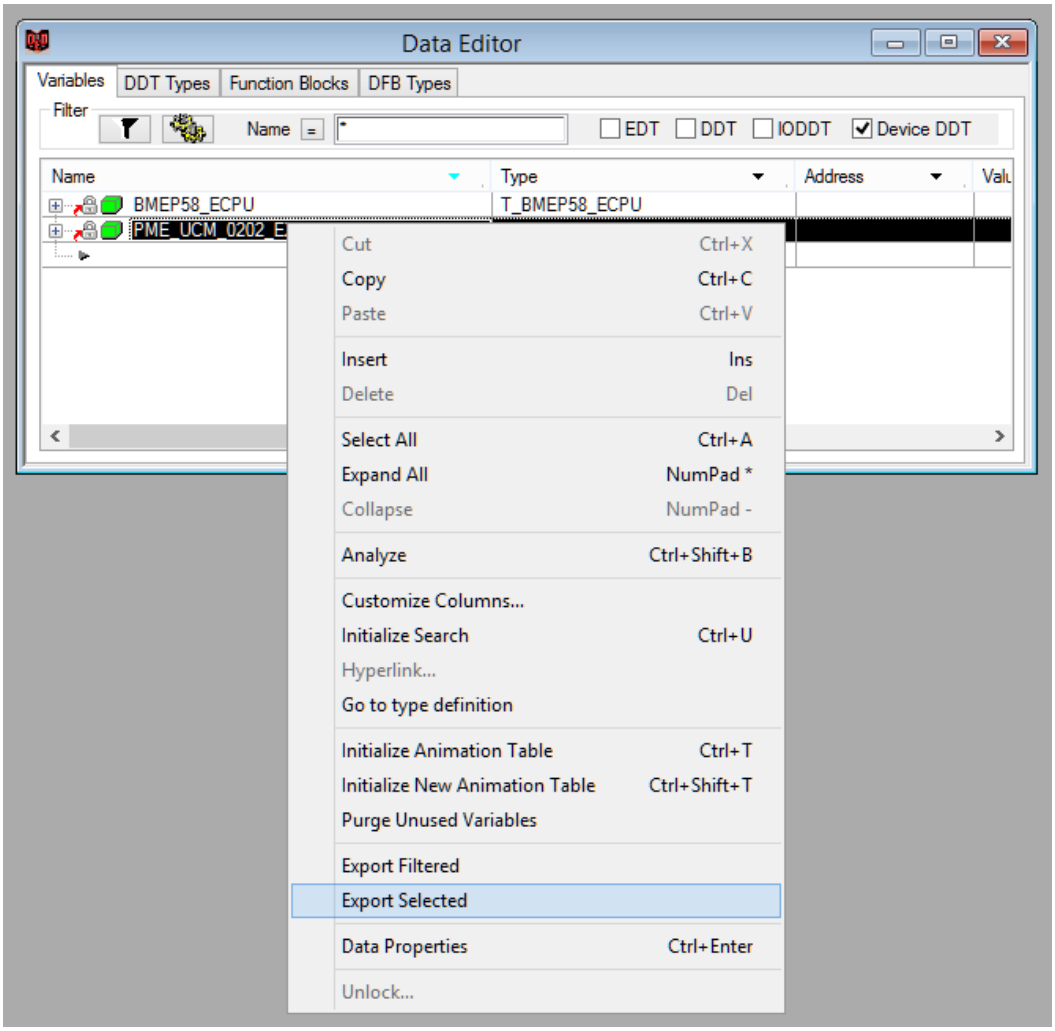
The first step is to open the Project Browser in the Unity Pro project and right click on the Variables & FB Instances to select “OPEN”.



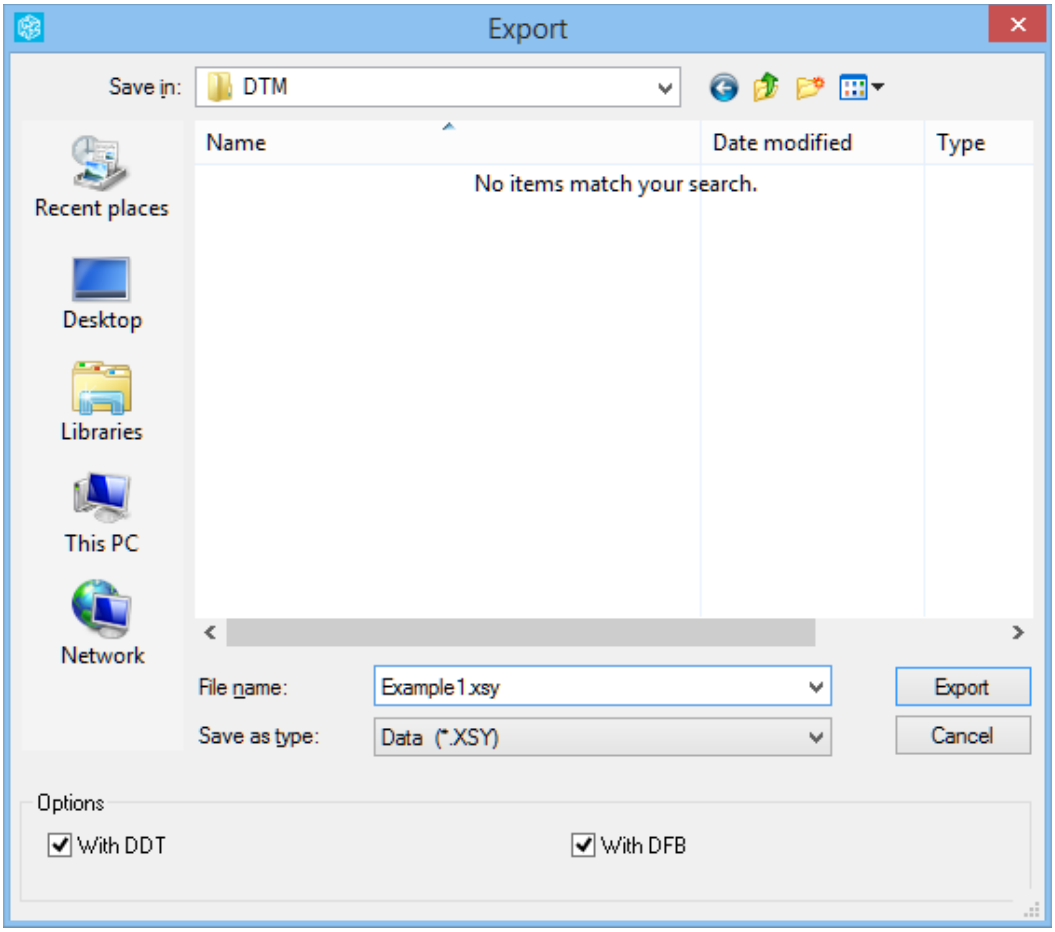
The Data Editor will open. Narrow down the displayed items to only include “Device DDT”.



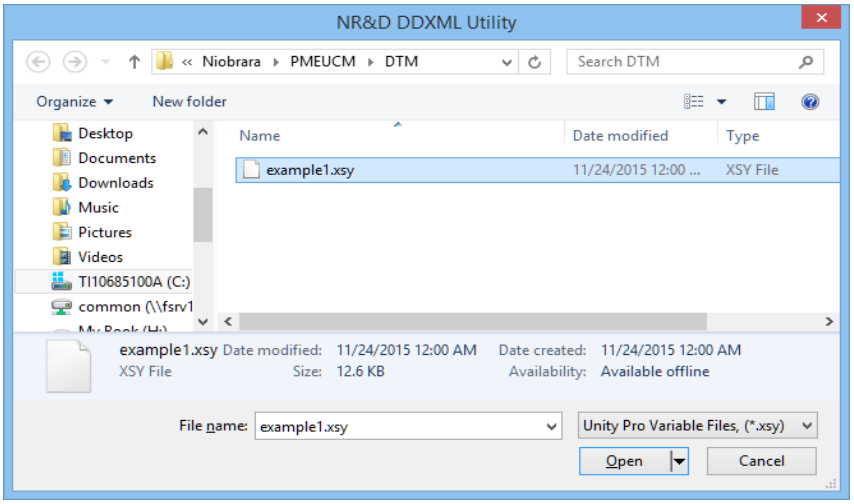
Now right click on the PME_UCM_0202_EXAMPLE1 structure and select "Export Selected".



Now Export the file with the .xsy extension into the c:\Niobrara\PMEUCM\DTM\ folder.



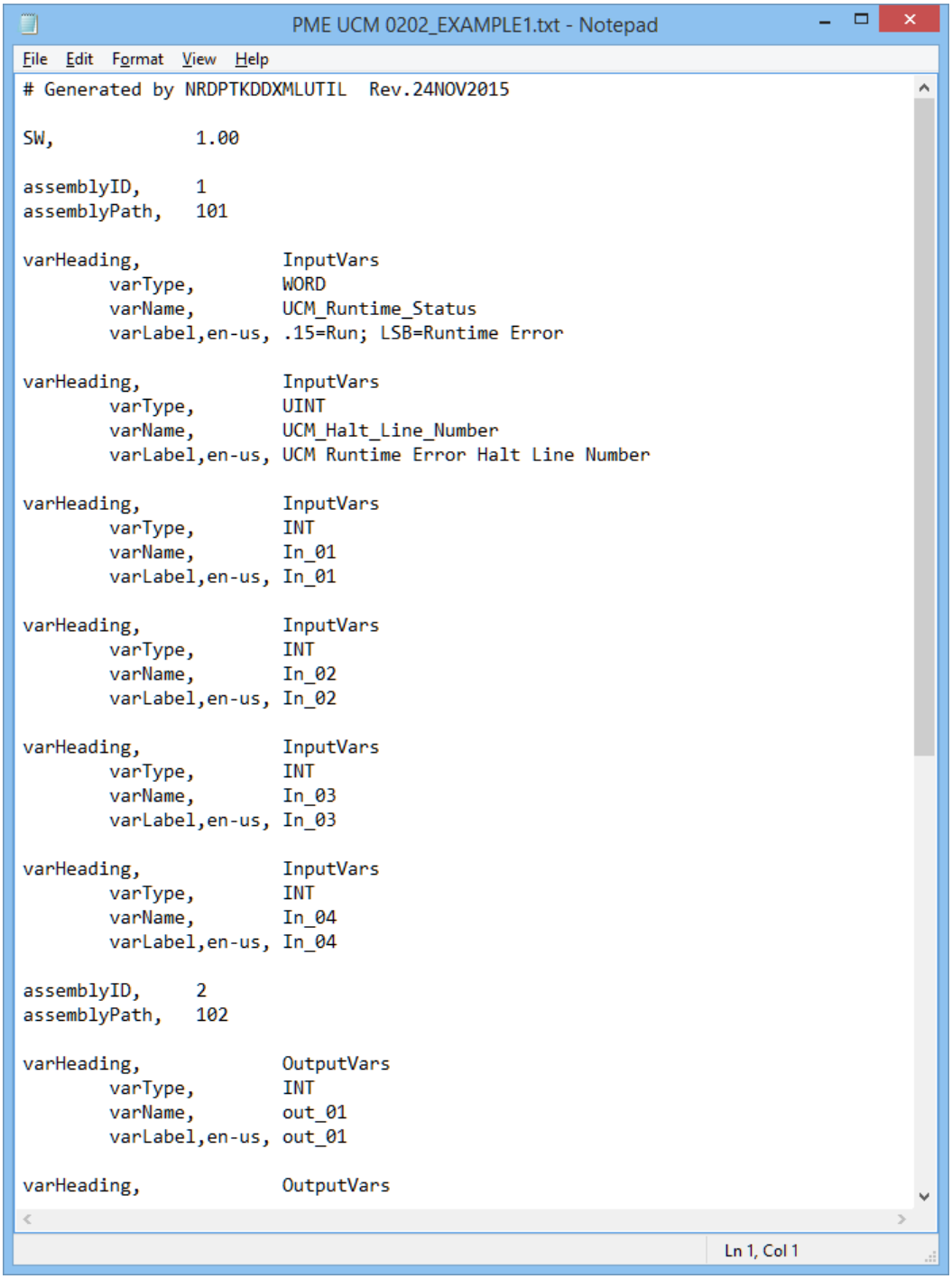
Now select File > “Rebuild .txt from Unity Pro .xsy” and select the new file.



NOTE: If prompted “OK to overwrite PME UCM 0202_EXAMPLE1.txt”, select Cancel because the .txt file is already present and need not be rebuilt.

NOTE: The generator will make the new txt file with the name “PME UCM 0202_name.txt” format.

Now the .txt file may be opened in Notepad and altered if necessary.



When satisfied that the new .txt file will work, it may be now be installed like normal using FILE > “Install new.txt”.

7 QLOAD the Example1 UCM Application

The standard PMEUCM is shipped from the factory with the Example1 application preloaded. The following chapters use the Example1 application and this chapter explains how to install this application.

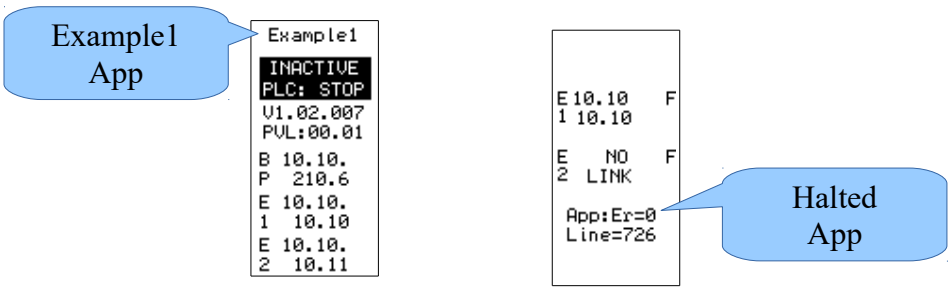
NOTE: It is good practice to follow these procedures even if the Example1 application is already installed in the PMEUCM.

Determine the Installed Application

The first step is to determine what (if any) application is currently installed and possibly running in the test PMEUCM.

For the purposes of this manual, it is best to stop and erase any installed application, reset the module to factory defaults, and qload the Example1.qcc file.

Install the module in an Ethernet slot of a powered M580 rack. The module will beep, blink some lights at the top, and eventually show something on the front LCD screen.



If the screen shows “Example1” at the top then the proper application is already loaded, but it is a good idea to qload a new copy anyway.

If the screen shows the OS default splash screen with App:Er=0, then the currently loaded application is halted.

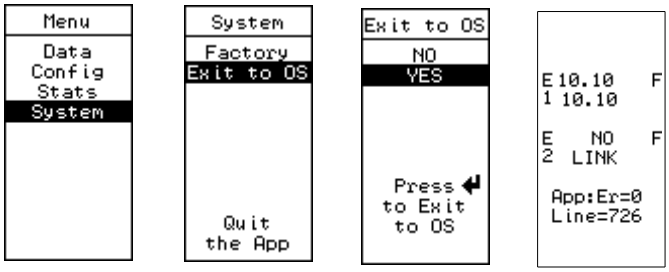
If the screens shows something else, some other application may be loaded and should be

halted and erased at this time.

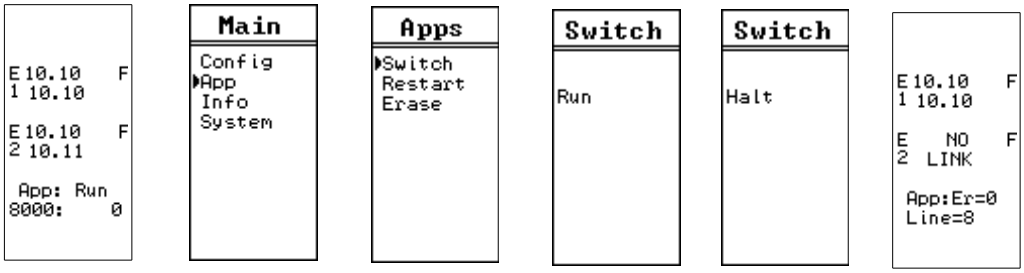
Halting a running Application

Most PMEUCM applications written by Niobara will include some type of screen driver with standardized menus. Use the joystick to navigate the menu to exit to the OS.

NOTE: Push “in” on the joystick for the “Enter” selection.



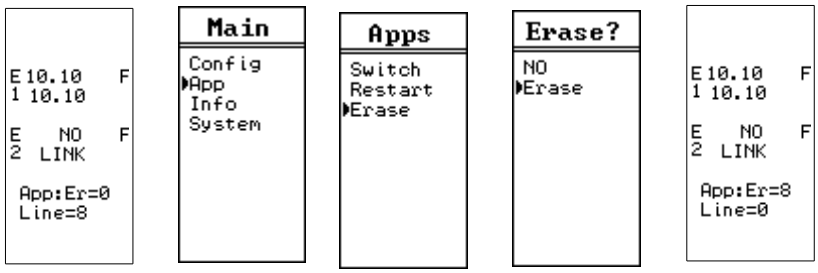
If the application does not include this type of screen driver but shows the OS screens, navigate to the Main > App > Switch > Halt screen.



If all else fails, push the joystick “IN” for “Enter” and “UP” at the same time and hold for 10 seconds. This will cause the application to halt and exit to the OS.

Erasing the Installed Application

Once the screen is under the UCM OS control, it is simple to erase the installed application.

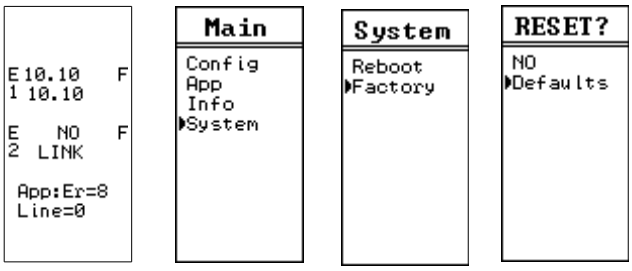


Notice that the App:Er=8 after the erase command is processed. Error 8 means that the checksum for the application is bad and the app will not run.

Factory Default

Now reset the module to factory defaults.

NOTE: Factory defaults does not erase a user application. It simply sets the Ethernet and serial ports to their default settings.



The module will reboot and come up with the following settings:

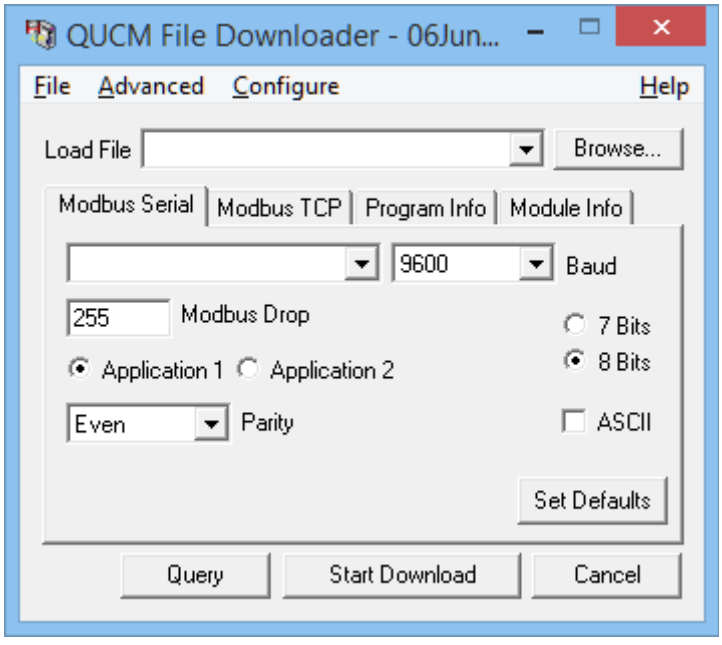
Setting	Factory Default Value
E1 IP Address	10.10.10.10
E2 IP Address	10.10.10.11
E1 and E2 Subnet Mask	255.0.0.0
E1 and E2 Default Gateway	0.0.0.0
OS Modbus/TCP Port	502
S1 and S2 Mode	Modbus RTU Slave
S1 and S2 Baud Rate	9600
S1 and S2 Parity	EVEN
S1 and S2 Data Bits	8
S1 and S2 Stop Bits	1
S1 and S2 Driver	RS-232 (Fixed)

NOTE: Rebooting the UCM does not reboot the PTK board inside the PMEUCM0202. Cycling power on the rack or the PMEUCM will reboot both the UCM board and PTK board.

QLOAD Example1.qcc

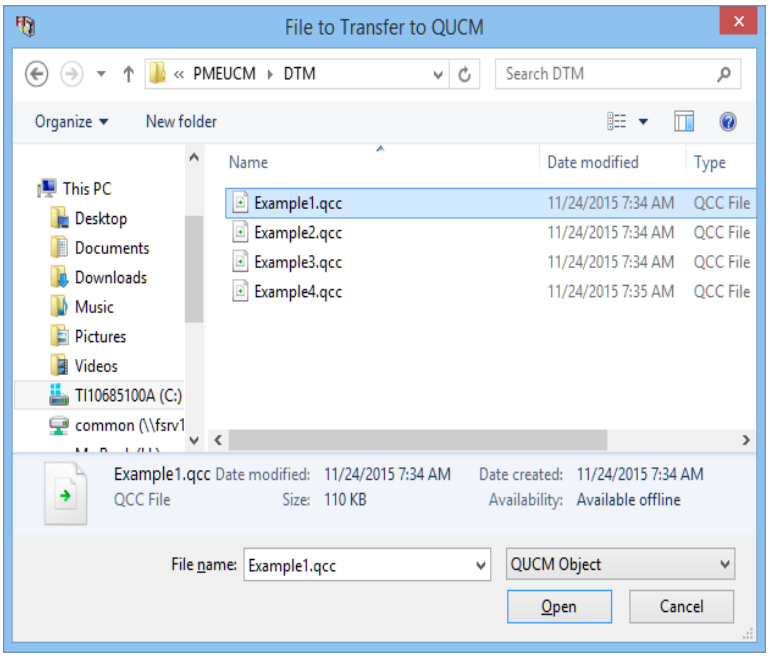
The QLOAD utility is used to load applications into the PMEUCM. Start QLOAD by Start > Programs > Niobrara > QLOAD.

The first time QLOAD is started, it should look something like this:

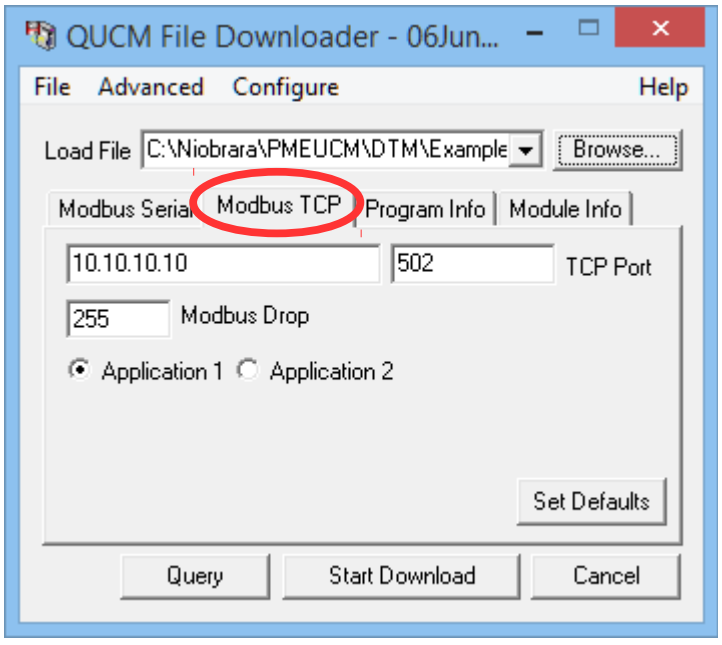


Click on the Browse button and select this file:

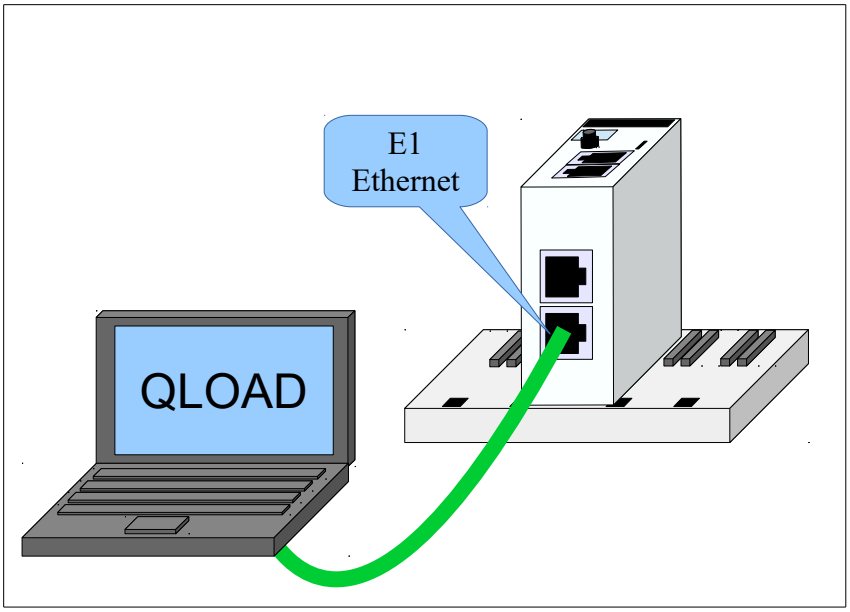
c:\Niobrara\PMEUCM\DTM\Example1.qcc



Now select the ModbusTCP tab.



