

# ETH3 user guide

Updated to v3.15+



By Ricardo Medina  
Mircom OpenBAS  
ETH3 user guide

# ETH3 user guide

## Contents

Companion documents for the ETH3.....	3
Training videos for the ETH3.....	3
Description and features. ....	4
Setup the initial IP configuration to make connection in a network.....	5
Upgrading the firmware to use the new features.....	12
Using the web page server and updating the internal flash web pages. ....	14
Enabling and disabling the SPI port. ....	16
Using a USB memory to host user customized web pages. ....	18
Unlimited data trending on the USB flash memory .....	21
Using the ETH3 as an Ethernet protocol converter BACnet/IP & Modbus-TCP .....	23
Know the internal database local to the ETH3.....	24
Access the expanded remote database of a companion NX controller via the high-speed SPI bus.....	30
Setting the ETH3's field buses as masters in BACnet, Modbus or Optomux.....	31
Accessing slaves for programming on the fieldbuses on any protocol using bridging and tunneling.....	32
Adding remote points both manually and using BACnet discovery.....	37
Using the PLC to create powerful logic to interoperate between slave devices.....	45
Use the schedules.....	51
Add the i2c eight-channel current metering front mounted expander.....	52
Add the i2c eight-channel binary input monitor front mounted expander.....	53
Add the i2c seven-channel Input/Output front mounted expander.....	54
Using the e-mail generator.....	55
Connecting a Mircom fire panel using the printer port to convert to BACnet and Modbus.....	63
Attach to Arduinos and Raspberry PI.....	69
Connect via Telnet and use SQL commands.....	73

---

## Companion documents for the ETH3.

Below is a list and the links for documents that will help you better use the ETH3.

- ❖ **LT-2204** OpenBAS-NWK-ETH3\_Installation\_Manual.
  - [http://www.rikmed.com/OpenBAS/Installation/LT-2204\\_OpenBAS-NWK-ETH3\\_Installation\\_Manual%20rev%200.pdf](http://www.rikmed.com/OpenBAS/Installation/LT-2204_OpenBAS-NWK-ETH3_Installation_Manual%20rev%200.pdf)
- ❖ **LT-6500** OpenBAS Programming Manual.
  - <http://www.rikmed.com/OpenBAS/Programming/LT-6500%20OpenBAS%20Programming%20Manual%20rev%200.pdf>
- ❖ **LT-6630** OpenBAS\_Driver\_Setup.
  - [http://www.rikmed.com/OpenBAS/Programming/LT-6630\\_OpenBAS\\_Driver\\_Setup%20rev%202%202018-12-19.pdf](http://www.rikmed.com/OpenBAS/Programming/LT-6630_OpenBAS_Driver_Setup%20rev%202%202018-12-19.pdf)
- ❖ **LT-6631** eZ HVAC and building automation App Wizard.
  - <http://www.rikmed.com/OpenBAS/Programming/LT-6631%20eZ%20HVAC%20and%20building%20automation%20App%20Wizard%20rev%203.pdf>
- ❖ Mapping Table NX controllers for: **BACnet, Modbus, Optomux and N2-Open.**
  - <http://www.rikmed.com/OpenBAS/Programming/Mapping%20Table%20NX%20controllers.pdf>
- ❖ ETH3 **CGI** (common gateway interface) user manual for creating your custom web pages.
  - [http://www.rikmed.com/OpenBAS/Programming/ETH3\\_CGI\\_userManual.pdf](http://www.rikmed.com/OpenBAS/Programming/ETH3_CGI_userManual.pdf)
- ❖ Using Arduinos and Raspberry PI with the ETH3 with **SQL.**
  - <http://www.rikmed.com/OpenBAS/Programming/OpenBAS%20Arduino%20Query%20protocol%20rev.C.pdf>

---

## Training videos for the ETH3.

Below is a list and the links for videos that will help you better use the ETH3.