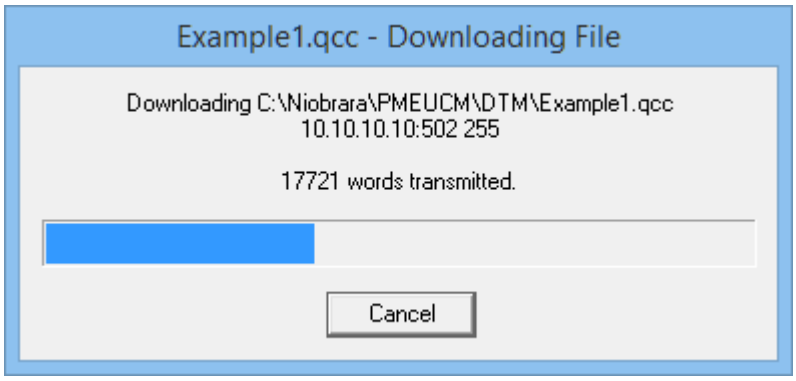
Make sure that the IP Address is set to match the PMEUCM E1 port of 10.10.10.10, the TCP Port is set to 502, Modbus Drop is 255, and Application 1 radio button is set.



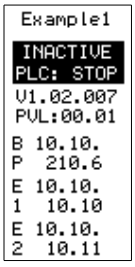
Connect the Ethernet port of the computer to E1 on the PMEUCM with a standard CAT5/6 cable.

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:



8 Unity Pro Operations

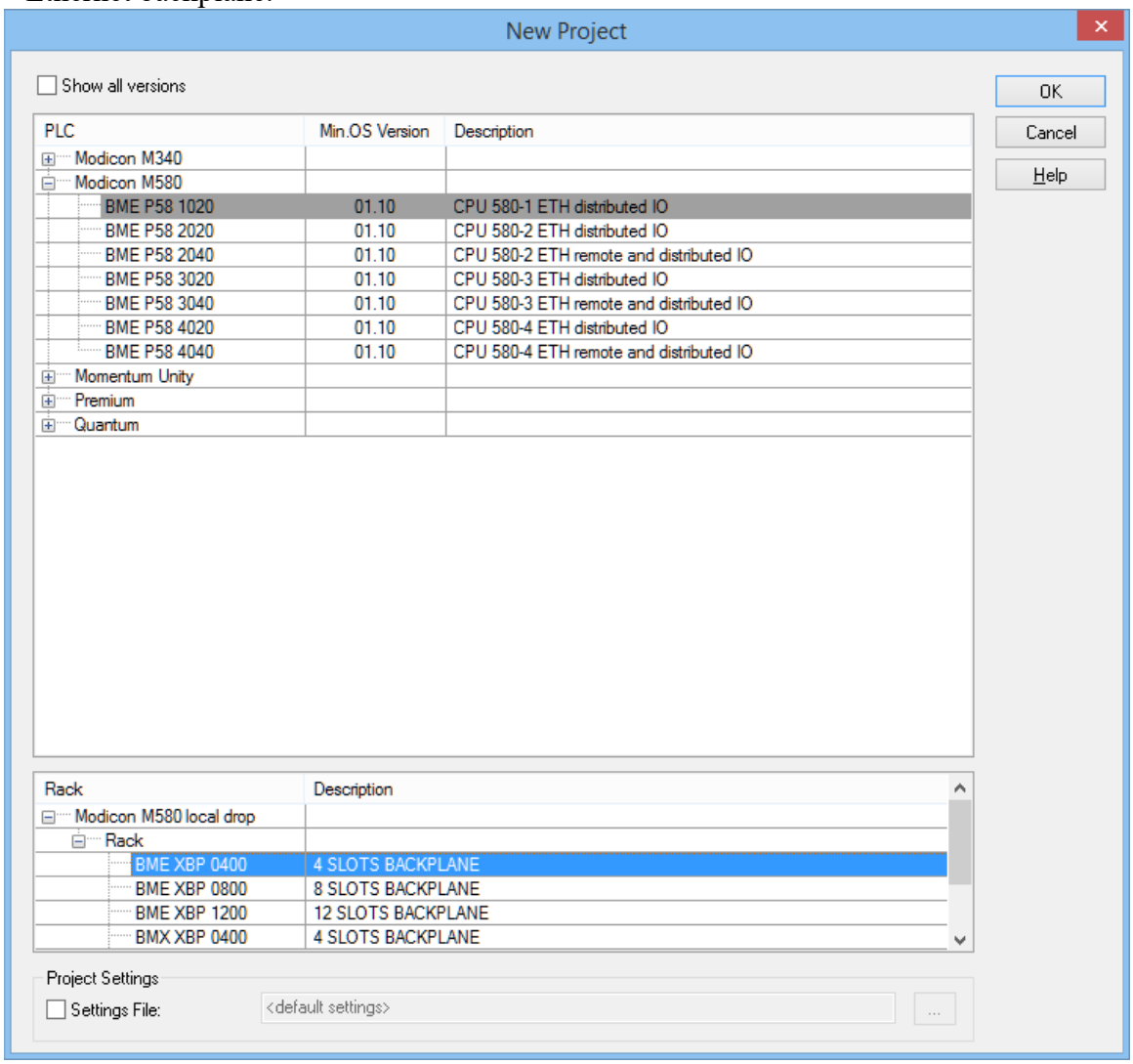
NOTICE: The newest version of the PTK_DTM_Libraray must be installed before attempting to use the PMEUCM. See Chapter 3.

New Project

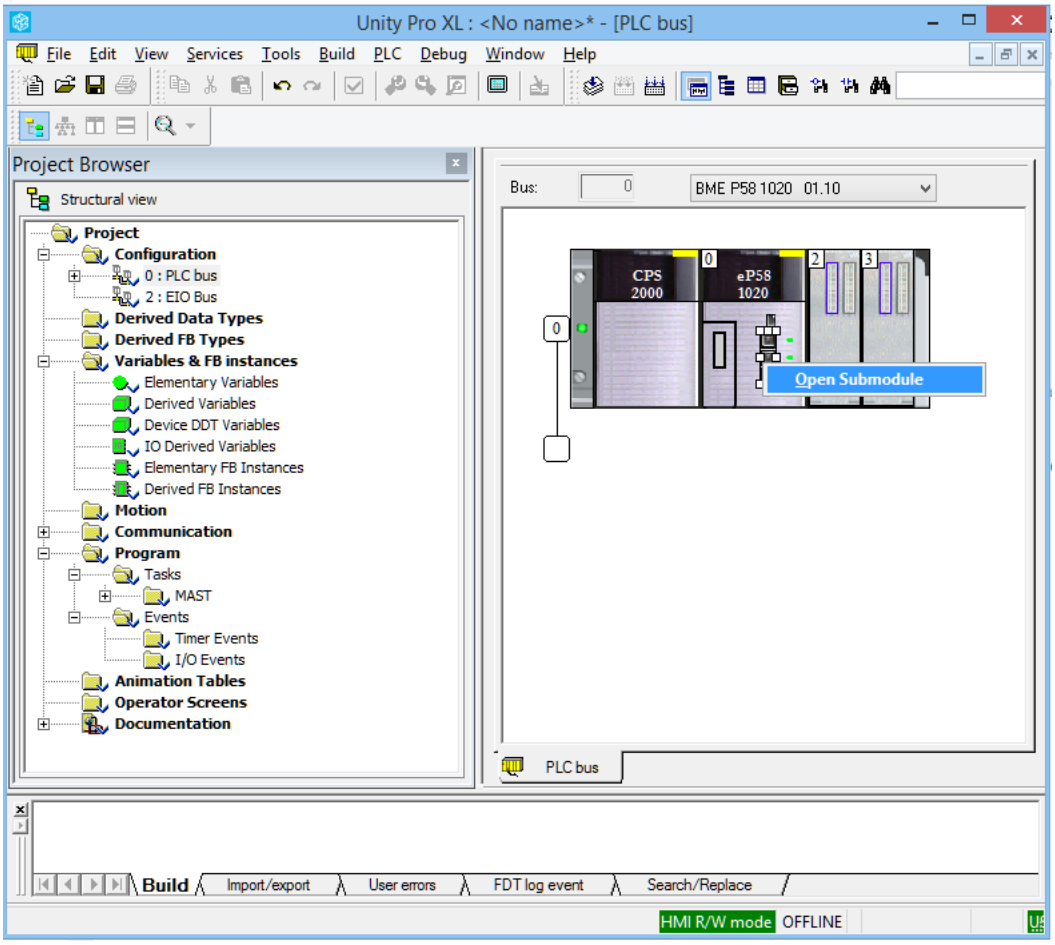
This example starts with a new project in Unity Pro XL V8.1.

- The PME UCM 0202 will be installed in the CPU rack slot 2.
- The M580 P581020 is the chosen CPU.
- All of the IP Addresses will be left at their default settings.
  - The CPU will be at default IP Addresses of 192.168.10.1 and 192.168.11.1
  - The PME UCM backplane (PTK) will be at 192.168.10.3
  - The PME UCM E1 and E2 ports will be at 10.10.10.10 and 10.10.10.11

The BME P58 1020 CPU is chosen, along with a BME XBP 0400 four slot Ethernet backplane.

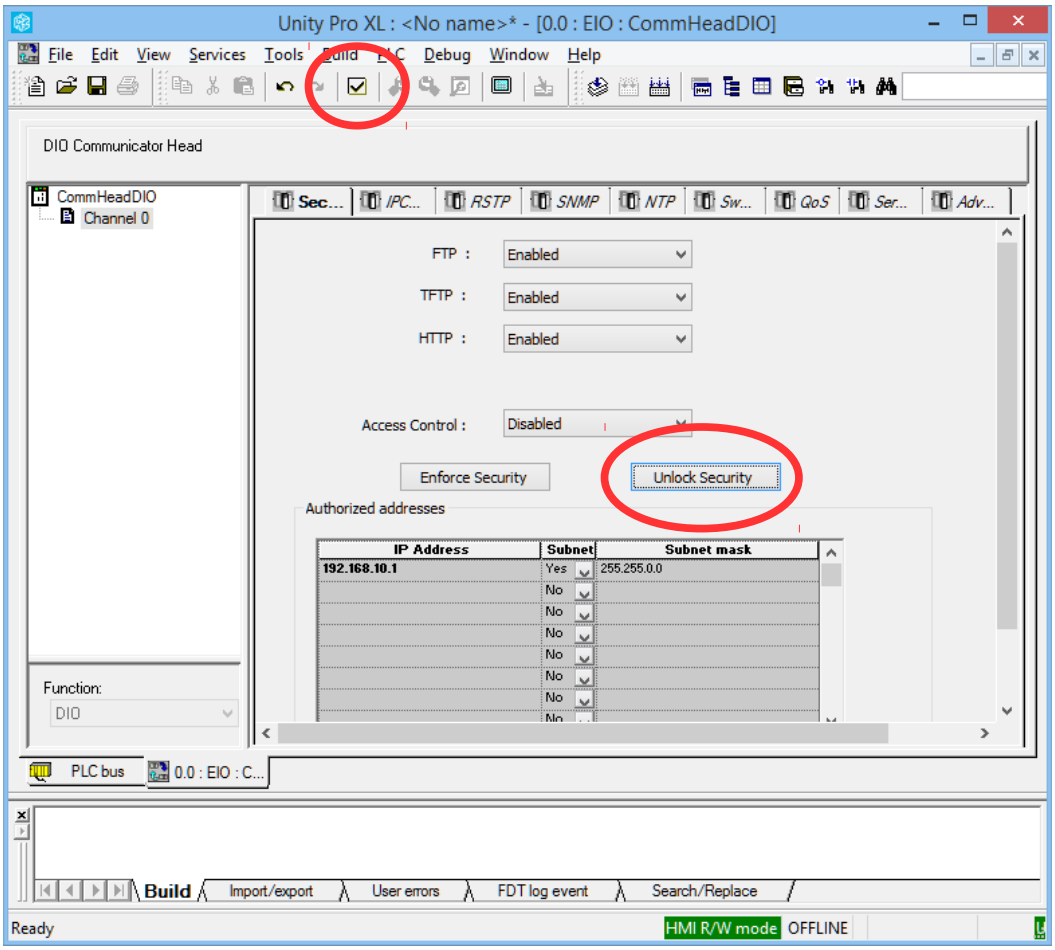


After selecting the “PLC Bus” in the Structural View Tree, right click on the Ethernet ports of the CPU to open the configuration submodule.



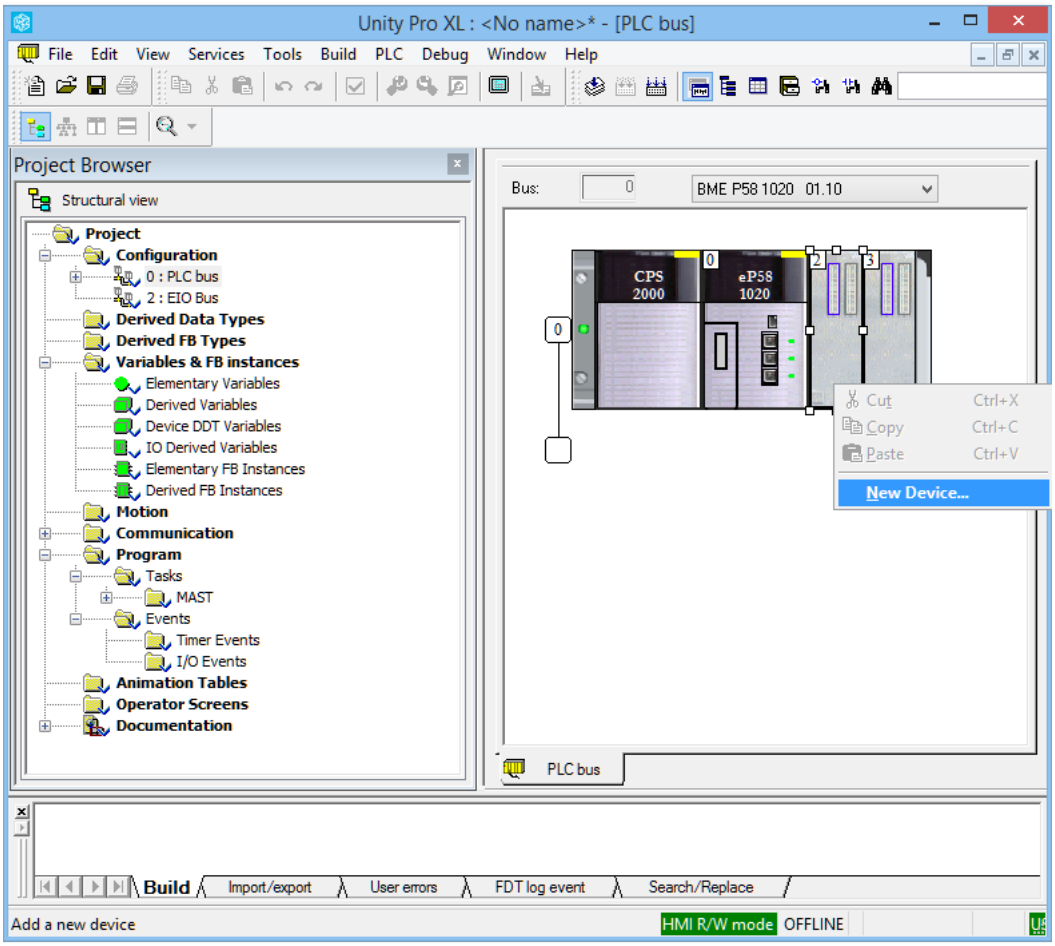
The FTP server must be enabled in the PLC for any PTK Partner module to function. The easy way to enable this server is to select “Unlock Security”.

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

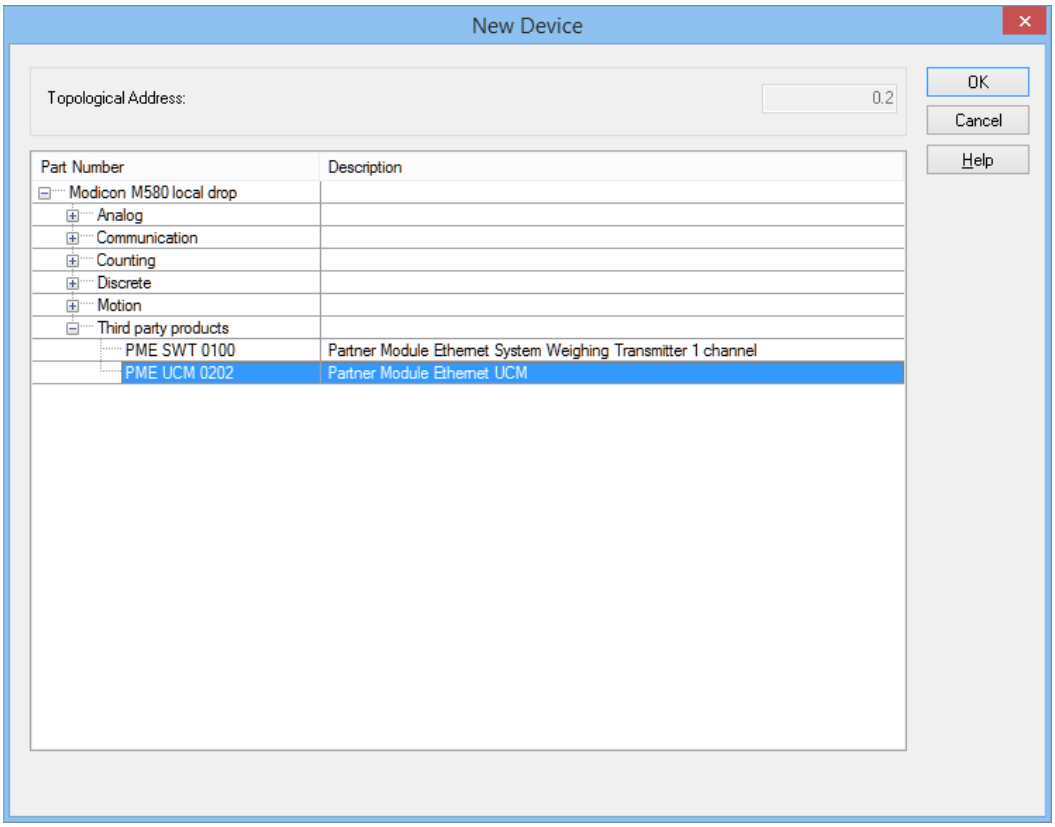


Now close the submodule.

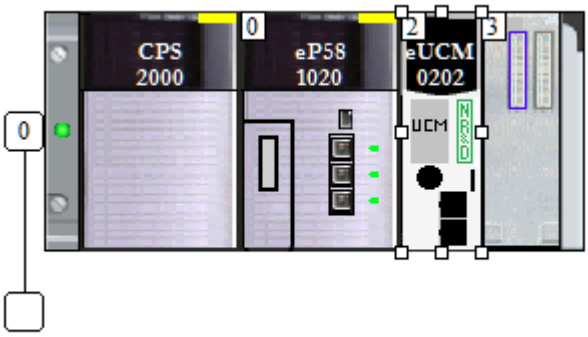
After right clicking on slot 2, a “New Device” is added to slot 2.



The PME UCM 0202 is selected from the “Third Party products” section.



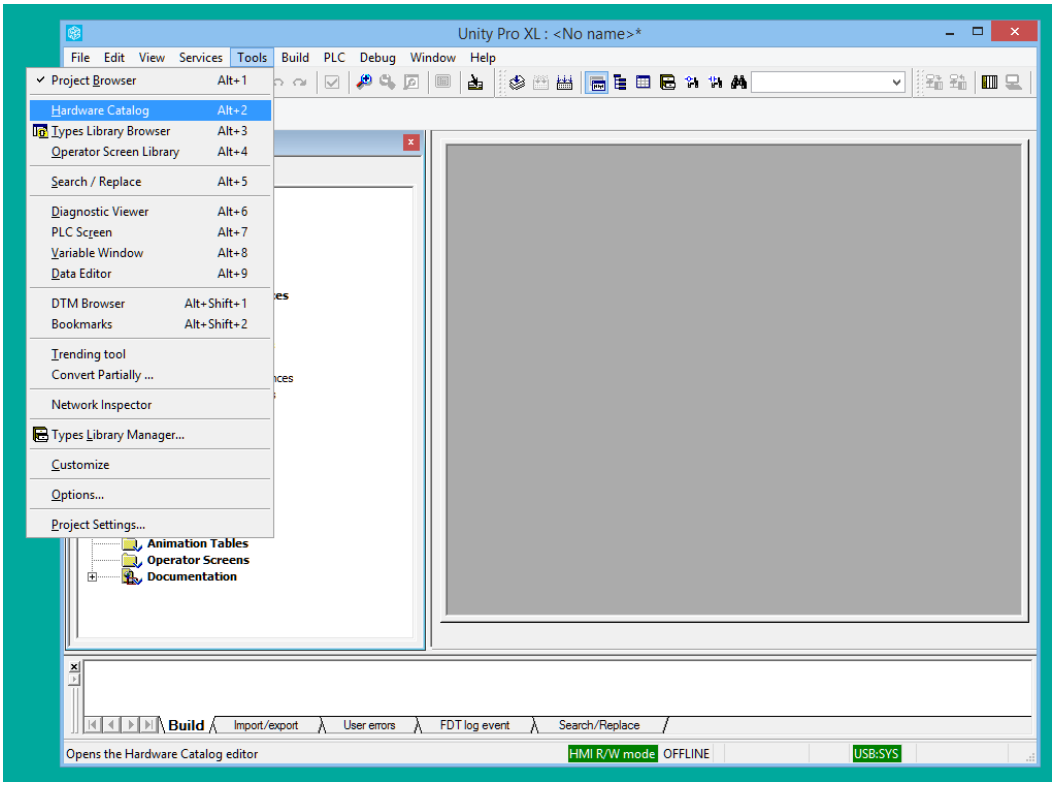
The UCM will now appear in the rack.



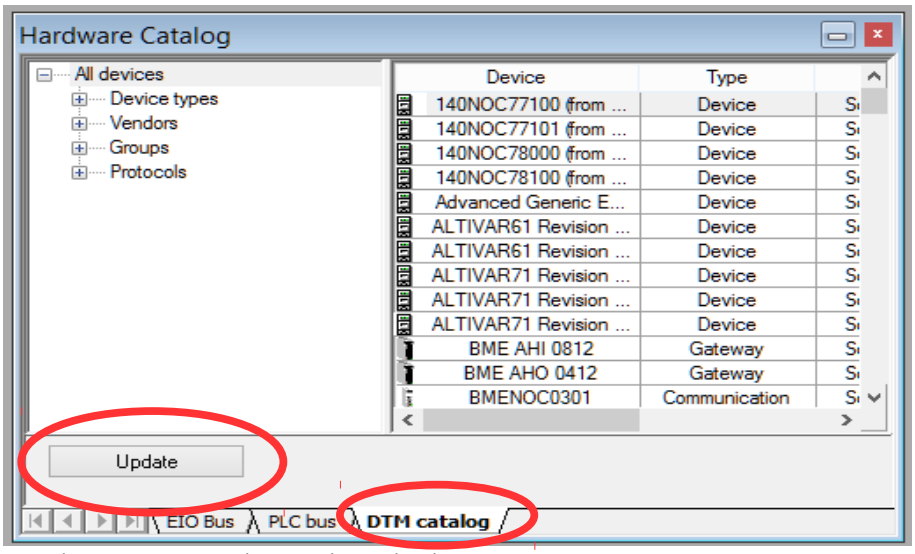
The PLC rack window may now be closed.

DTM Hardware Catalog Update

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:

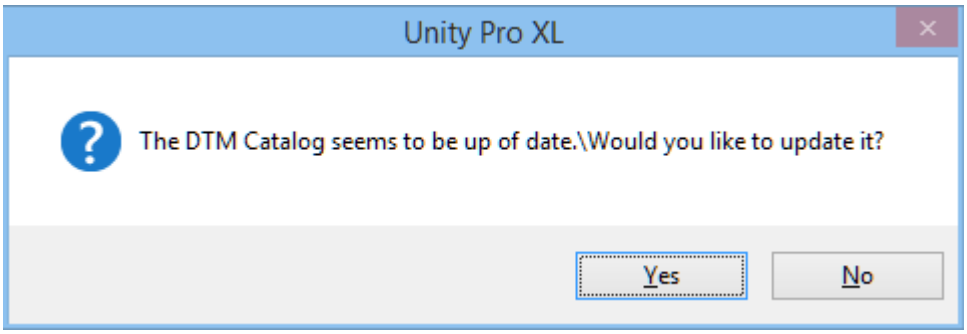


Click on the “DTM catalog” tab at the bottom.

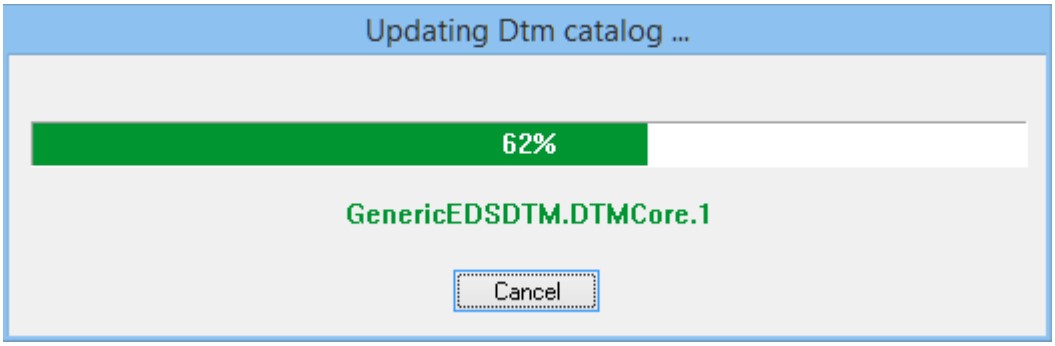
Then Click on the “Update” button.

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

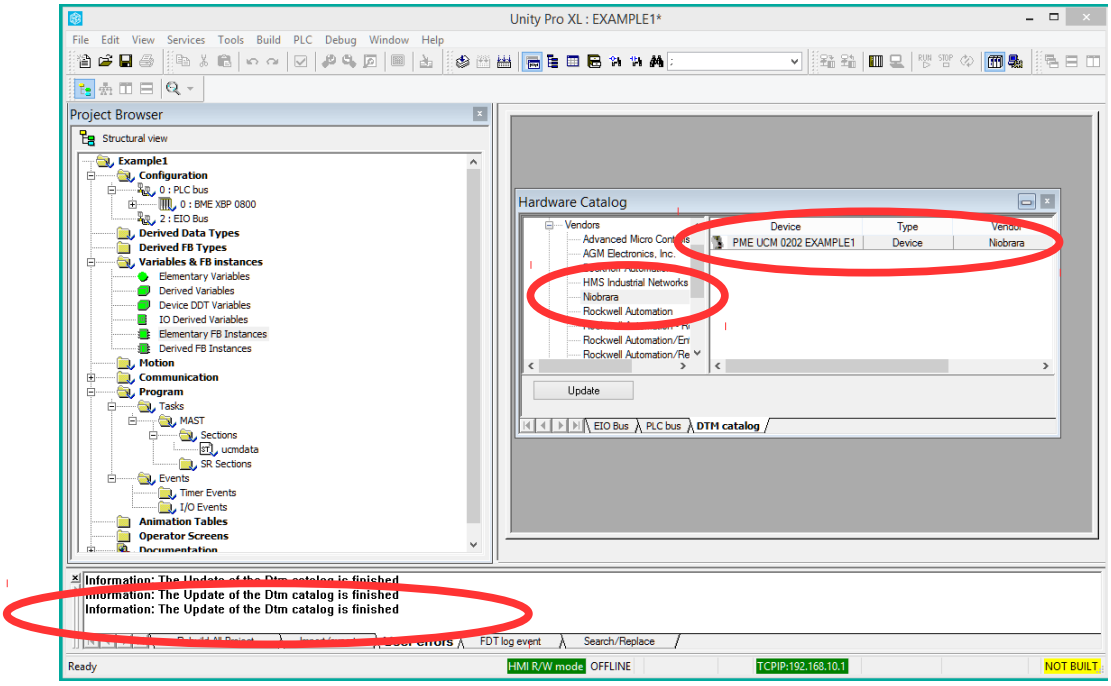
Note: This box opens every time the Update button is clicked – even if the catalog is updated twice in a row with no other changes.



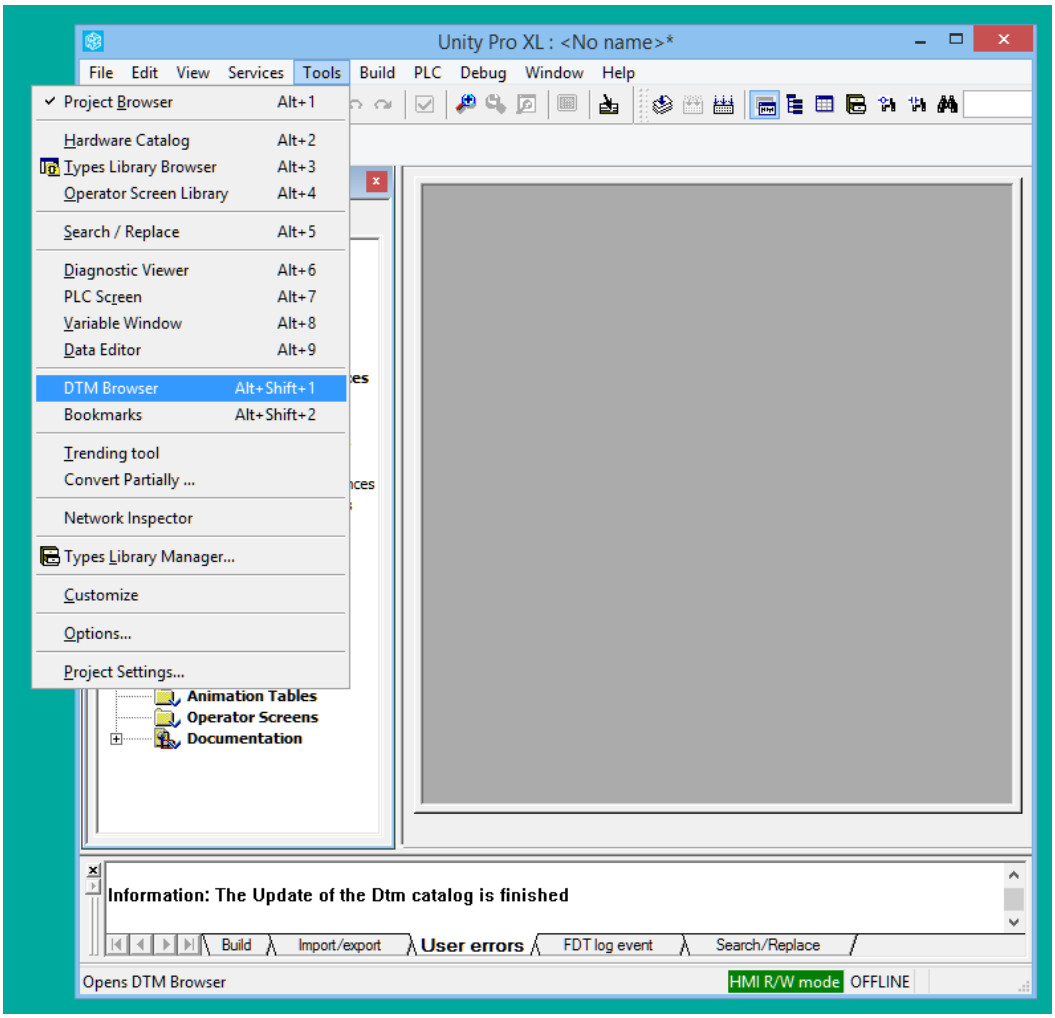
A progress window pops open.



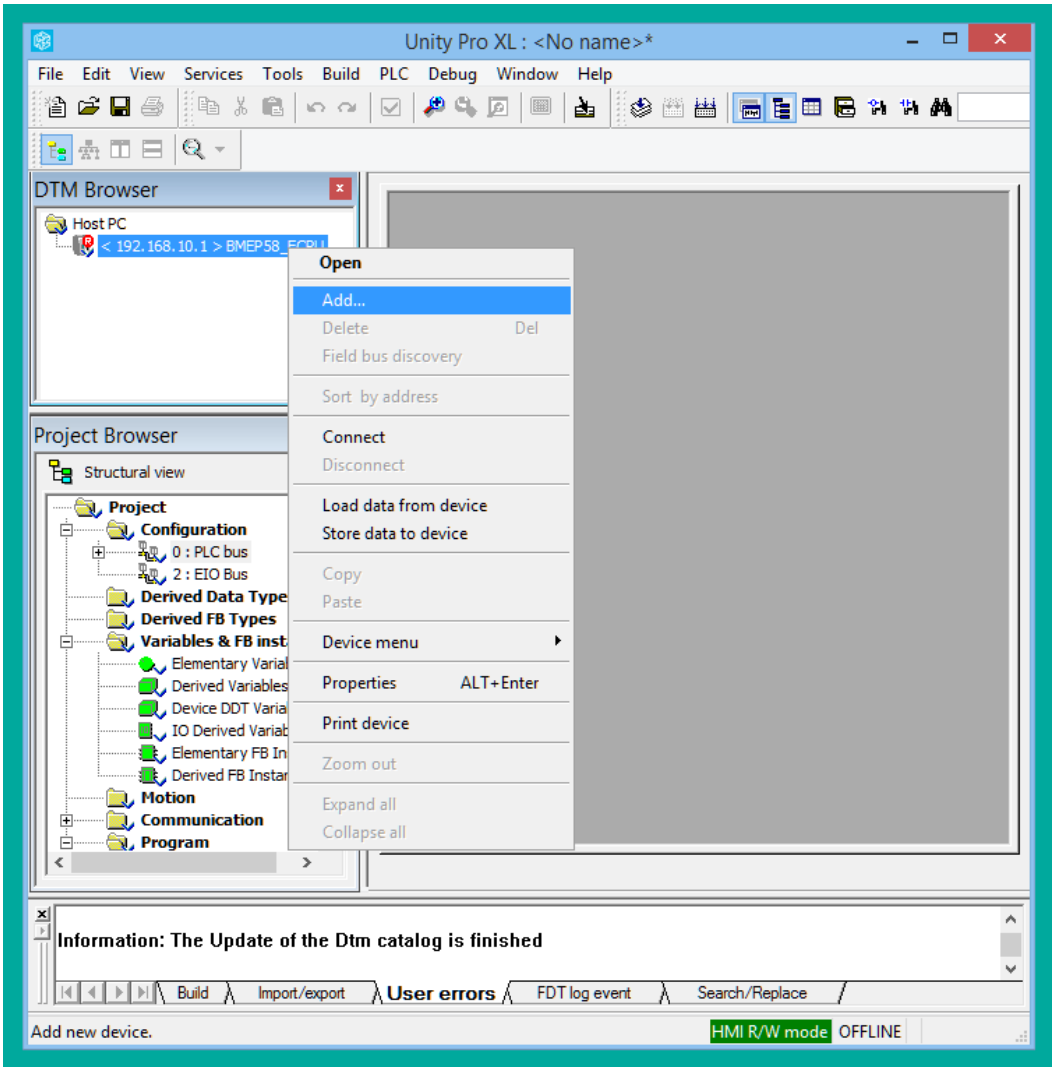
After the catalog update is complete, the new Niobrara DTM device should be listed in the hardware catalog. Also, the "User Errors" display should show "Information: The update of the Dtm catalog is finished"


Unity Pro XL : EXAMPLE1*
File Edit View Services Tools Build PLC Debug Window Help
Project Browser
Structural view
Example1
Configuration
0 : PLC bus
0 : BME XBP 0800
2 : EIO Bus
Derived Data Types
Derived FB Types
Variables & FB instances
Elementary Variables
Derived Variables
Device DDT Variables
IO Derived Variables
Elementary FB Instances
Derived FB Instances
Motion
Communication
Program
Tasks
MAST
Sections
ucmdata
SR Sections
Events
Timer Events
I/O Events
Animation Tables
Operator Screens
Hardware Catalog
Vendors
Advanced Micro Controls
AGM Electronics, Inc.
HMS Industrial Networks
Niobrara
Rockwell Automation
Rockwell Automation/En
Rockwell Automation/Re
Device
Type
Vendor
PME UCM 0202 EXAMPLE1
Device
Niobrara
Update
EIO Bus
PLC bus
DTM catalog
Information: The Update of the Dtm catalog is finished
Information: The Update of the Dtm catalog is finished
Information: The Update of the Dtm catalog is finished
FDT log event
Search/Replace
Ready
HMI R/W mode
OFFLINE
TCPIP:192.168.10.1
NOT BUILT

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



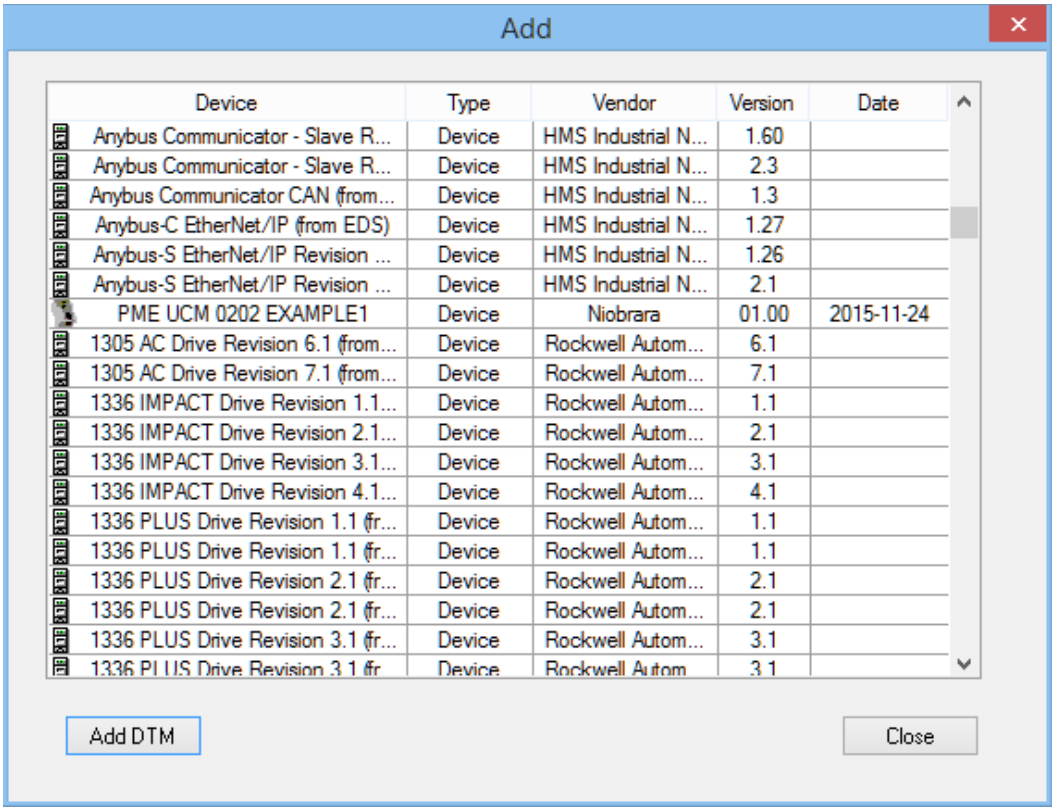
The DTM Browser will open and show a tree with the CPU at 192.168.10.1. Right click on the CPU and select “Add”.



A window will pop up showing all of the installed DTMs. Scroll down until you reach the PME UCM 0202 EXAMPLE1 device by Niobrara.

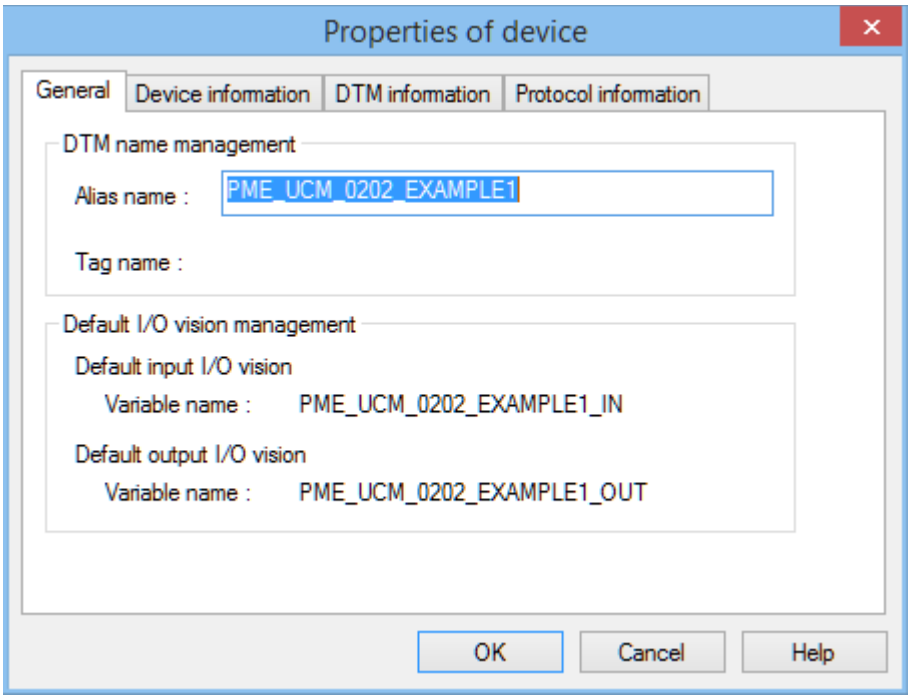
NOTE: This installation is using the original version of the EXAMPLE1 file with all variables being INTs.

Notice that it has the version 01.00 which matches the SW version in the txt file.

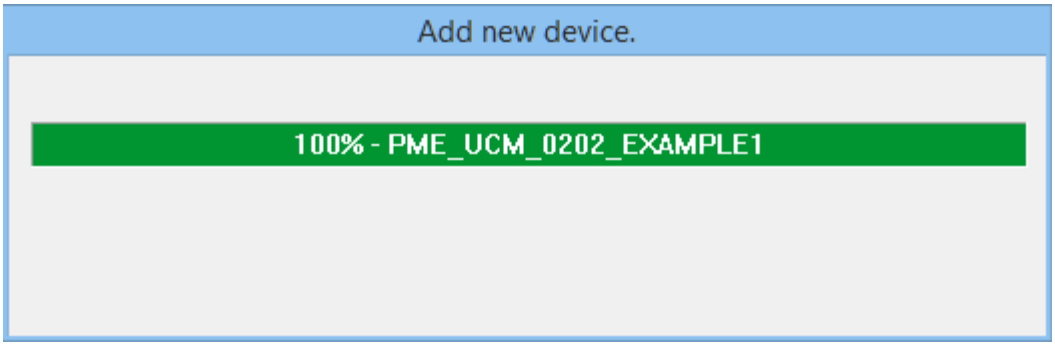


Press Enter or “Add DTM” to load the DTM for the PMEUCM. A window will pop up with information about the DTM.

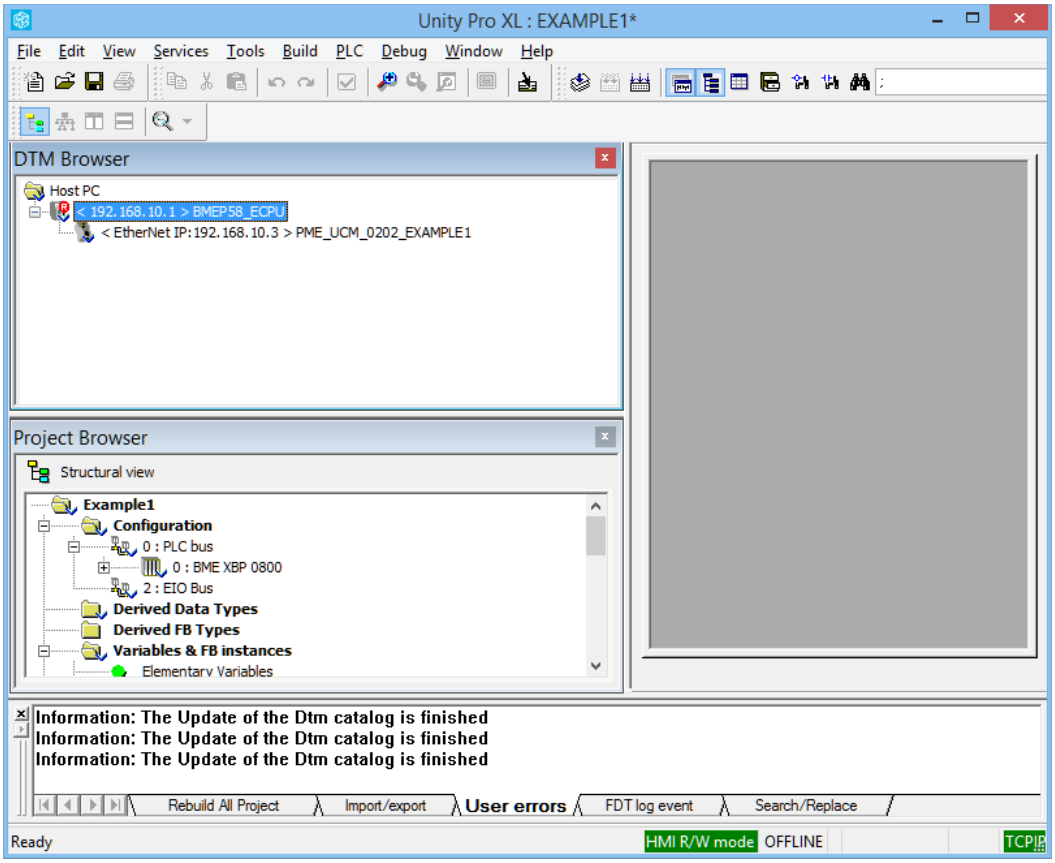
At this point the “Alias name” may be modified. The can be very handy for shortening the variable names created by the DTM.