- ❖ Use the ETH3 as an **IP gateway** for NX controllers.
  - <https://www.youtube.com/watch?v=rF98eV6k2RU>
- ❖ Set up the E-mail server with an **FX-3500** Mircom fire panel.
  - <https://www.youtube.com/watch?v=zAVriI0D5-Q&t=6s>

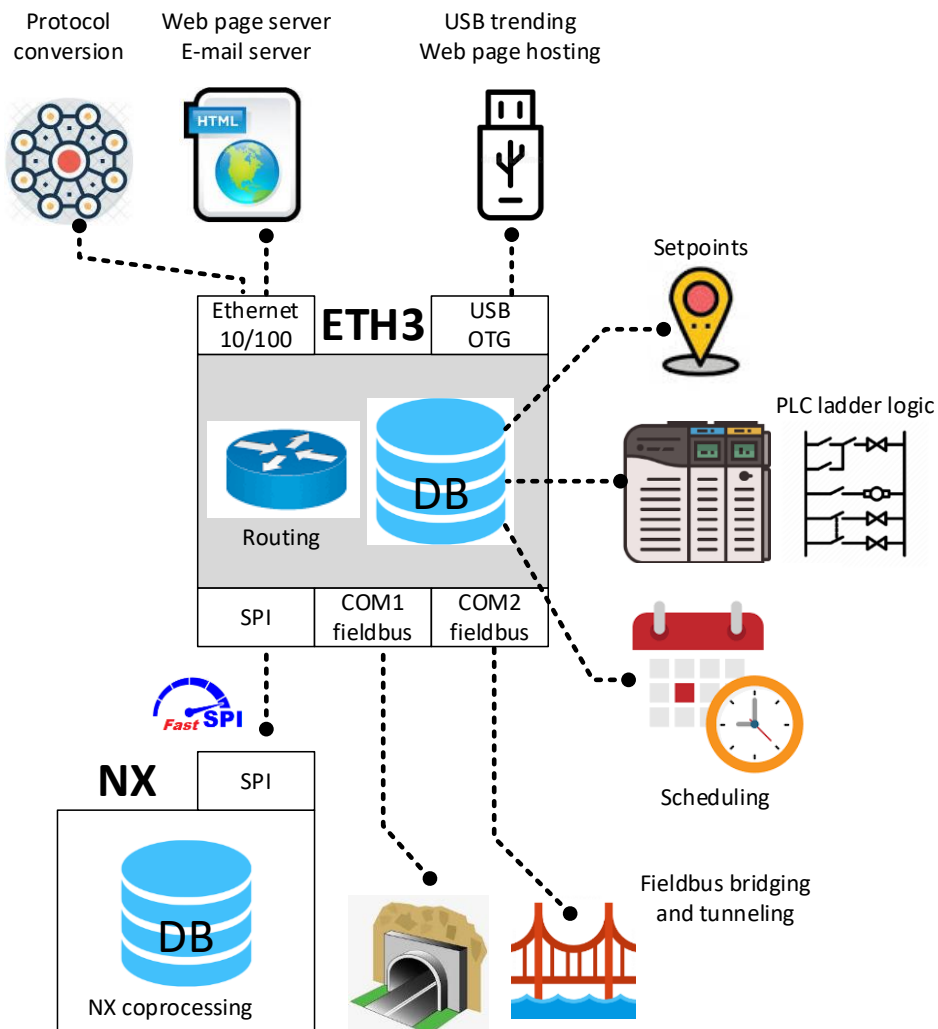
## Description and features.

The **OpenBAS-NWK-ETH3** initially started out only as an IP gateway to provide NX controllers IP connectivity via an Ethernet port that was achieved by attaching the two controllers over a high-speed SPI bus.