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

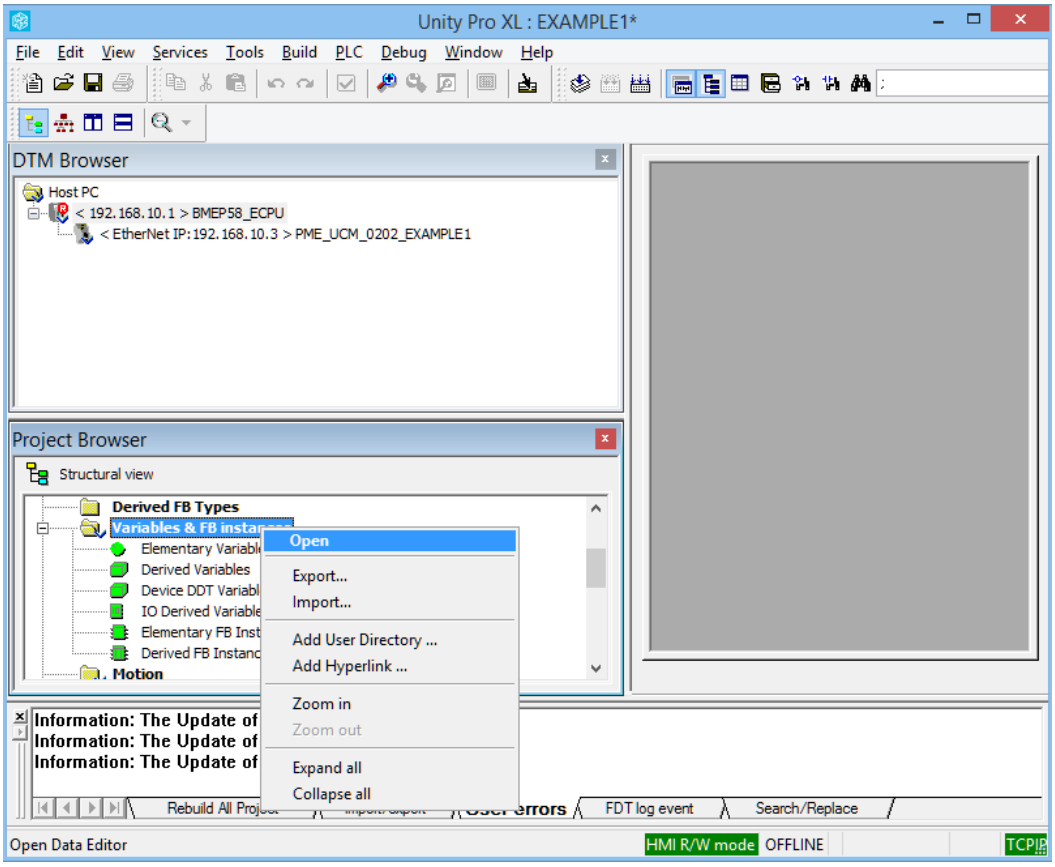


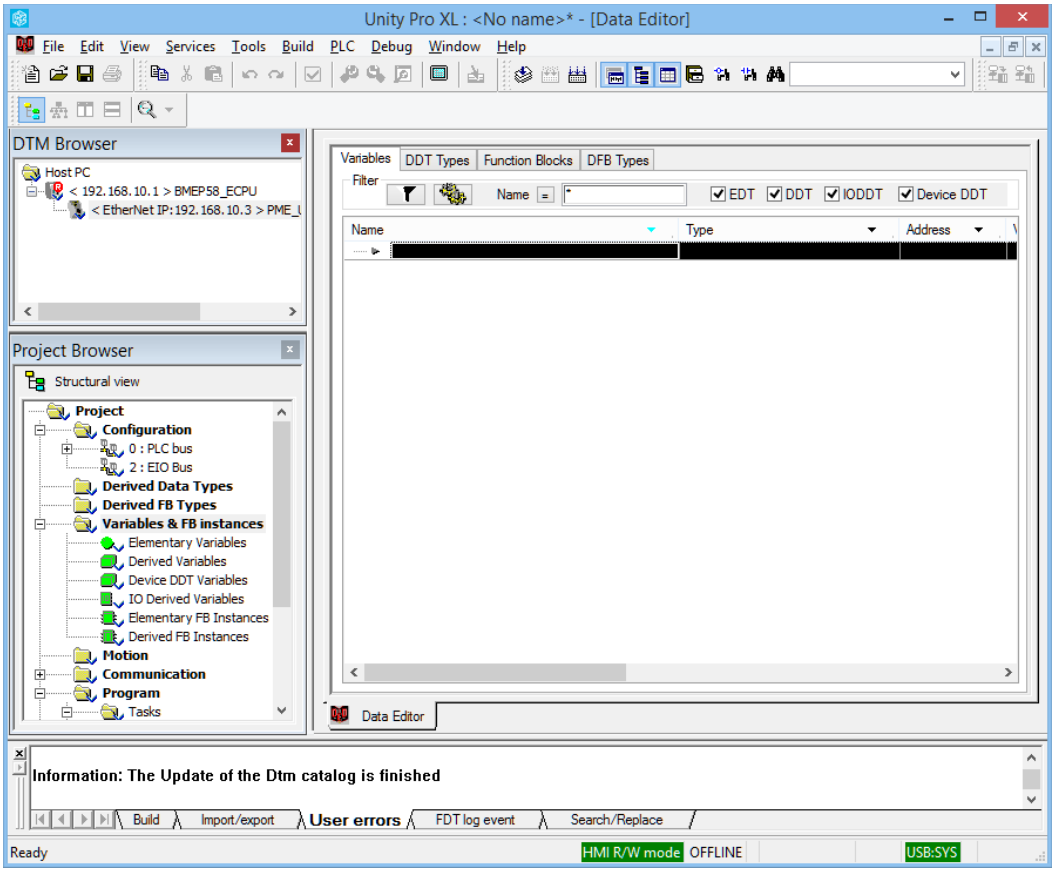
The PMEUCM is now added to the tree below the PLC.



Unity Variables

At this point the variables defined in the DTM are not yet present in the Unity Project. This is because the project has not been Analyzed or Built. Opening the "Variables & FB instances" item will verify that there are no variables in the system.





Now select Build > Analyze Project. A window will pop up showing the progress.

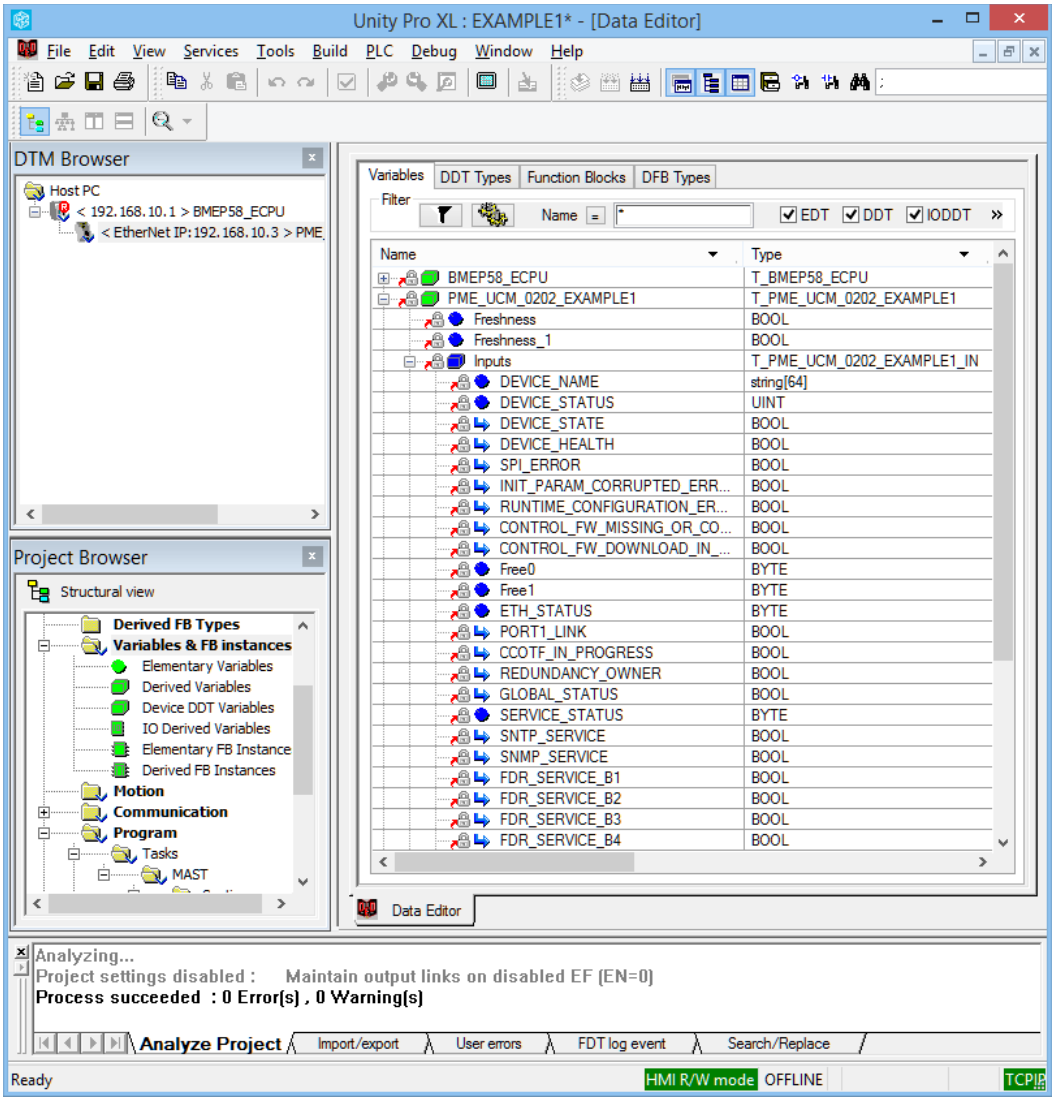
After the completion of the Analyze Project, two variable structures will appear. One for the CPU and one for the PMEUCM.

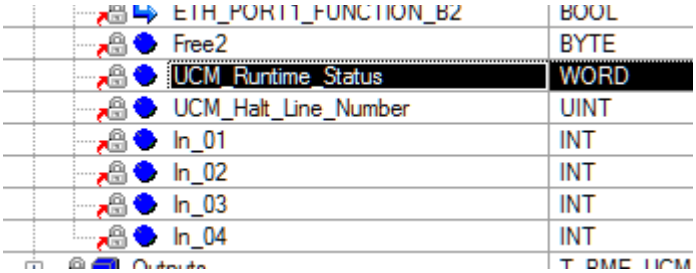
Notice that the PMEUCM structure includes Freshness BOOLs and both Input and Output structures.

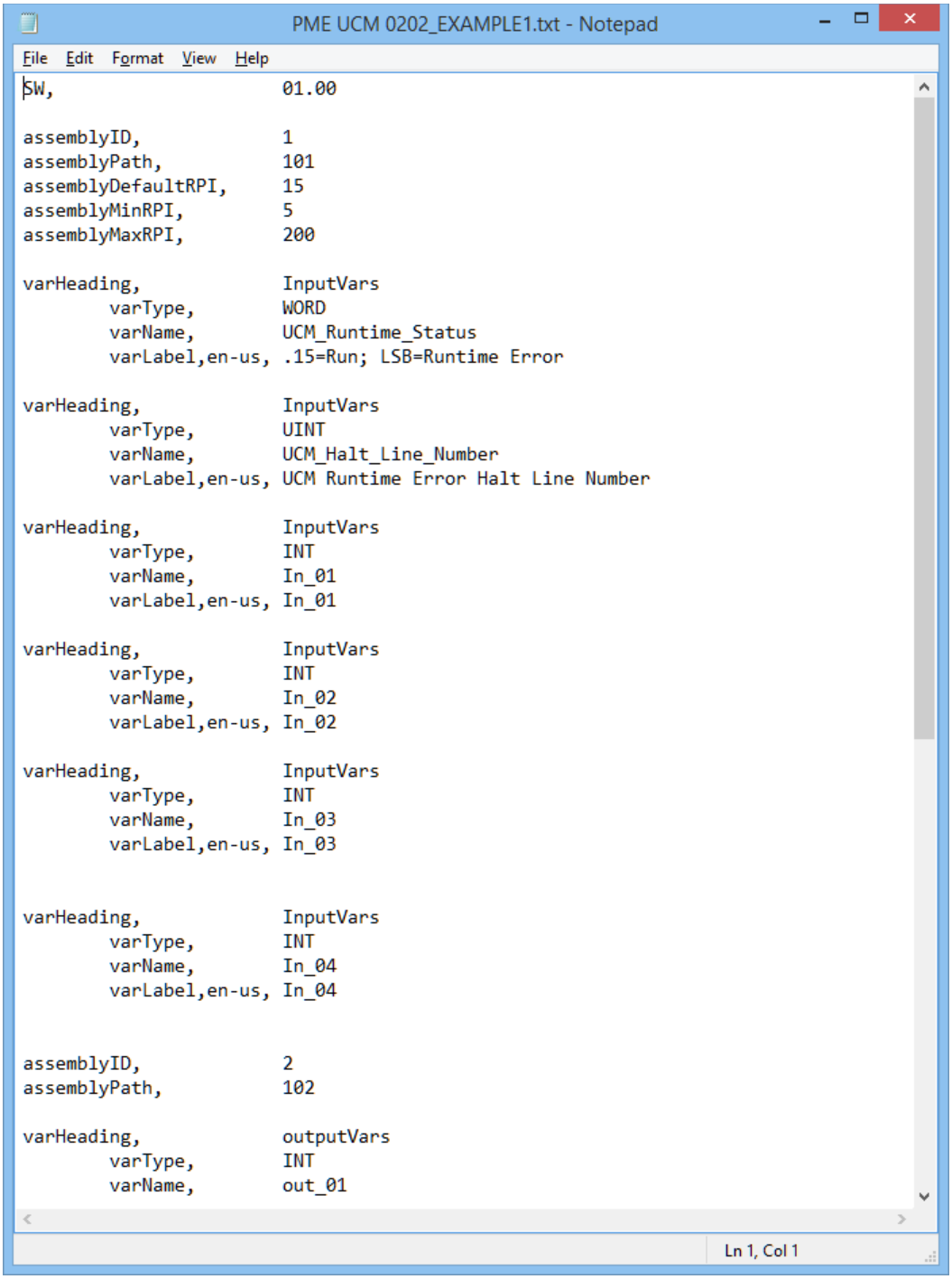
The Freshness BOOL values are 0 if the PLC is unable to connect with the PMEUCM via Ethernet/IP across the backplane. If the data connection is working, the Freshness variables will have the value 1.

The Input structure includes 76 bytes of PTK header information. Most of this can be ignored in normal operation. The UCM data starts after the “Free2” BYTE. Variables named FreeX are place holder variables that are used to align certain variable types to 2, 4, or 8 byte boundaries in the PLC memory.

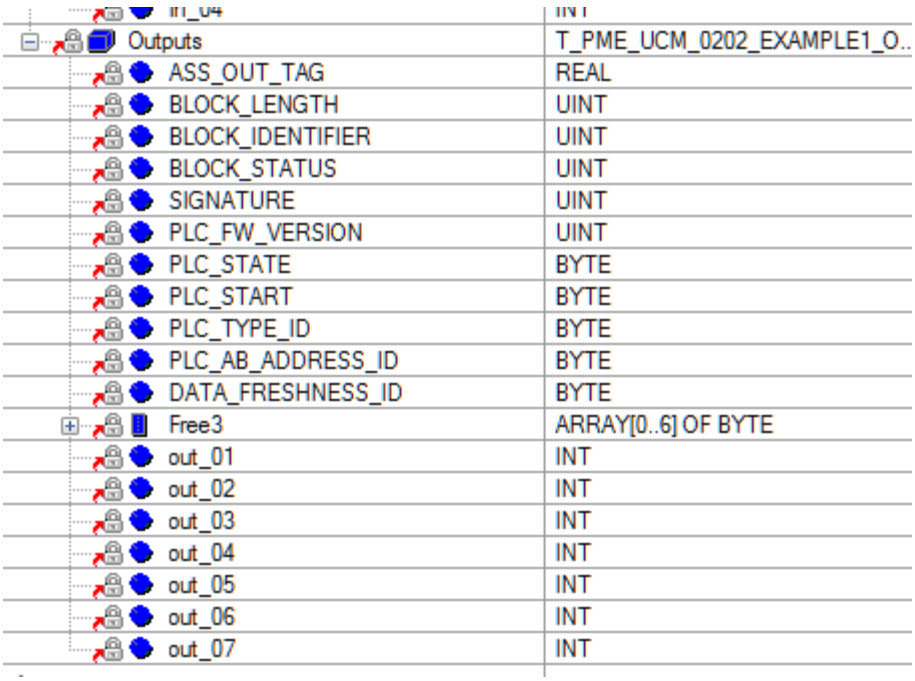
A closer look at the UCM variables shows they match the names, types, and descriptions from the txt file.


Unity Pro XL : EXAMPLE1* - [Data Editor]
File Edit View Services Tools Build PLC Debug Window Help
DTM Browser
Host PC
< 192.168.10.1 > BMEP58_ECPU
< EtherNet IP:192.168.10.3 > PME
Project Browser
Structural view
Derived FB Types
Variables & FB instances
Elementary Variables
Derived Variables
Device DDT Variables
IO Derived Variables
Elementary FB Instance
Derived FB Instances
Motion
Communication
Program
Tasks
MAST
Variables
DDT Types
Function Blocks
DFB Types
Filter
Name
EDT
DDT
IODDT
Name
Type
BMEP58_ECPU
T_BMEP58_ECPU
PME_UCM_0202_EXAMPLE1
T_PME_UCM_0202_EXAMPLE1
Freshness
BOOL
Freshness_1
BOOL
Inputs
T_PME_UCM_0202_EXAMPLE1_IN
DEVICE_NAME
string[64]
DEVICE_STATUS
UINT
DEVICE_STATE
BOOL
DEVICE_HEALTH
BOOL
SPI_ERROR
BOOL
INIT_PARAM_CORRUPTED_ERR...
BOOL
RUNTIME_CONFIGURATION_ER...
BOOL
CONTROL_FW_MISSING_OR_CO...
BOOL
CONTROL_FW_DOWNLOAD_IN_...
BOOL
Free0
BYTE
Free1
BYTE
ETH_STATUS
BYTE
PORT1_LINK
BOOL
CCOTF_IN_PROGRESS
BOOL
REDUNDANCY_OWNER
BOOL
GLOBAL_STATUS
BOOL
SERVICE_STATUS
BYTE
SNTP_SERVICE
BOOL
SNMP_SERVICE
BOOL
FDR_SERVICE_B1
BOOL
FDR_SERVICE_B2
BOOL
FDR_SERVICE_B3
BOOL
FDR_SERVICE_B4
BOOL
Data Editor
Analyzing...
Project settings disabled : Maintain output links on disabled EF (EN=0)
Process succeeded : 0 Error(s) , 0 Warning(s)
Analyze Project
Import/export
User errors
FDT log event
Search/Replace
Ready
HMI R/W mode
OFFLINE
TCPIP

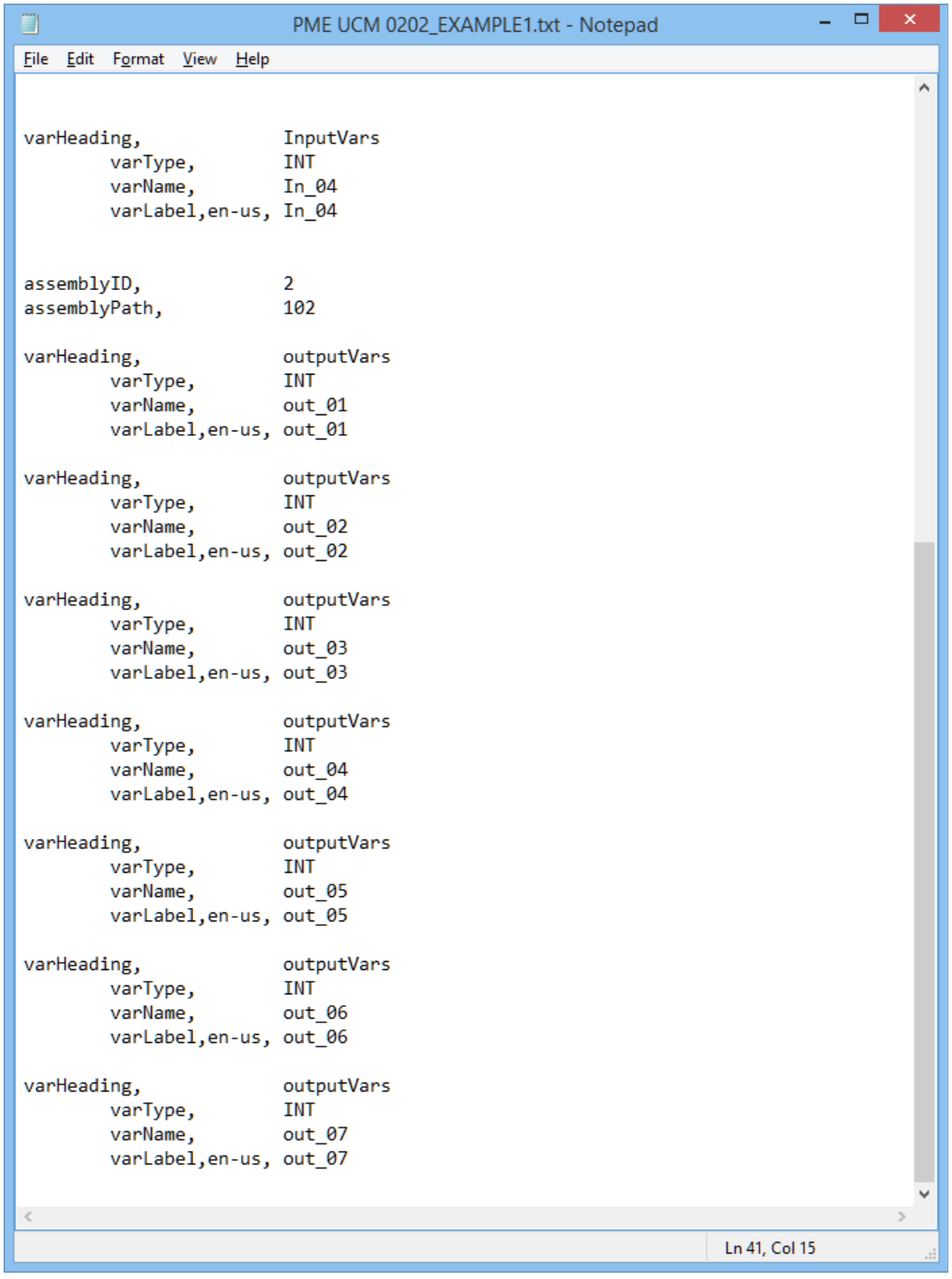

ETH_PORT1_FUNCTION_B2
BOOL
Free2
BYTE
UCM_Runtime_Status
WORD
UCM_Halt_Line_Number
UINT
In_01
INT
In_02
INT
In_03
INT
In_04
INT


PME UCM 0202_EXAMPLE1.txt - Notepad
File Edit Format View Help
SW, 01.00
assemblyID, 1
assemblyPath, 101
assemblyDefaultRPI, 15
assemblyMinRPI, 5
assemblyMaxRPI, 200
varHeading, InputVars
varType, WORD
varName, UCM_Runtime_Status
varLabel,en-us, .15=Run; LSB=Runtime Error
varHeading, InputVars
varType, UINT
varName, UCM_Halt_Line_Number
varLabel,en-us, UCM Runtime Error Halt Line Number
varHeading, InputVars
varType, INT
varName, In_01
varLabel,en-us, In_01
varHeading, InputVars
varType, INT
varName, In_02
varLabel,en-us, In_02
varHeading, InputVars
varType, INT
varName, In_03
varLabel,en-us, In_03
varHeading, InputVars
varType, INT
varName, In_04
varLabel,en-us, In_04
assemblyID, 2
assemblyPath, 102
varHeading, outputVars
varType, INT
varName, out_01
Ln 1, Col 1

The Output variable structure includes the 26 byte PTK header information followed by the UCM data.



Outputs	T_PME_UCM_0202_EXAMPLE1_O..
ASS_OUT_TAG	REAL
BLOCK_LENGTH	UINT
BLOCK_IDENTIFIER	UINT
BLOCK_STATUS	UINT
SIGNATURE	UINT
PLC_FW_VERSION	UINT
PLC_STATE	BYTE
PLC_START	BYTE
PLC_TYPE_ID	BYTE
PLC_AB_ADDRESS_ID	BYTE
DATA_FRESHNESS_ID	BYTE
Free3	ARRAY[0..6] OF BYTE
out_01	INT
out_02	INT
out_03	INT
out_04	INT
out_05	INT
out_06	INT
out_07	INT


PME UCM 0202_EXAMPLE1.txt - Notepad
File Edit Format View Help
varHeading, InputVars
varType, INT
varName, In_04
varLabel,en-us, In_04
assemblyID, 2
assemblyPath, 102
varHeading, outputVars
varType, INT
varName, out_01
varLabel,en-us, out_01
varHeading, outputVars
varType, INT
varName, out_02
varLabel,en-us, out_02
varHeading, outputVars
varType, INT
varName, out_03
varLabel,en-us, out_03
varHeading, outputVars
varType, INT
varName, out_04
varLabel,en-us, out_04
varHeading, outputVars
varType, INT
varName, out_05
varLabel,en-us, out_05
varHeading, outputVars
varType, INT
varName, out_06
varLabel,en-us, out_06
varHeading, outputVars
varType, INT
varName, out_07
varLabel,en-us, out_07
Ln 41, Col 15

Steps for Modifying the Installed DTM

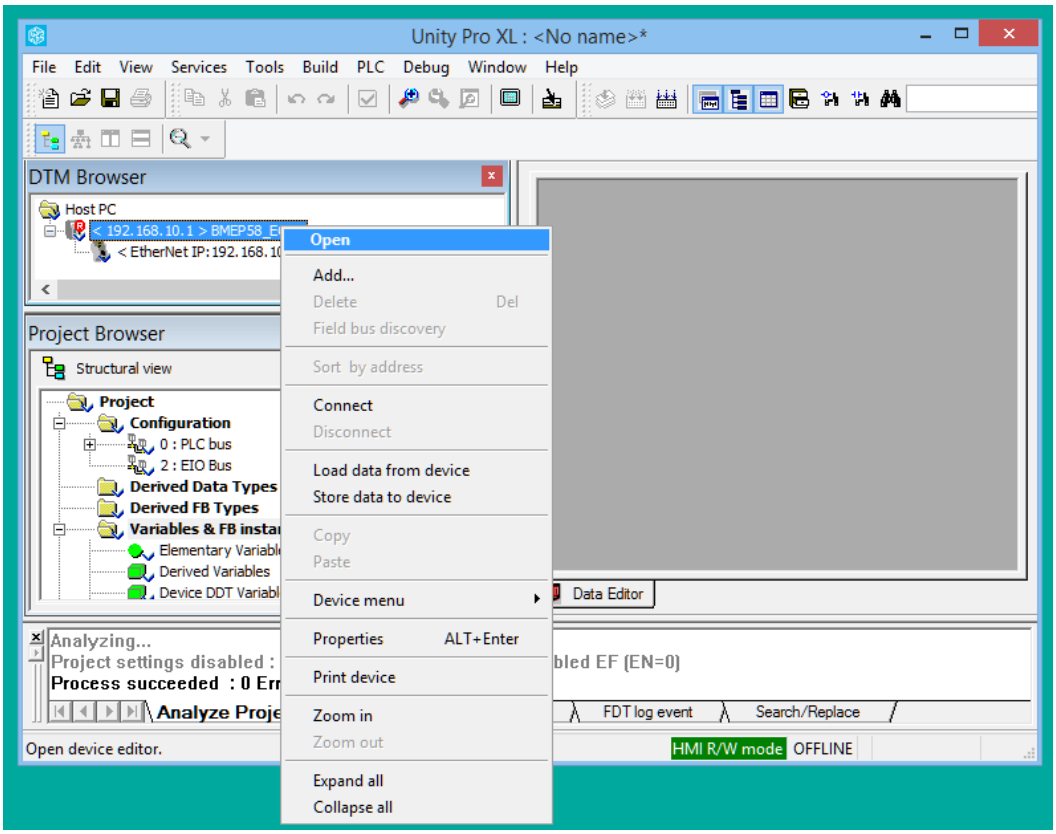
1. Edit the txt file.
2. Save the changes in the txt file.
3. “Install” the txt file with the NRD DTM Util.
4. Perform an UPDATE on the DTM Hardware Catalog in Unity Pro.
5. Delete the UCM DTM item from the DTM Browser.
6. Add the UCM DTM item back into the DTM Browser.
7. BUILD or ANALYZE the Project to generate the new variables.

NOTE: Step 5 is vitally important. Simply updating the DTM Hardware Catalog does not affect installed DTMs. Installed DTMs must be removed and then Added for the changes to the DTM to occur.

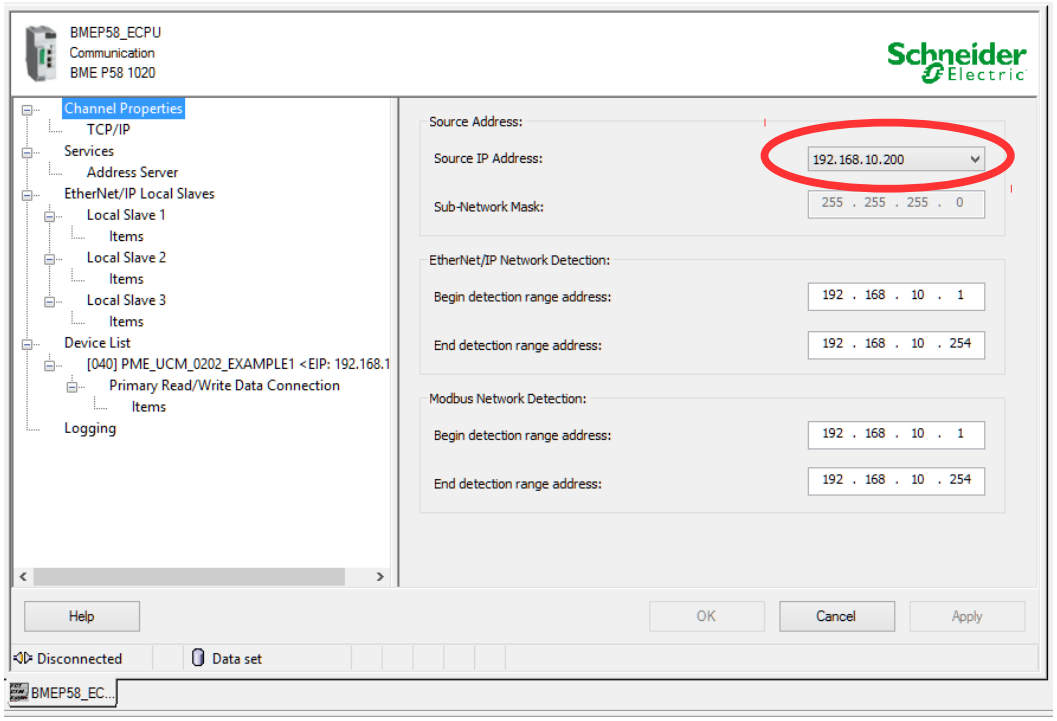
Link the DTM to the PMEUCM Hardware

Once the variable lists have been finalized, 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.

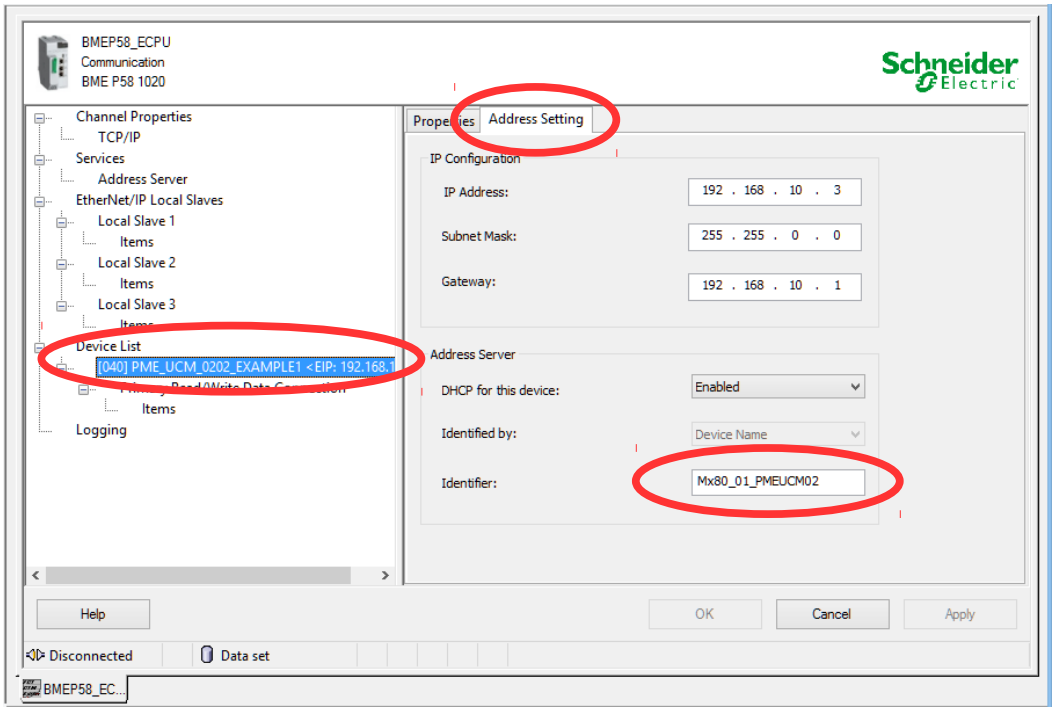


NOTE: The “Source IP Address” is a pull-down listing of all of the IP Addresses of the Unity Pro PC. Make sure to select an address that is on the same subnet as the M580 PLC. In this case the IP Address of 192.168.100.200 is selected since the PLC is at 192.168.10.1.



After Setting the Source IP Address, click on the PME_UCM_0202_… entry in the list on the left.

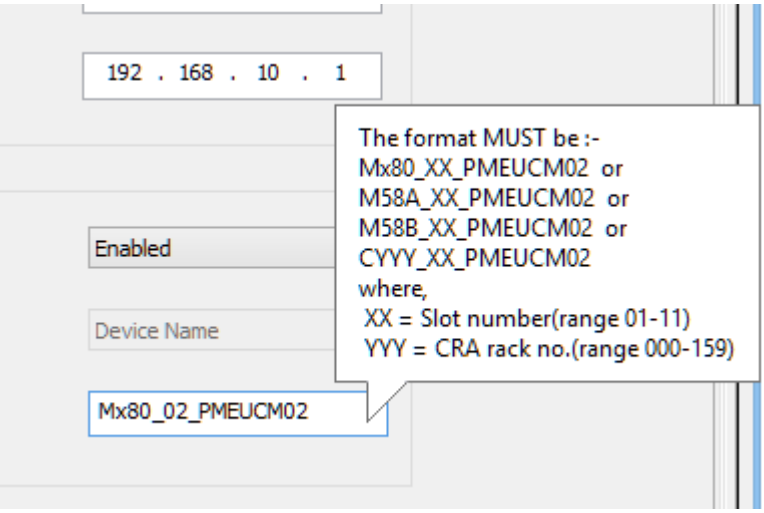
Select the “Address Setting” Tab.



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 3. Therefore, the Identifier must be set for “Mx80_02_PMEUCM02”.

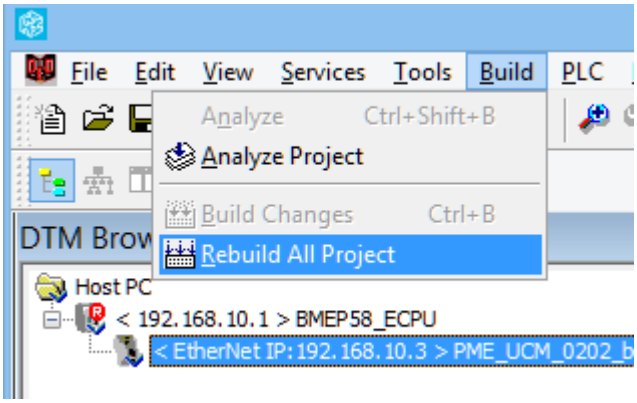


NOTE: If the PMEUCM is located in a remote rack, the YYY value is the thumbwheel (rotary switches) setting of the eCRA, not necessarily the logical rack number.

Build All

After setting the Identifier, it is time to do a Build All of the Project.

Select “Build > Rebuild All Project”

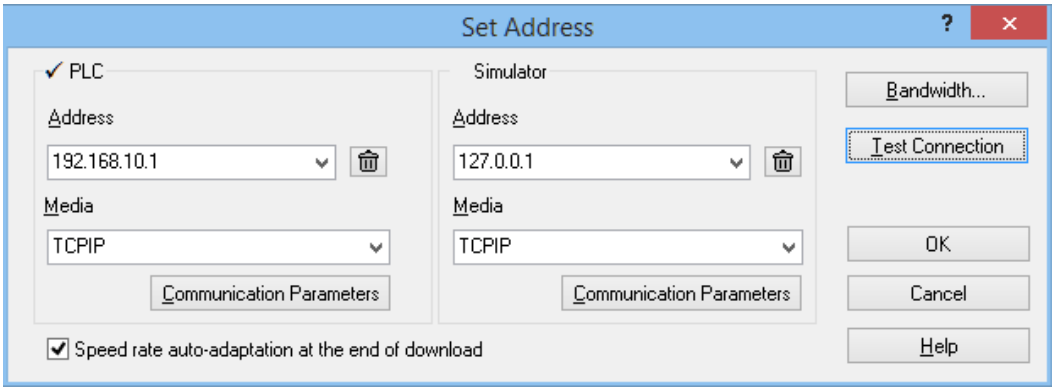


Transfer Project to PLC

After a successful Build, it is time to transfer the project to the M580. Connect the Ethernet port of the PC to the Service Port of the M580.

PLC Set Address

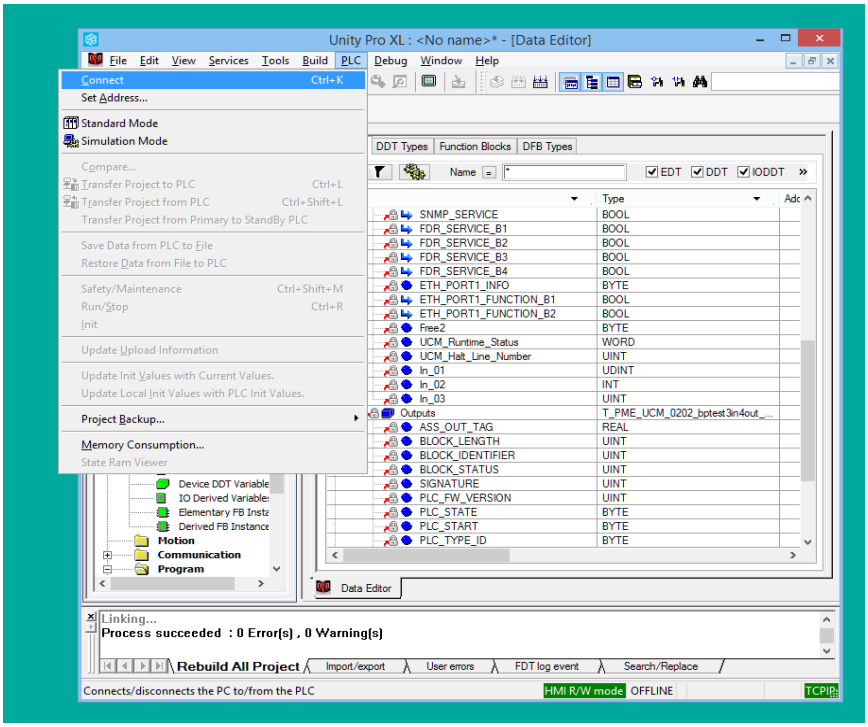
Select PLC > Set Address and choose TCPIP for the Media and set the Address of the M580 (192.168.10.1).



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

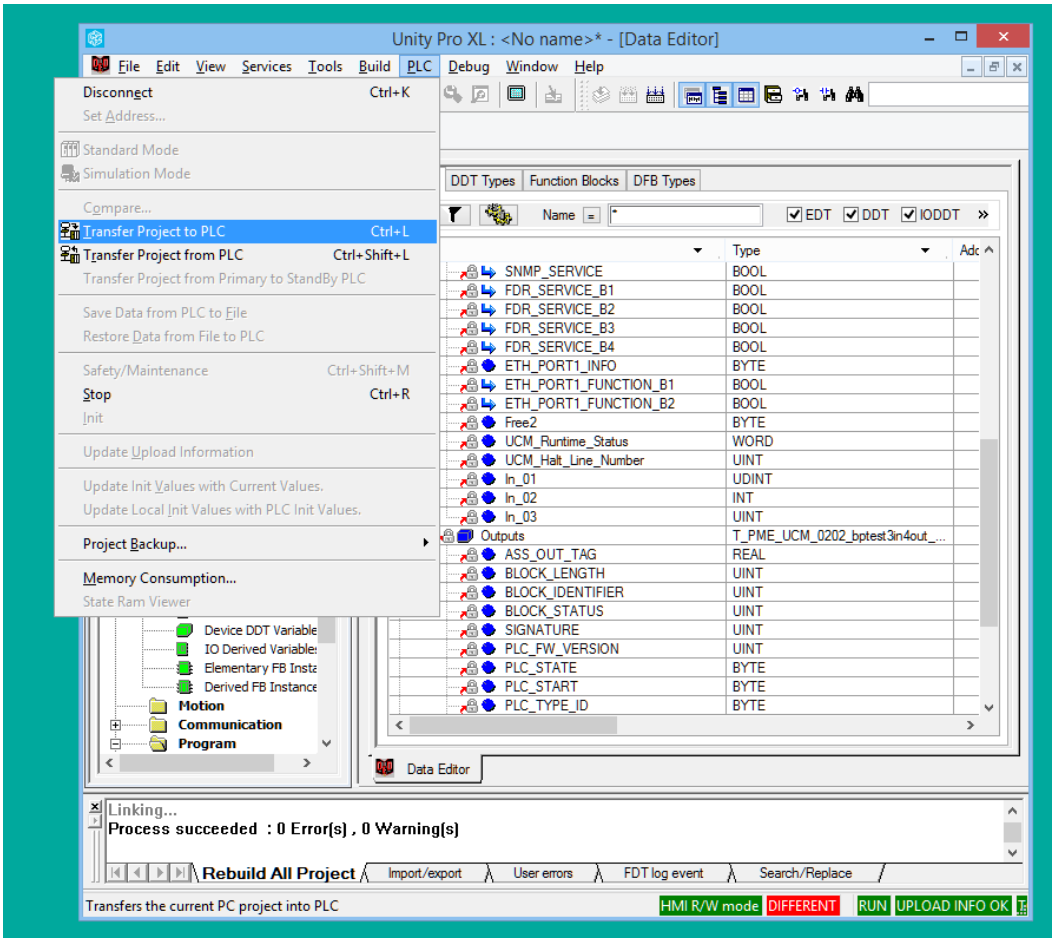
PLC Connect

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

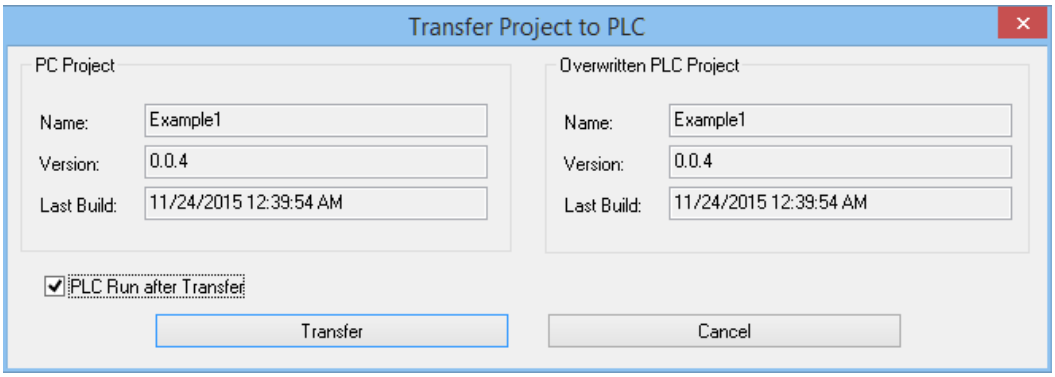


Transfer Project to PLC

After connecting, transfer the project to the PLC.



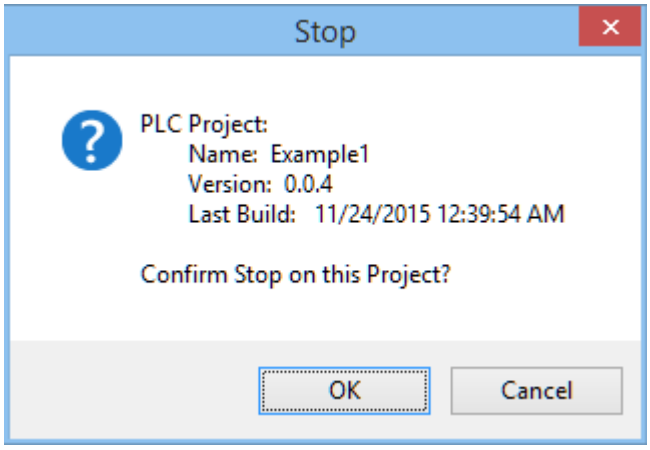
The Transfer Project to PLC window should look something like this:



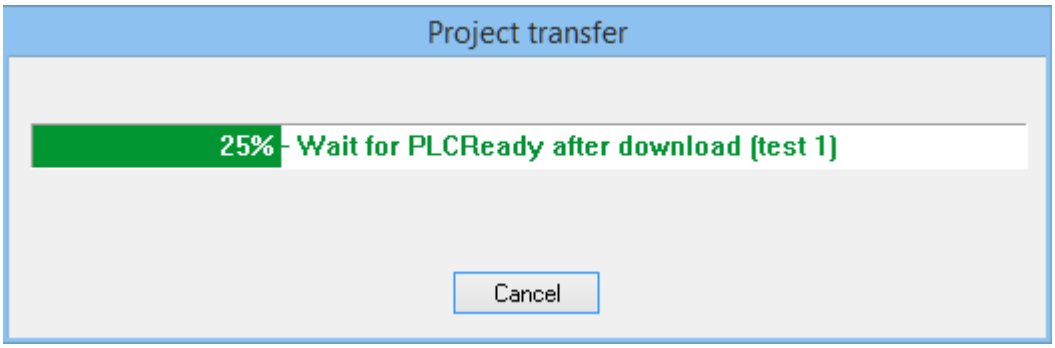
It is usually convenient to check the PLC Run after Transfer box.

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

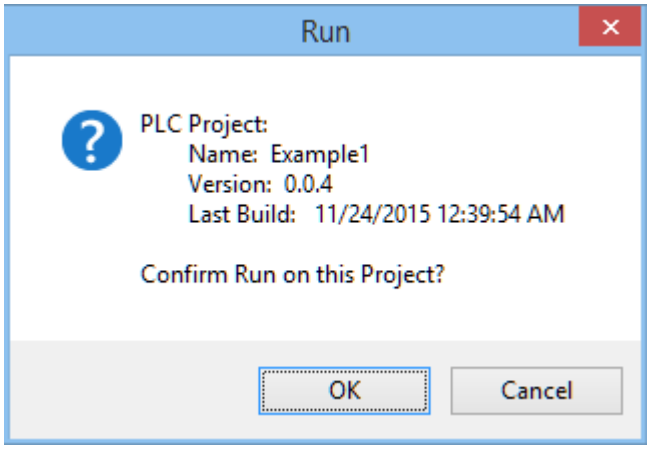
WARINING: Stopping a running PLC may result in injury or death. Make sure that you understand the consequences of halting a running program.