It has over time outgrown this initial design goal and it can now operate as a full standalone controller with many much more capabilities such as:

- Web page server with user configurable web pages loaded into an inserted USB memory.
- E-mail server to forward alarms or events.
- Multiprotocol server for Optomux-HTTP, Modbus-TCP, and BACnet-IP.
- Field bus slave and multi-master in Optomux, N2-Open, Modbus-RTU, BACnet/MSTP.
- SQL over a telnet connection.
- Dual PLC with 800 ladder instructions with full dual database access with variable scoping.
- Contains its own local database of setpoints, schedules, timers and working registers besides the original NX's database which is still accessible when an NX is attached over the SPI bus.
- Router functionality to bridge and tunnel configuration and programming commands over SPI and fieldbuses.
- Data logger functionality storing trends in USB memory.

The image below graphically depicts all these new additions.



## Setup the initial IP configuration to make connection in a network.

When the ETH3 comes out of the factory it has default IP settings that on most networks allow it to acquire an IP address automatically by a DHCP server. The Dynamic Host Configuration Protocol is a network management protocol used on Internet Protocol networks for automatically assigning IP addresses and other communication parameters to devices connected to the network using a client–server architecture.

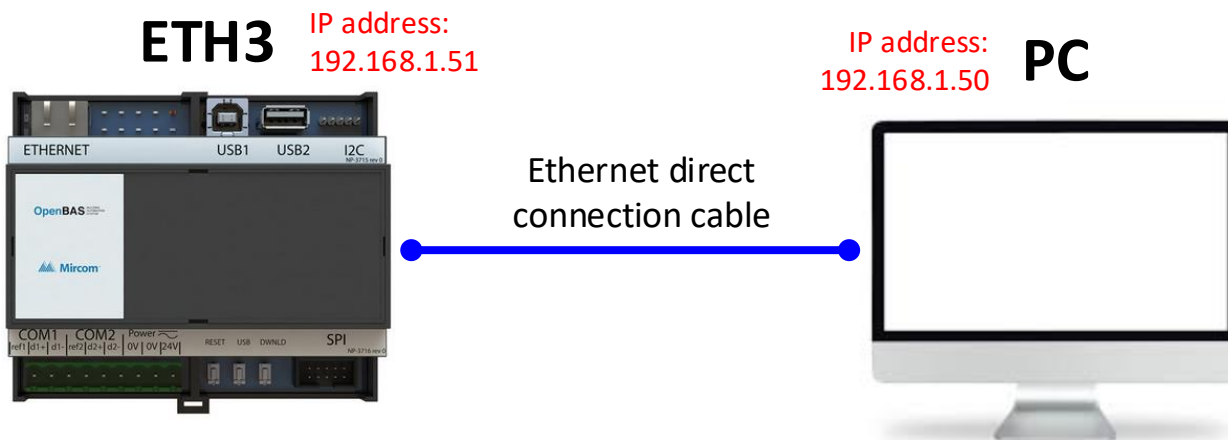
### Default IP settings for a factory new ETH3:

Hoste name	ETH3			
DHCP server	No			
DHCP client enabled	Yes			
Default IP address	192	168	1	51
Default mask	255	255	255	0
Default gateway	192	168	1	1
Primary DNS server	192	168	1	254
Secondary DNS server	0	0	0	0

### Set your PC's IP settings for direct connect:

The screenshot shows the 'General' tab of the Windows network settings window. It is configured for a static IP address. The IP address is set to 192.168.1.50, the subnet mask is 255.255.255.0, and the default gateway is . . . . The DNS settings are also configured for static addresses: Preferred DNS server is . . . and Alternate DNS server is . . . . The 'Obtain an IP address automatically' and 'Obtain DNS server address automatically' options are unselected. The 'Validate settings upon exit' checkbox is also unselected. An 'Advanced...' button is visible at the bottom right.

Therefore, to connect with a **factory new ETH3** if you use a direct cable connection between the **ETH3** and the **PC** you will be able to connect if the **IP settings of your PC's** Ethernet network adapter is set as shown above in the **right image**.



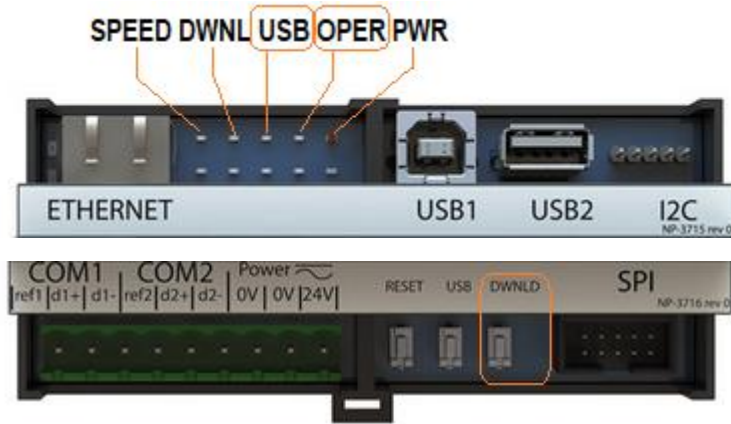
There are several ways you can view or modify the **IP address** the ETH3 has when it is connected into a network with a DHCP server and also change the settings manually when needed. Each method is described below:

- Reset to factory default IP settings using the **DOWNLOAD** button during power on
- View and modify the IP settings with a USB memory inserted into the ETH3.
- View and modify the settings using the web browser.
- View and modify the settings using the configuration software tool.
- View the current IP address on the LCD of an NX10L or NX10D controller attached via the SPI bus.

---

## Reset to factory default IP settings using the DOWNLOAD button during power on

When you need to reset the IP settings on an ETH3 to their factory reset value this is the detailed procedure that you can do on an ETH3 that has already been used and you simply want to have a starting point to connect to it with your PC over the network.

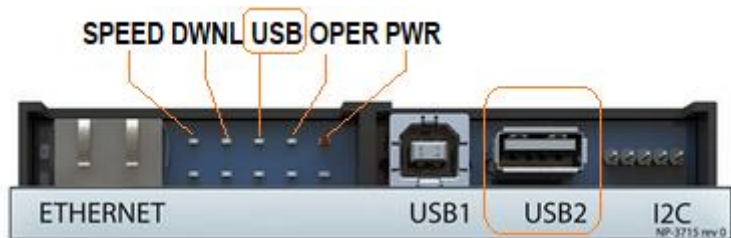


- ⊗ Reset the **ETH3** controller by either pressing for 1 second the **RESET** button or cycling power to the controller.
- ⊗ After any kind of reset, the two yellow LED's **USB** and **OPER** blink 10 times each second.
- ⊗ During this blink time press and keep holding the **DWNL** button.
- ⊗ After the 10 seconds start-up sequence has elapsed and still keep holding the **DWNL** button, the **USB+OPER** yellow LEDs will blink faster during 4 more seconds with a 200-millisecond blink rate.
- ⊗ Still keep holding down the **DWNL** button and finally after the 4 second period expires, both LED's will stay **steady-ON** for 3 seconds and reprogram the IP parameters in FLASH to their default factory values.
- ⊗ Now finally release the **DWNL** button.
- ⊗ The **ETH3** will now reset again and start with the new factory default IP values.