The transfer should look like this:



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

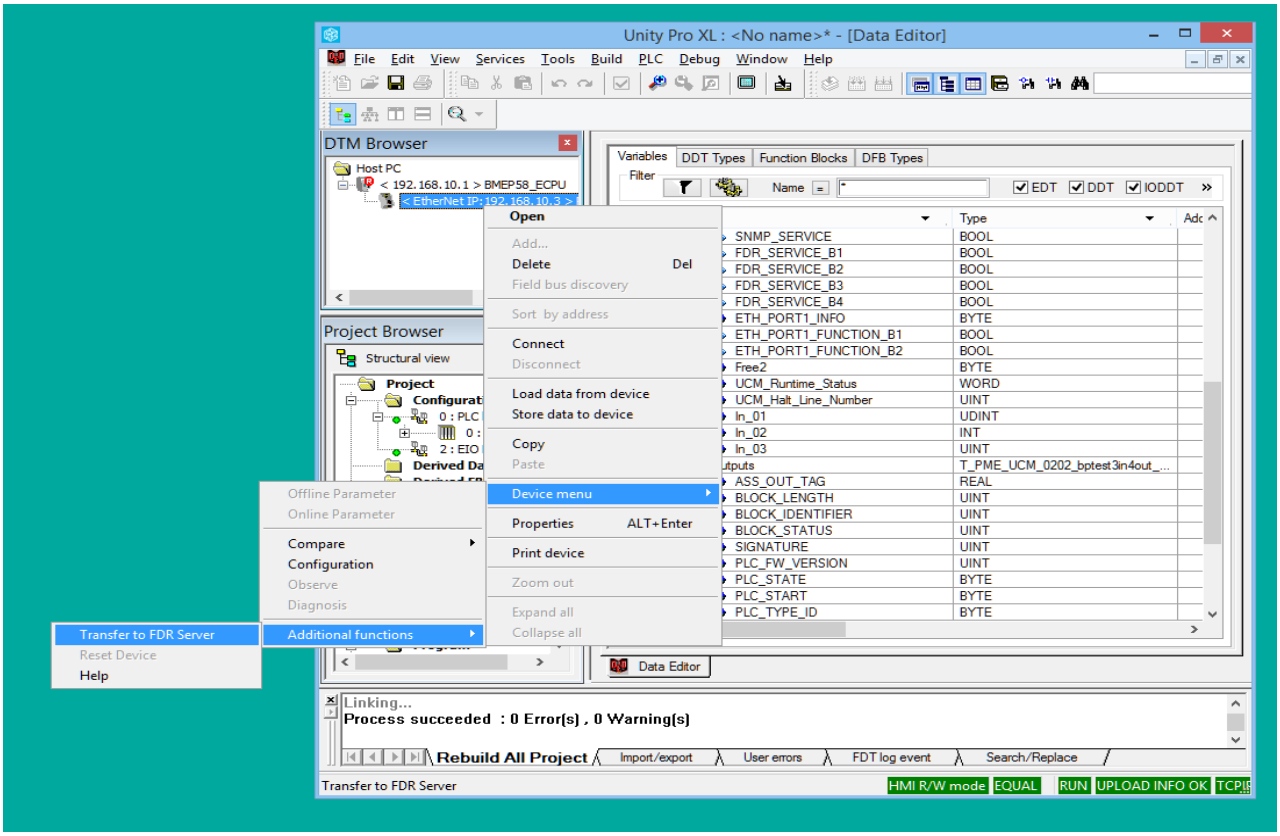


Selecting “OK” will start the PLC.

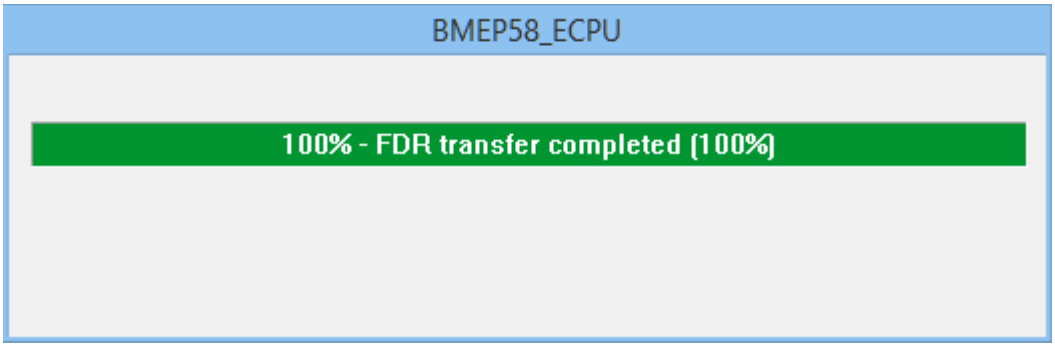
Transfer to FDR Server

An additional step that must be followed after any configuration change within the DTM is the “Transfer to FDR Server”.

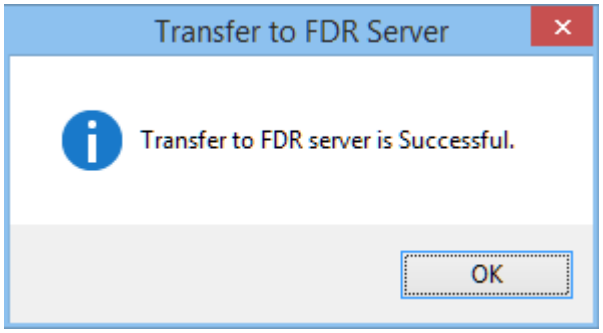
This is accomplished by right clicking on the UCM's entry in the DTM Browser, select “Device Menu”, “Additional Functions”, and finally “Transfer to FDR Server”



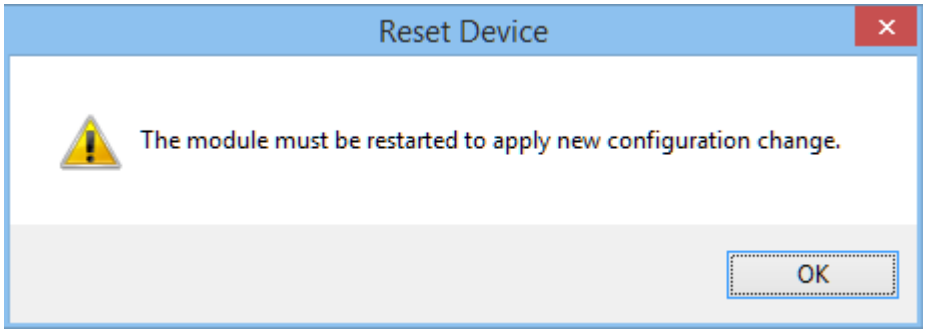
A progress box will be shown during the transfer.



A confirmation box will be shown to show the success of the transfer.



A box will then be shown to remind the user of the requirement to cycle power on the PTK board (entire PMEUCM module) before the new settings will be used.



Cycle Power on the PMEUCM

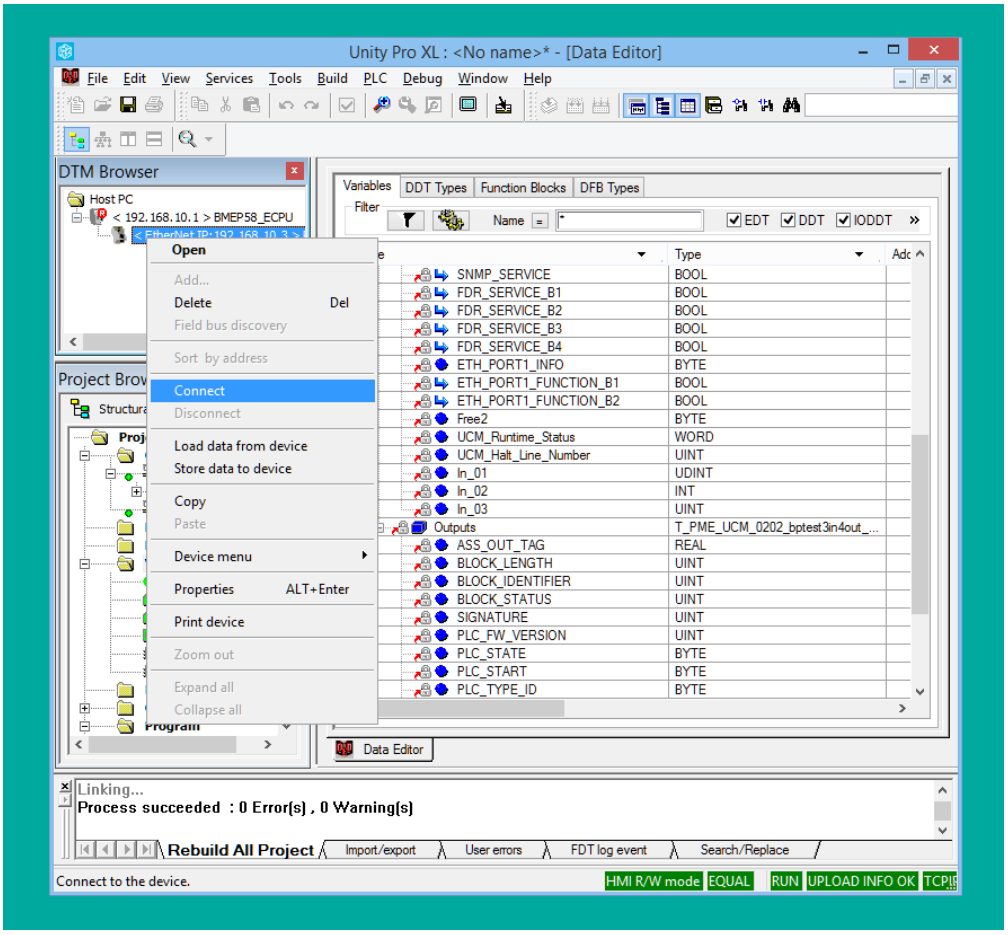
There are three methods of cycling power on the PMEUCM.

1. The quickest is to press the “Reset” button on the power supply of the PMEUCM's rack.

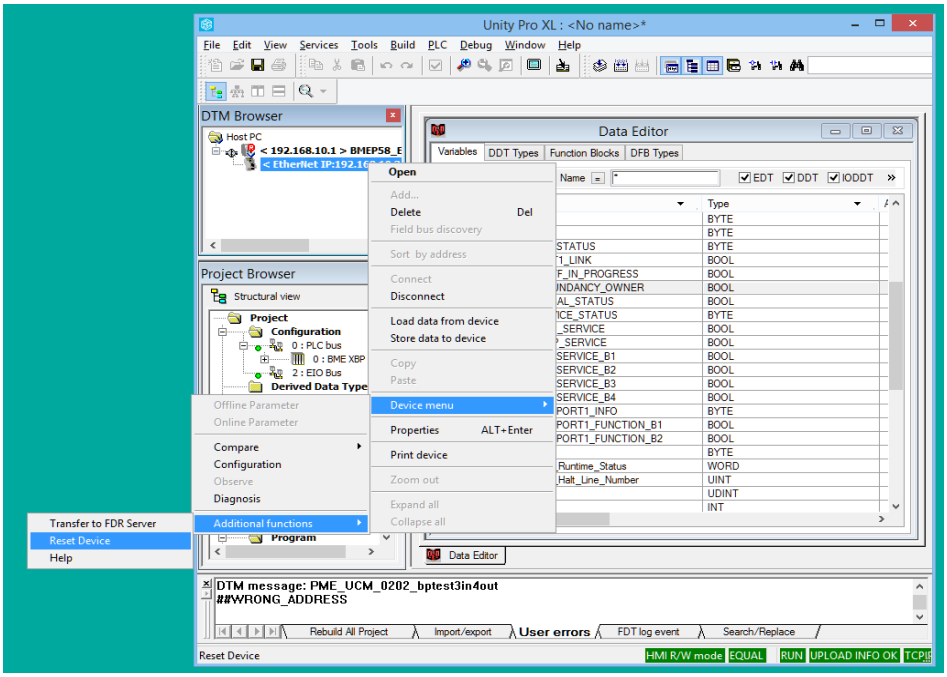
   WARTING: Cycling power on the rack will also restart every device in that rack which may result in injury or death.

2. Remove the mounting screw on the PMEUCM and remove it from the powered rack. Re-install the PMEUCM into the rack to cycle power on the entire device.

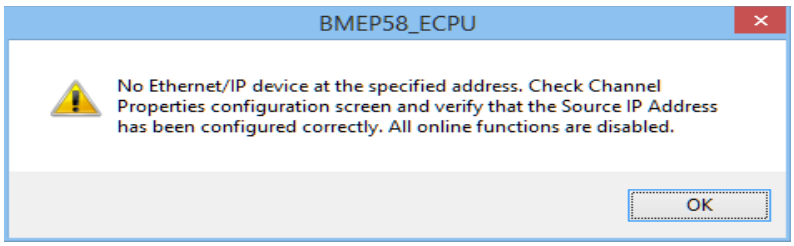
3. Remote resetting of the PMEUCM may be done through the DTM Browser. Right click on the PMEUCM entry in the DTM Broweser and select “Connect”.

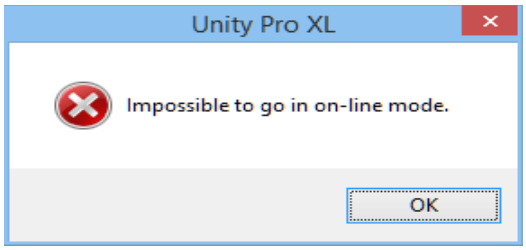


If the connection is successful you will be able to select: Device Menu > Additional Functions > Reset Device.



Otherwise this windows will be shown:



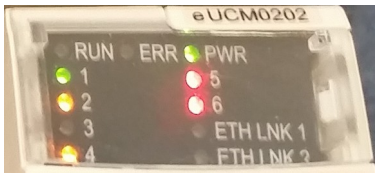


Cycling power on the PMEUCM is the most reliable method for forcing the PTK card to reboot.

9 Using Example1

LED Panel

Most of the LED indicators on the top panel are controlled by the user application. Example1 blinks many of these lights at different rates just for fun. Other LEDs are controlled by the PTK board or the UCM operating system.



PTK Board Controlled Lights

The top three lights are controlled by the PTK board. The meaning of these lights is described in the following table.

<table>
<tr><th>Label</th><th>Color</th><th>Description</th></tr>
<tr><td>RUN</td><td>Green</td><td>ON – The PTK board is properly configured and exchanging data across the backplane with the M580 PLC.
NOTE: This is NOT an indication of the run/halt state of the PLC.</td></tr>
<tr><td>ERR</td><td>Red</td><td>The blink pattern of the ERR light indicates the state of the PTK backplane interface.
<table>
<tr><th>Blink Rate</th><th>Meaning</th></tr>
<tr><td>Steady OFF</td><td>If RUN light is ON – No errors present
If RUN light is OFF – No backplane configuration</td></tr>
<tr><td>Steady ON</td><td>UCM has not started backplane operation or invalid MAC address or no Ethernet link</td></tr>
<tr><td>2 blinks ON</td><td>No IP Address</td></tr>
<tr><td>3 blinks ON</td><td>Invalid configuration</td></tr>
<tr><td>4 blinks ON</td><td>Duplicate IP Address</td></tr>
<tr><td>5 blinks ON</td><td>Awaiting served IP Address</td></tr>
<tr><td>6 blinks ON</td><td>IP Address invalid</td></tr>
<tr><td>7 blinks ON</td><td>Error on UCM board</td></tr>
</table></td></tr>
<tr><td>PWR</td><td>Green</td><td>The PMEUCM has proper 24Vdc power from the Ethernet backplane when lit.</td></tr>
</table>

UCM OS Controlled Lights

The ETHLINK1 and ETHLINK2 lights are controlled by the UCM operating system.

Label	Color	Description
ETHLINK1	Green	ON – Link OK on port E1 OFF – NO Ethernet Link on port E1
ETHLINK2	Green	ON – Link OK on port E2 OFF – NO Ethernet Link on port E2

USER Controlled Lights

The LEDs labeled 1 through 6 on the top panel are under control of the running application. Lights 1, 2, 3, 4, and 6 are blinked at various timed rates as an example for controlling these lights. Light 5 is used to indicate the UCM's status of the communication with the M580 PLC.

NOTE: While the label on the top panel shows the number 1 through 6, the UCM programming language labels these lights as 3 through 8. This is because UCM lights 1 and 2 are the red LEDs behind the LCD to maintain compatibility with other Niobrara UCM products with these two LEDs behind the LCD (MUCM, QUCM, DUCM, and CUCM).

Panel Label	UCM Programming Reference	Color	Example1 Functional Description
1	3	Green	Toggles at 50mS rate
2	4	Yellow	Toggles at 100mS rate
3	5	Green	Toggles at 200mS rate
4	6	Yellow	Toggles at 400mS rate
5	7	Red	ON – Data comms with PLC BAD OFF – Data comms with PLC GOOD
6	8	Red	Toggles at 800mS rate

Example code for blinking lights

At the end of Thread 1 of the program Example1.ucm2 are several timers that expire to change on/off state of the blinking lights.

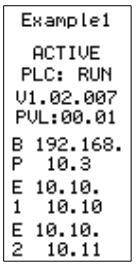
```
if expired(lighttimer[3]) then
        set timer lighttimer[3] 50
        toggle light 3
endif
if expired(lighttimer[4]) then
        set timer lighttimer[4] 100
        toggle light 4
endif
if expired(lighttimer[5]) then
        set timer lighttimer[5] 200
        toggle light 5
endif
if expired(lighttimer[6]) then
        set timer lighttimer[6] 400
        toggle light 6
endif
if expired(lighttimer[8]) then
        set timer lighttimer[8] 800
        toggle light 8
endif
```

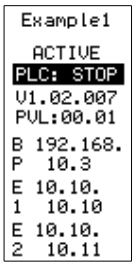
LCD and Joystick Operation

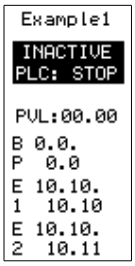
The front panel LCD provides status information about the PMEUCM and user interaction with the setup and operation of the card/application.

Most of the UCM code for controlling the screen and handling the joystick input is in Thread 2 of Example1.ucm2. In fact, most of the code for this program is the screen driver.

The information displayed on the "splash" screen varies depending on the configuration and state of the PTK board.







The text "ACTIVE" or "INACTIVE" describes the backplane operation. ACTIVE means that the UCM is exchanging data with the M580 while INACTIVE means that it is not.

PLC: RUN or STOP shows the UCMs understanding of the state of the M580 CPU.

NOTE: RUN is only possible if the connection is ACTIVE. STOP will always be displayed if the connection is INACTIVE. This description may not actually represent the RUN/STOP state of the CPU if the link is INACTIVE.

V1.02.007 shows the software version of the PTK board. This value may not be displayed if the UCM is unable to configure the PTK board. The most likely reason is that the PMEUCM is installed in an unconfigured slot of an active PLC system.

PVL:00.01 shows the PVL version of the SPI interface between the UCM board and the PTK board.

BP IP Address indicates the actual IP Address on the Ethernet backplane. In the above screen shots, the PTK board is at 192.168.10.3 or 0.0.0.0. If the PMEUCM is mounted in a powered slot of a rack without a PLC, the PTK board will revert to an IP address determined by its MAC address.

E1 IP Address shows the IP Address of the E1 port on the PMEUCM. By default it is 10.10.10.10.

E2 IP Address shows the IP Address of the E2 port on the PMEUCM. By default it is 10.10.10.11.

Backlight

The backlight time is controlled user code. In this case there is a timer that keeps the backlight on for 60000 mS 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

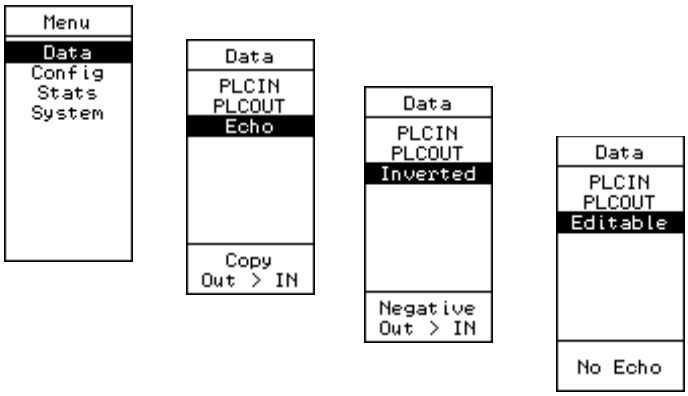
Moving the joystick will cause the application to show various menus to access status or setup screens. Move the highlighted cursor around with the joystick. Typically a right press will act as “Enter” while a left press will act as “Escape”. Sometimes a push in “Enter” is needed (Factory Default for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

Data Echo Config

The Example1 performs one of three different tasks:

1. Echo PLCOUT data to PLCIN data
2. Echo the inversion of PLCOUT to PLCIN
3. Allow the user to manually edit the PLCIN data from the front panel

The mode of operation is set in the Data menu screen by moving selecting the bottom entry and moving the joystick to the right to cycle through the options.



PLCOUT Data

The PLCOUT data may be vied by selecting the PLCOUT menu item. The radix of the displayed value may be changed between SIGNED, UNSIGNED and HEX by moving the joystick right while on the PLCOUT data screen.

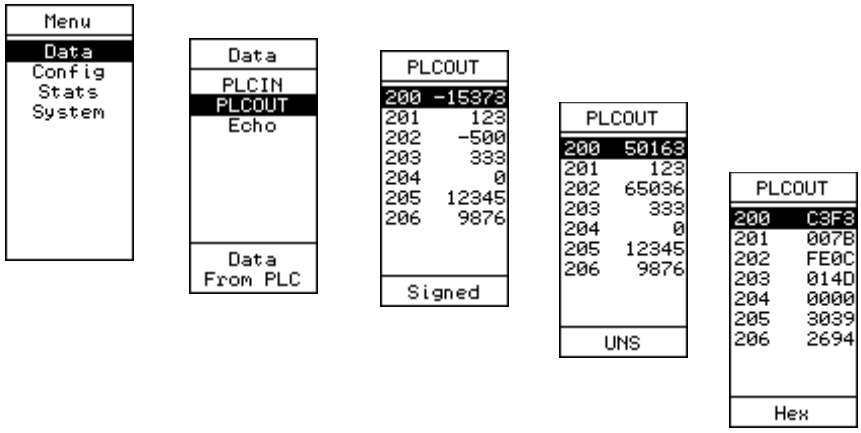
In the following example, an animation table has been built to allow modification of the PLCOUT data to the UCM.