---

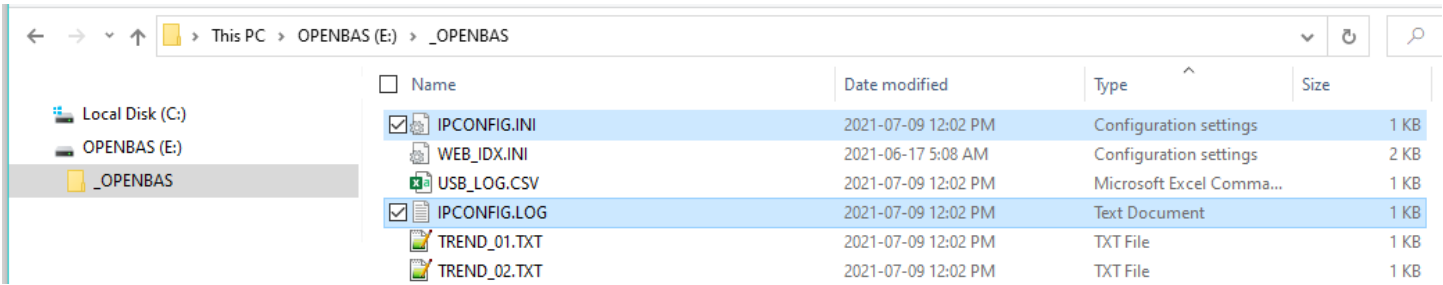
## View and modify the IP settings with a USB memory inserted into the ETH3.

From version **3.06.0** support for viewing and modifying the IP parameters via a USB memory inserted into **USB2** was added, the procedure is as described below.

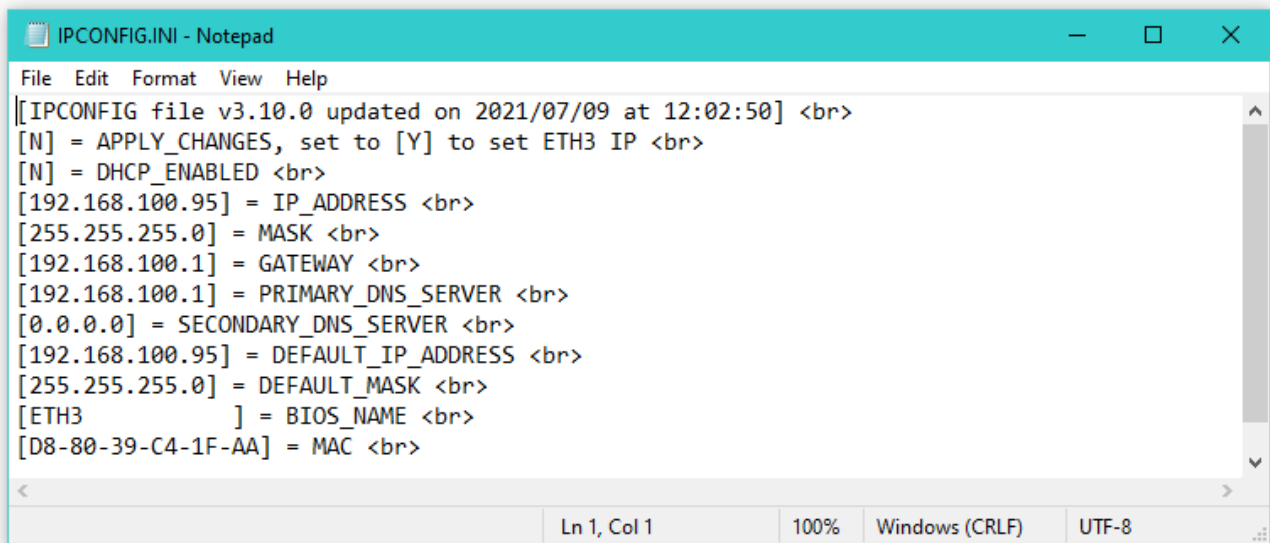


- ⊗ Insert an empty **USB memory** formatted as **FAT32** into the **USB2** connector, making sure nothing else is connected into the USB1 connector.
- ⊗ When the **ETH3** mounts the USB memory, the **USB** yellow LED will blink fast for 2 seconds, after it stops blinking you can safely remove the USB memory. **NOTE: never remove the USB memory while the USB yellow LED has activity and is blinking, as the file system might get corrupt and need to be reformatted.**

- ⊗ Insert the **USB memory** now into your PC and in the **OPENBAS** folder created on the root, and you will find two files for the IP settings:
  - **IPCONFIG.INI** Which contains the IP settings for viewing or modifying.
  - **IPCONFIG.LOG** A log file where any change to the IP settings via the USB memory is stored.