Table[Data Editor]

Modification | Force

Name	Value	Type	Comment
PME_UCM_0202_EXAM...		T_PME_UCM_0...	
Freshness	1	BOOL	Global Freshness
Freshness_1	1	BOOL	Freshness of Object
Inputs		T_PME_UCM_0...	Input Variables
Outputs		T_PME_UCM_0...	Output Variables
ASS_OUT_TAG	0.0	REAL	
BLOCK_LEN...	0	UINT	
BLOCK_IDEN...	0	UINT	
BLOCK_STAT...	0	UINT	
SIGNATURE	0	UINT	
PLC_FW_VE...	0	UINT	
PLC_STATE	0	BYTE	
PLC_START	0	BYTE	
PLC_TYPE_ID	0	BYTE	
PLC_AB_ADD...	0	BYTE	
DATA_FRES...	0	BYTE	
Free3		ARRAY[0..6] OF...	Unused Variable
out_01	-15373	INT	out_01
out_02	123	INT	out_02
out_03	-500	INT	out_03
out_04	333	INT	out_04
out_05	0	INT	out_05
out_06	12345	INT	out_06
out_07	9876	INT	out_07

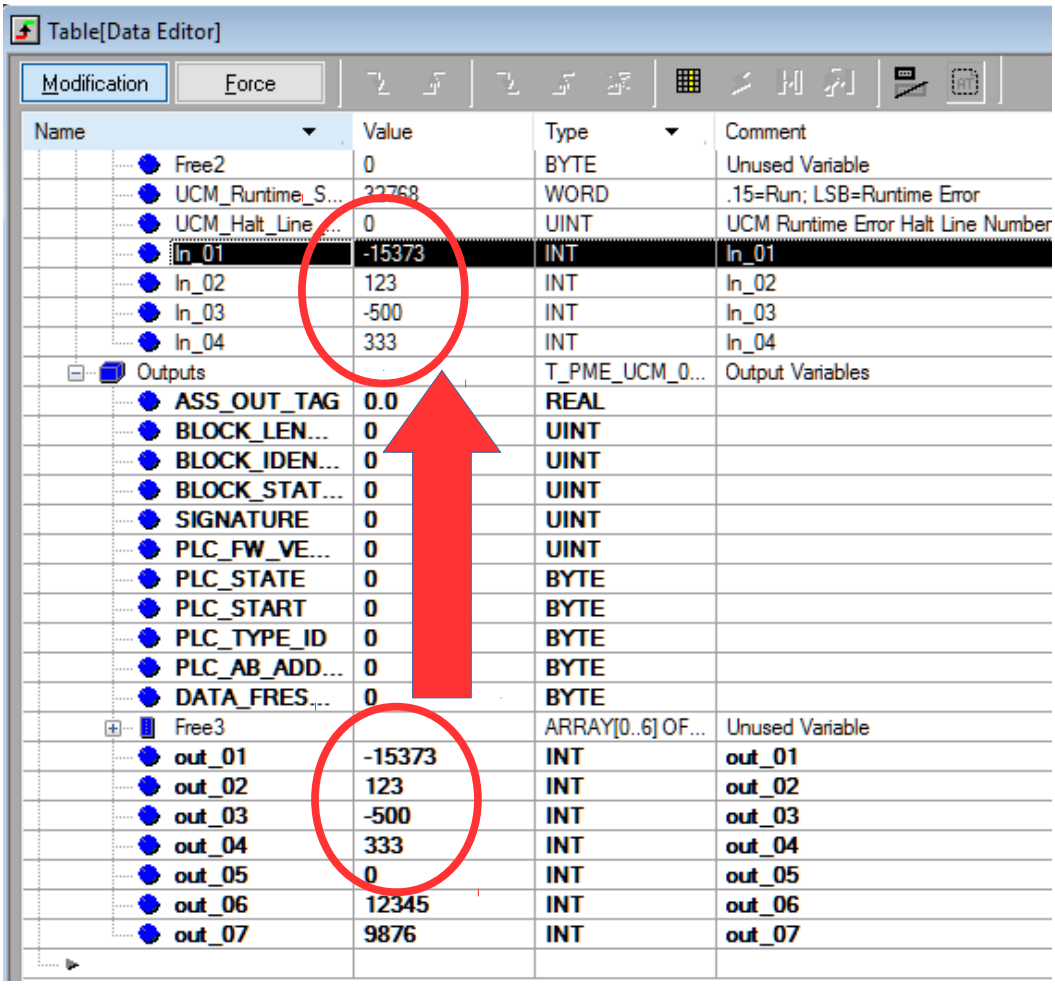
Now, the PLCOUT data may be viewed on the PMEUCM screen:



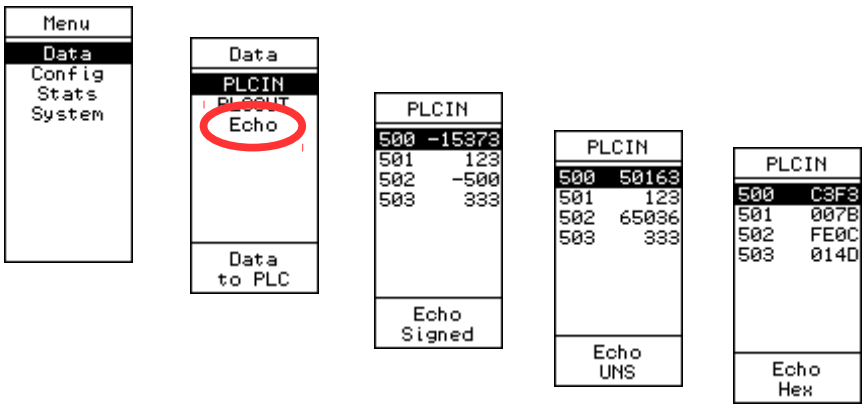
PLCIN Data

The data returned to the M580 by the UCM depends upon the setting of the ECHO, INVERTED, or EDITABLE menu item on the Data menu.

ECHO – The data is copied directly so In_01 = Out_01 and In_02 = Out_02.



Viewing the PLCIN screens:

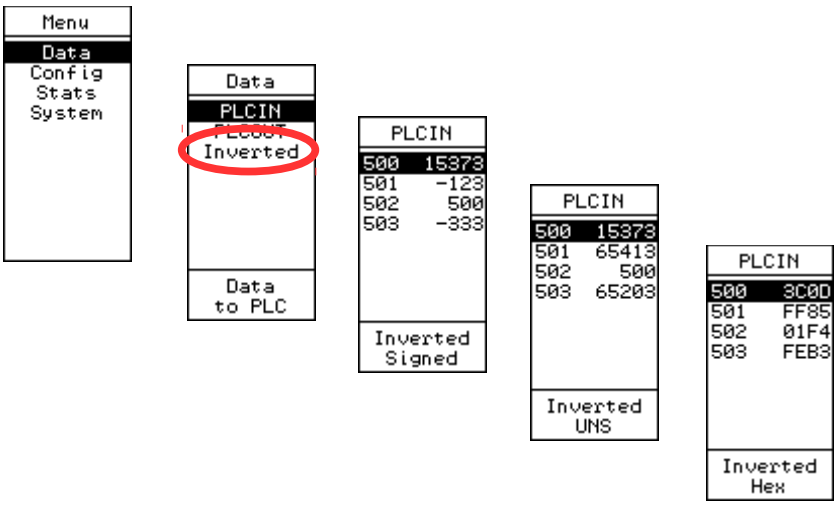


INVERTED – The data is copied and negated so in_01 = -out_01 and in_02 = -out_02.

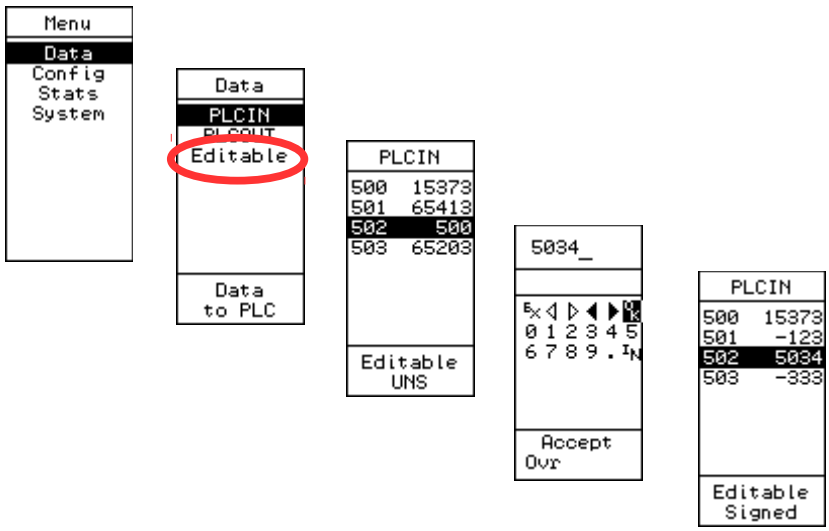
Table[Data Editor]

Modification | Force

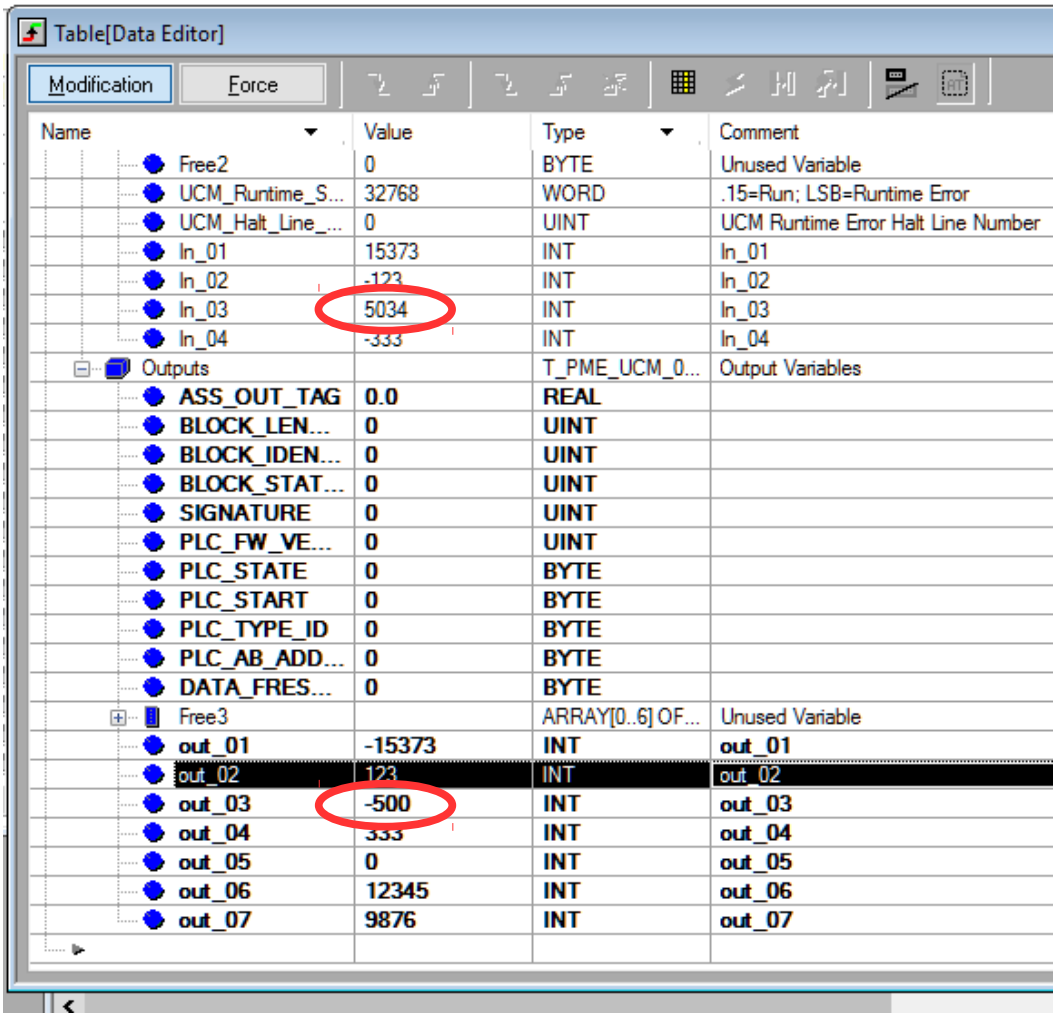
Name	Value	Type	Comment
Free2	0	BYTE	Unused Variable
UCM_Runtime_S...	32768	WORD	.15=Run; LSB=Runtime Error
UCM_Halt_Line_...	0	UINT	UCM Runtime Error Halt Line Number
In_01	15373	INT	In_01
In_02	-123	INT	In_02
In_03	500	INT	In_03
In_04	-333	INT	In_04
Outputs		T_PME_UCM_0...	Output Variables
ASS_OUT_TAG	0.0	REAL	
BLOCK_LEN...	0	UINT	
BLOCK_IDEN...	0	UINT	
BLOCK_STAT...	0	UINT	
SIGNATURE	0	UINT	
PLC_FW_VE...	0	UINT	
PLC_STATE	0	BYTE	
PLC_START	0	BYTE	
PLC_TYPE_ID	0	BYTE	
PLC_AB_ADD...	0	BYTE	
DATA_FRES...	0	BYTE	
Free3		ARRAY[0..6] OF...	Unused Variable
out_01	-15373	INT	out_01
out_02	123	INT	out_02
out_03	-500	INT	out_03
out_04	333	INT	out_04
out_05	0	INT	out_05
out_06	12345	INT	out_06
out_07	9876	INT	out_07



EDITABLE – The PLCIN data is set from the front panel of the PMEUCM.



In this example, the value in register 502 is changed from 500 to 5034. Now the value in the M580 has changed to 5034 while the Out_03 stays at -500.



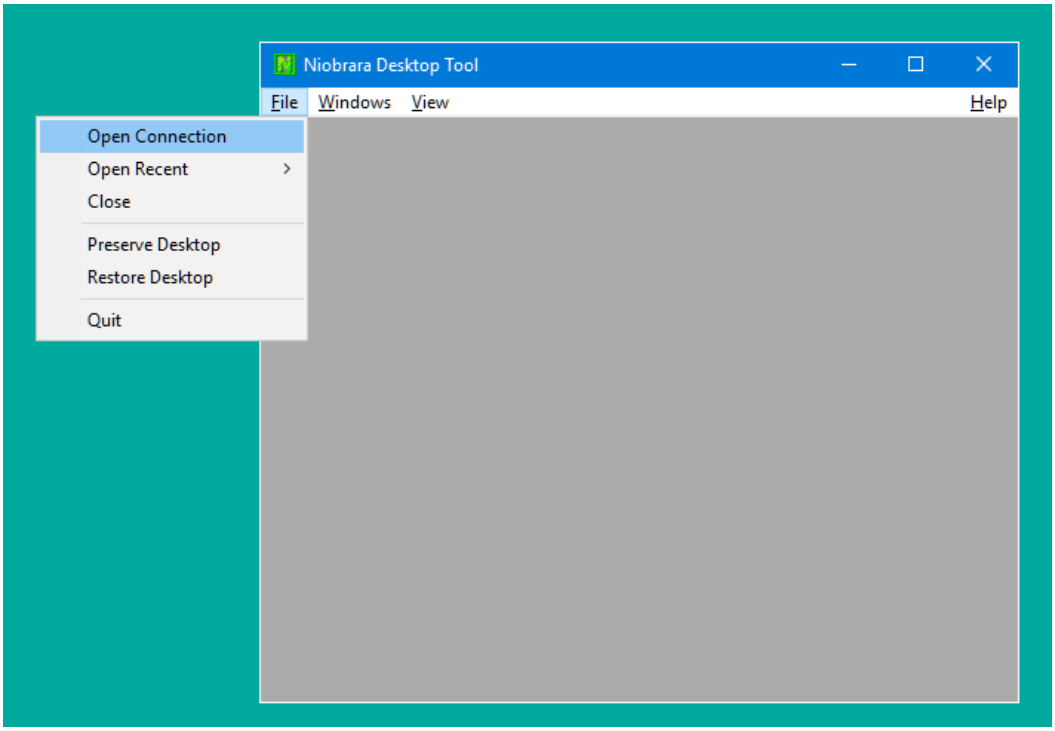
Modbus Registers

A close inspection of Thread 1 in Example1.ucm2 shows that the PLCOUT data is copied to output[200] through output[206]. These output registers may be accessed as Modbus Holding registers through the built-in Modbus/TCP server of the PMEUCM OS.

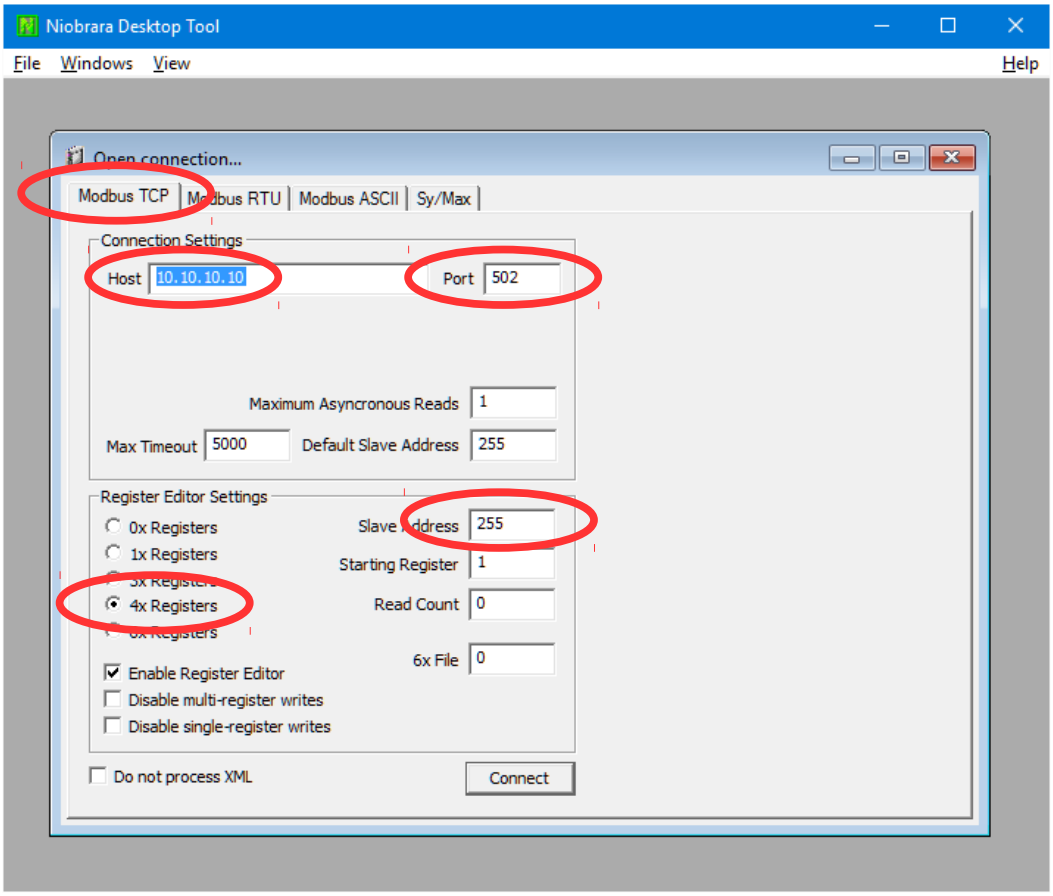
The PLCIN data is copied to output[500] through output[503].

The NRDTOOL program may be used to inspect/modify these Modbus registers.

NRDTOOL may be started by Start > Programs > Niobrara > NRDTOOL



To open a new Modbus/TCP connection to the PMEUCM, Select File >Open Connection.



Select the Modbus TCP tab.

Enter the IP Address of E1 (10.10.10.10) for the Host.

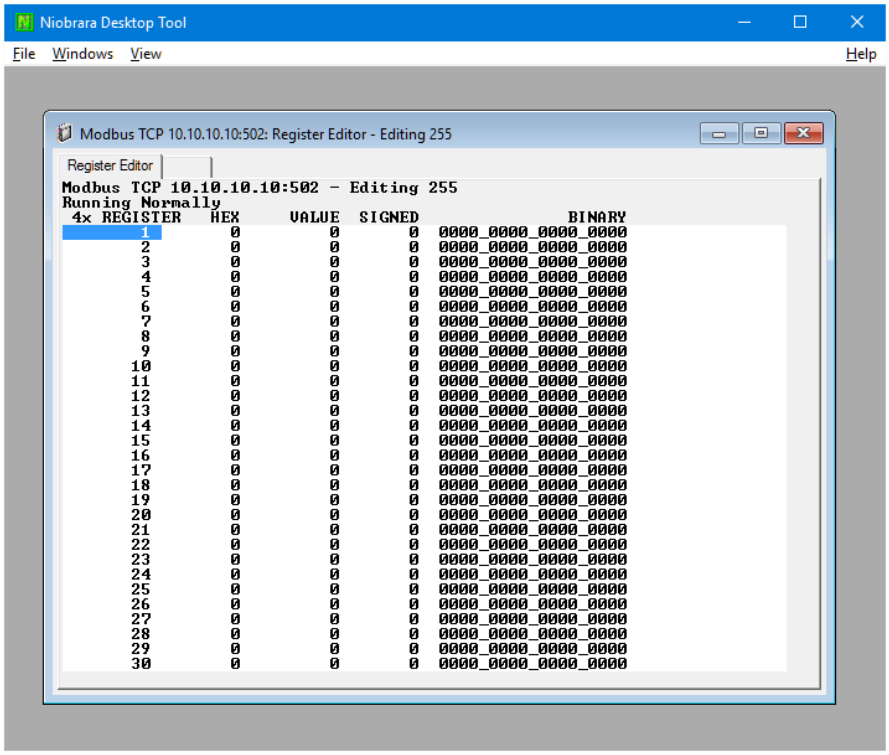
Make sure that the Port is set to match the OS Port of the UCM (502 default).

Make sure that the radio button is set to 4x.

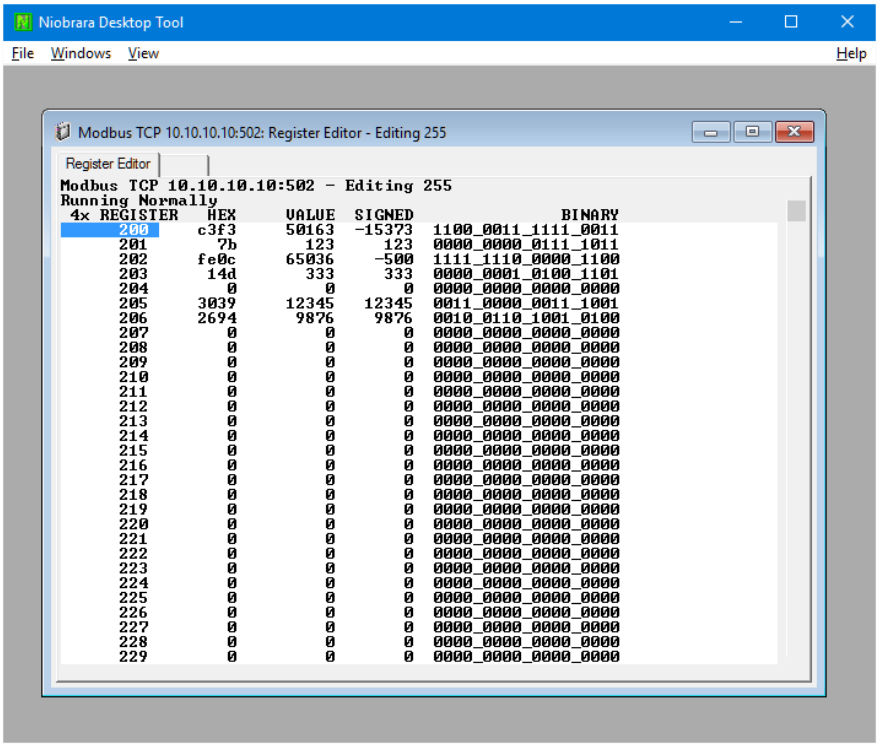
Make sure that the Slave Address is set to 255.

Now click “Connect”

The screen should now show the first block of 4x Holding Registers.

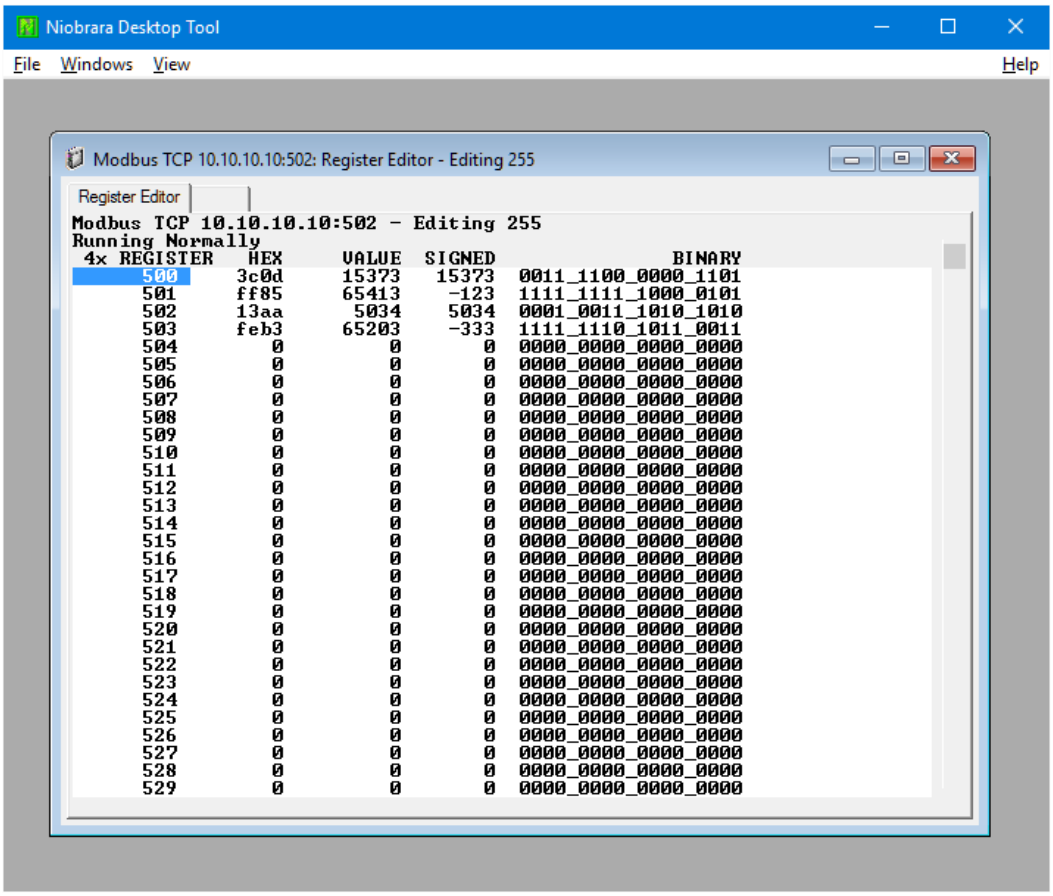


Typing in a value in the blue cursor enters the new value. Typing in 200 in the Register number will cause the viewer to jump to register 200.



Notice that now the PLCOUT data is shown.

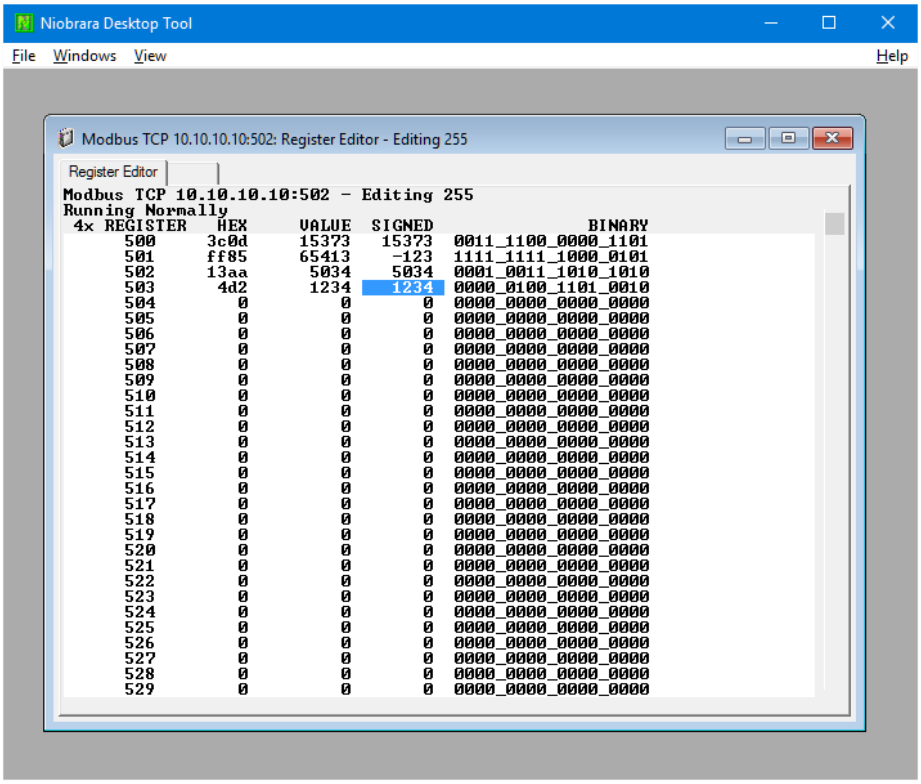
Type in the number 500 in the REGISTER column and the screen will jump to the location of the PLCIN data.



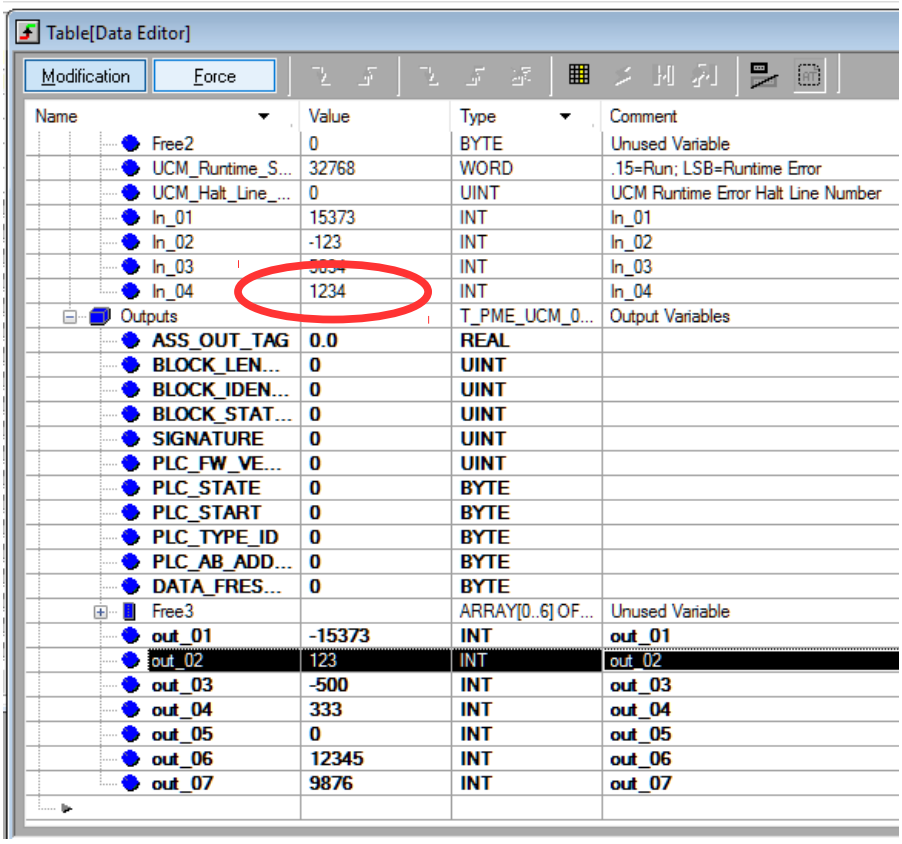
Notice that register 502 has the value 5034 that was entered from the front panel.

Because the DATA configuration is still set for EDITABLE, we can manually enter data in this Modbus viewer and see the new data show up in the M580.

Move the cursor into the SIGNED column on register 503 and change the -333 value to 1234 and press ENTER.



Now look in Unity Pro and see that In_04 has changed to 1234.



Modbus 6x Files

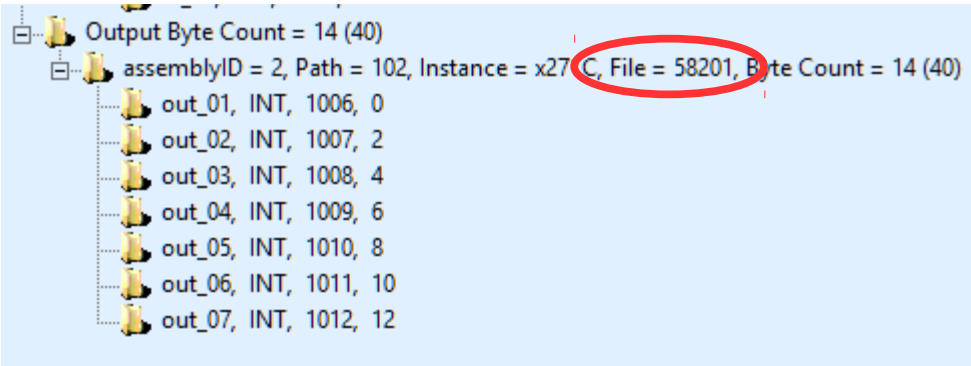
NRDTOOL may be used to directly inspect the data in the Modbus 6x files used for passing data between the PMEUCM and the PTK board.

Inspecting Thread 1 of Examle1.ucm2 shows this line:

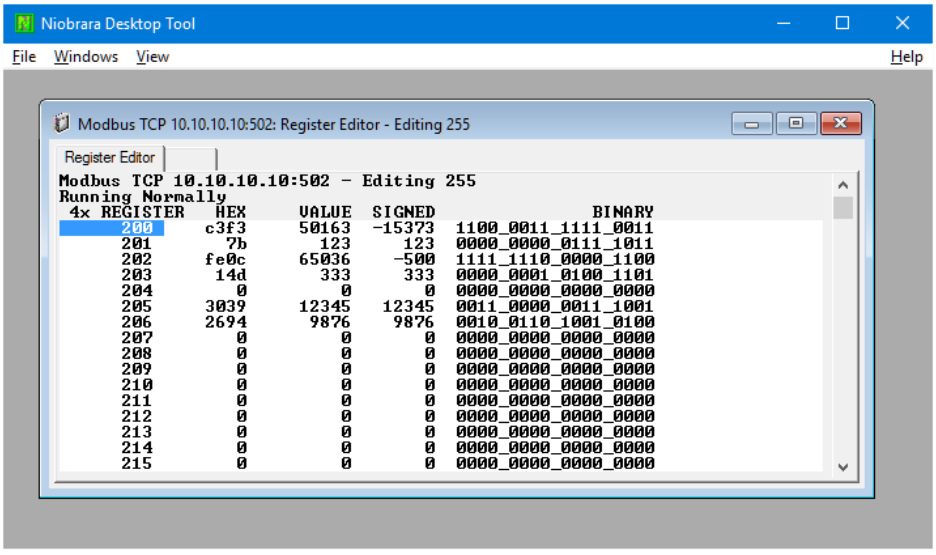
read file 58201 offset 0 BPOutputs { Copy outputs from backplane block 1}

This instruction copies the PLCOUT data from backplane block 1 located in 6x file 58201 into the array of bytes variable BPOutputs. The offset 0 means to start the file read at byte offset 0 (the beginning of the file).

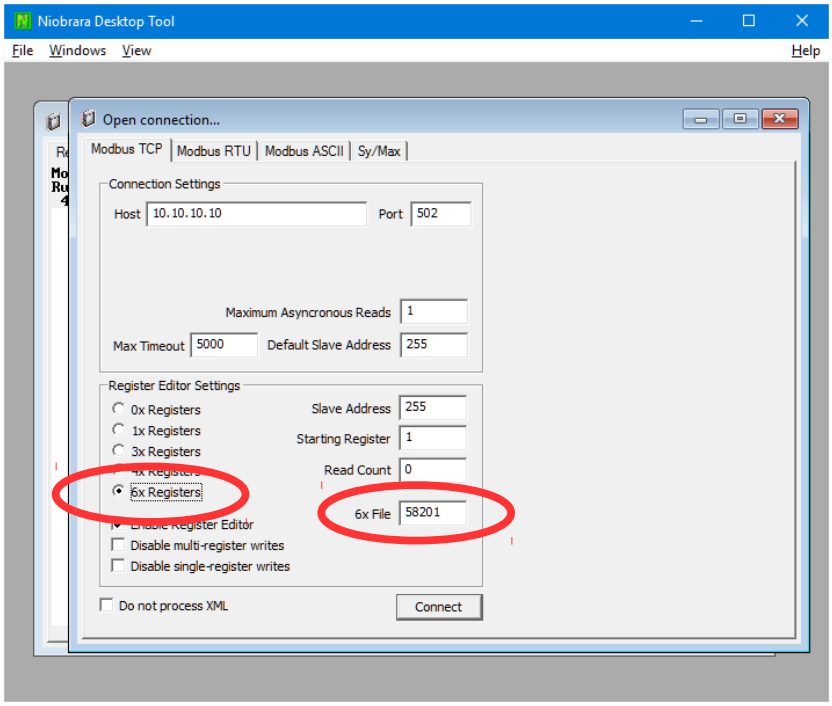
Note: File 58201 is used because this is the file specified by the DTM. Looking at the DTM Utility, we see file 58201 listed in the Output assembly.



Before we look at the 6x file, change the starting register back to 200 to see the PLCOUT data:

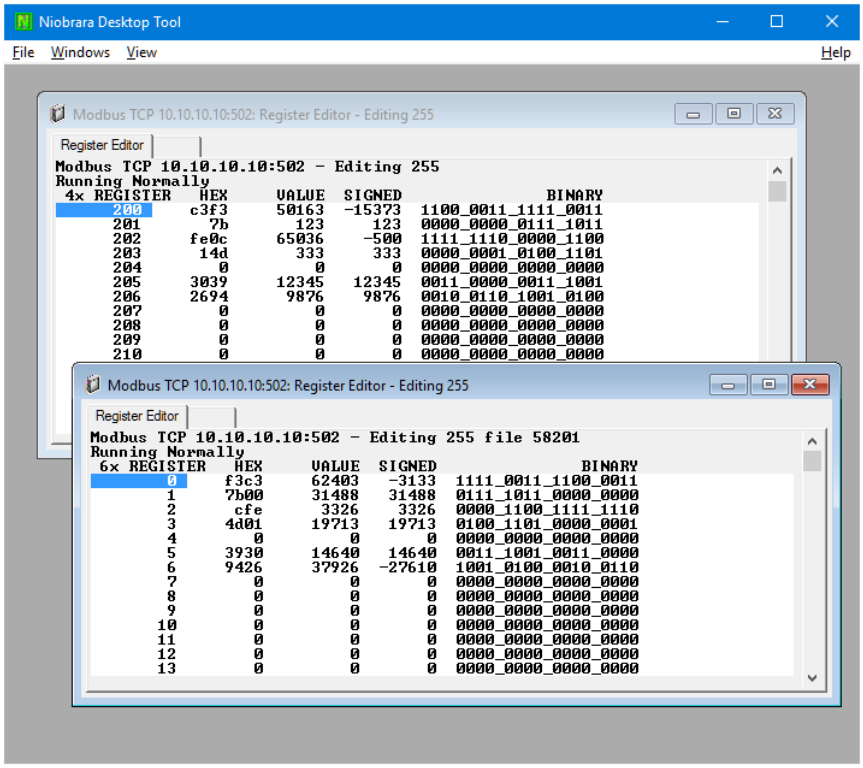


Now Open a new connection in NRDTOOL by selecting File >Open connection.



Select 6x Registers then enter 58201 in the 6x File and press “Connect”

Move the cursor up to register 0.

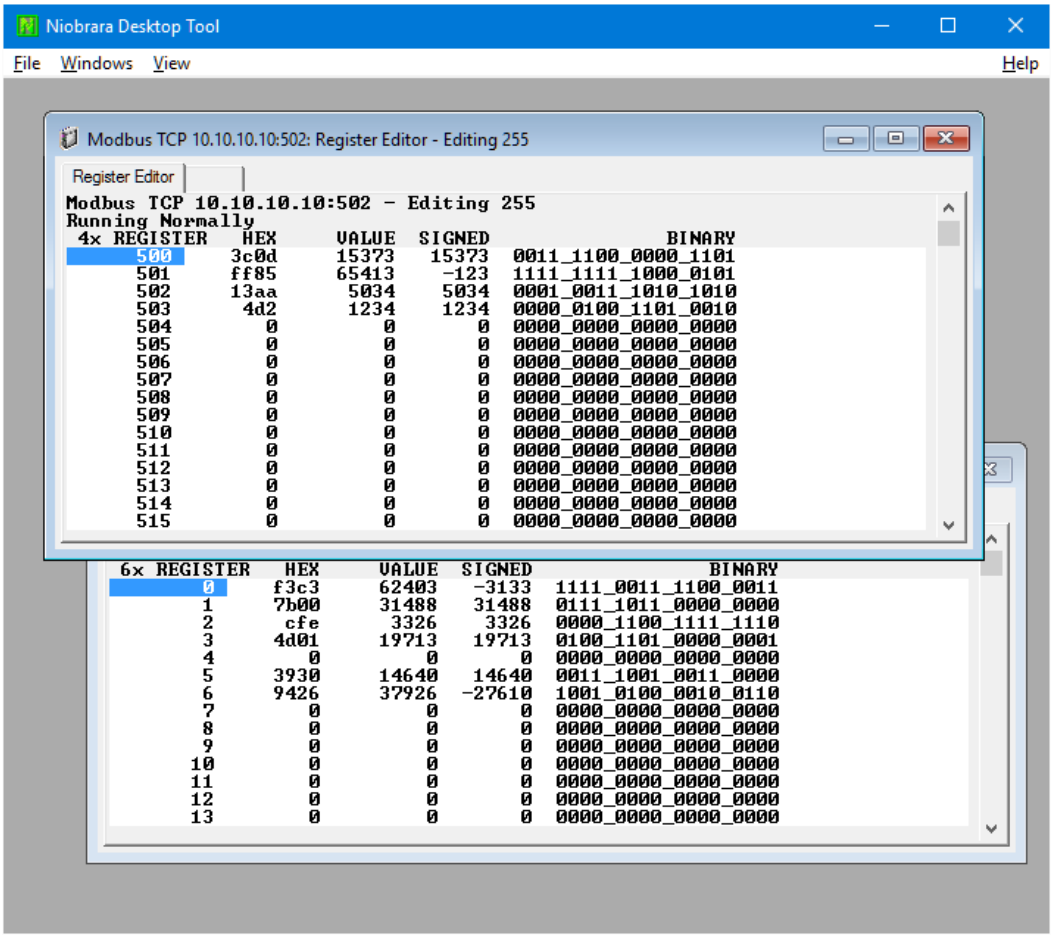


Look at the data in HEX and notice that the data is byte swapped. This is

because the data in the 6x file is straight from the Ethernet/IP data handed to the UCM by the PTK board. This explains the code in Thread 1 that swaps the byte order of the data as it copies to/from the OUTPUT[] registers.

We can look at the PLCINPUT data 6x file as well.

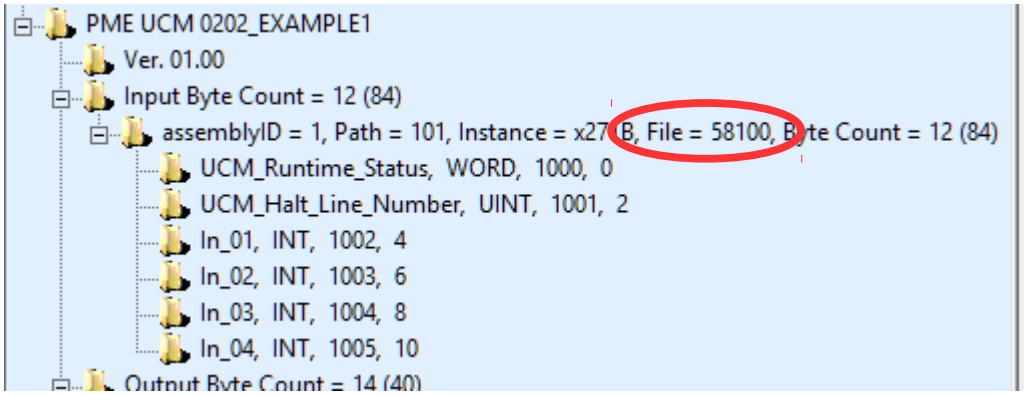
First change the start register from 200 back to 500.



Inspecting Thread 1 again shows the following line of code:

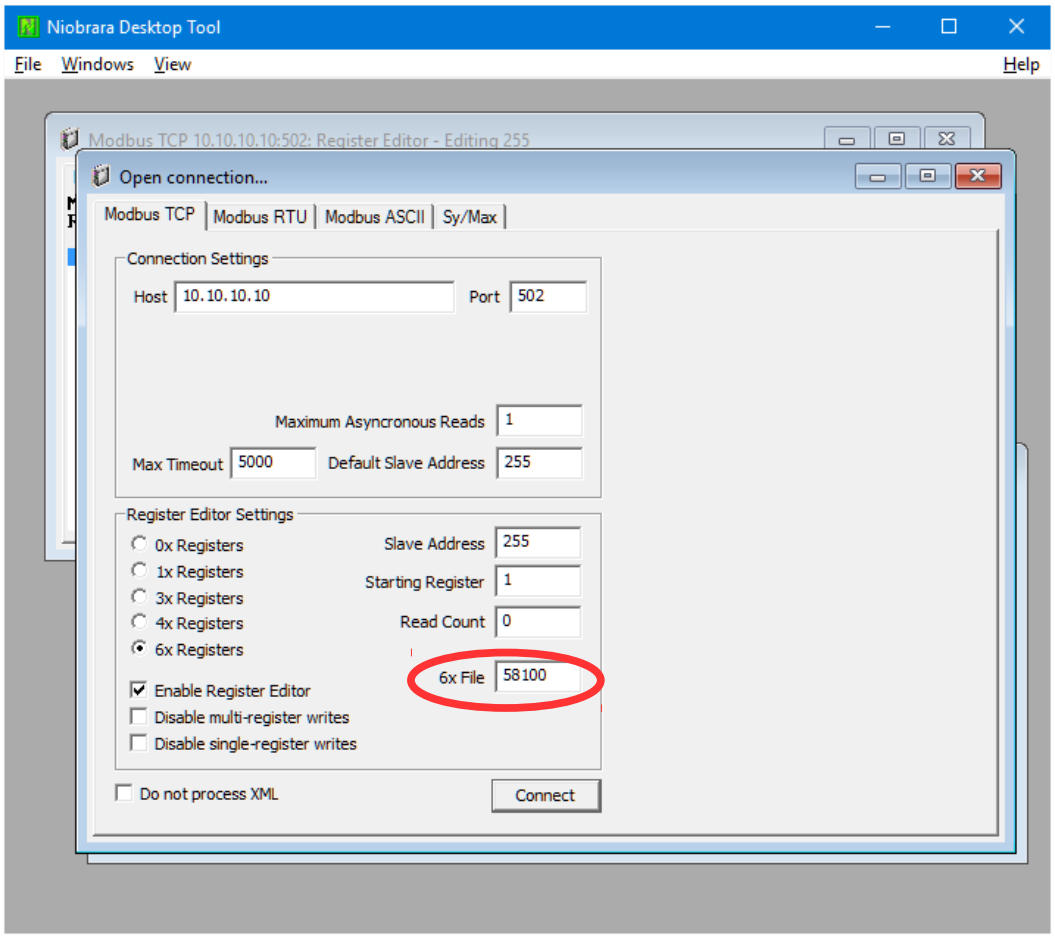
write file 58100 offset 4 BPInputs { Copy inputs to backplane block 0}

This procedure copies the data from the byte array BPInputs to 6x file 58100. Again looking back at the DTM shows why file 58100 is chosen.



Also notice that the first 4 bytes of this file are the UCM Runtime Status and Halt Line Number. The UCM operating system automatically fills in these two values so the UCM application must start its data with an offset of 4 bytes as shown in the WRITE FILE line of code above.

Open a new connection in NRDTOOL with 6x file 58100.



Again notice that the hex values are byte swapped because of the Ethernet/IP data

structures. Also notice that the data starting in register 500 is in register 2 of the 6x file because of the 4 byte offset of the write file command.

Using the 6x file register viewer can be very handy when byte aligning the data in a UCM application.

NOTE: Accessing the 6x files associated with the DIO interface has a big impact on the performance of the backplane. These registers should only be accessed during troubleshooting and not normal operation.