- ⊗ Open first the **IPCONFIG.INI** file using notepad or any text editor that reads and writes plain text files. If you want to do changes such as selecting a static instead of a dynamic address change the DHCP enabled from [Y] to [N], also write the new IP address or mask in their respective sections.
- ⊗ When all changes are ready also change the character in brackets [N] to [Y] in the line: [N] = APPLY\_CHANGES, set to [Y] to set ETH3 IP <br> line. **NOTE: Don't change the formatting on any line them such as brackets or spaces or swap them from position, just change the contents otherwise the ETH3's parser will report an error when reading back the IPCONFIG.INI file.**
- ⊗ Now save the file, safely remove the **USB memory** from your PC and re-insert it back into the ETH3's USB2 connector as you did in the first step of this sequence.



- ⊗ If changes to the IP settings were flagged in the **IPCONFIG.INI** file, then instead of the normal USB mount sequence you will see that after the USB memory is mounted the **ETH3** will do a reset cycle to apply the changes by blinking ten times the **USB+OPER** yellow LEDs and then resume normal operation.
- ⊗ You can now remove the USB memory from the ETH3 to verify that the changes took place correctly. Open the IPCONFIG.INI file and check that the [N] = APPLY\_CHANGES, set to [Y] to set ETH3 IP <br> line has the [N] at the beginning and that the new IP settings are correct.
- ⊗ Also, you can open the **IPCONFIG.LOG** file to double check there were no errors parsing the file and applying the changes as shown on the next page.

- ⊗ You can see that at 12:02:50 the **USB memory** was inserted with the **APPLIED\_CHANGES** flag set to **[Y]** and after parsing all the IP configuration lines in the file and detected no errors the **CHANGE\_IP\_SETTINGS** were applied and the **ETH3** was reset automatically to use the new IP settings.
- ⊗ Then at 12:30:23 the **ETH3** came out of reset and the new IP settings took effect and because the **IPCONFIG.INI** file had already been processed and marked with a **[N]** in the **APPLIED\_CHANGES** flag, a normal start-up process took place and the **ETH3** became operational with the new IP address the user selected.

```

IPCONFIG.LOG - Notepad
File Edit Format View Help
0 = [IPCONFIG file v3.09.4 updated on 2021/ 6/17 at 05:08:24] <br>
1 = [N] = APPLY_CHANGES, set to [Y] to set ETH3 IP <br>
--NO CHANGE-- (1) <br><br>
0 = [IPCONFIG file v3.10.0 updated on 2021/07/09 at 12:02:50] <br>
1 = [Y] = APPLY_CHANGES, set to [Y] to set ETH3 IP <br>
2 = [N] = DHCP_ENABLED <br>
   DHCP_ENABLED = 0 <br>
3 = 192.168.100.95
   IP_ADDRESS = 192.168.100.95 <br>
4 = 255.255.255.0
   MASK = 255.255.255.0 <br>
5 = 192.168.100.1
   GATEWAY = 192.168.100.1 <br>
6 = 192.168.100.254
   PRIMARY_DNS_SERVER = 192.168.100.254 <br>
7 = 0.0.0.0
   SECONDARY_DNS_SERVER = 0.0.0.0 <br>
8 = 192.168.100.95
9 = 255.255.255.0
10 = ETH3
     BIOS_NAME = ETH3 <br>
11 = D8-80-39-C4-1F-AA
     MAC <br>
--CHANGE IP SETTINGS-- (12) <br><br>
0 = [IPCONFIG file v3.10.0 updated on 2021/07/09 at 12:30:23] <br>
1 = [N] = CHANGES_APPLIED, set to [Y] to set ETH3 IP <br>
--NO CHANGE-- (1) <br><br>
Ln 1, Col 1    100%    Windows (CRLF)    UTF-8

```

It is always good practice to check with **PING** in the **command line** that the network connection is working as expected.

```

cmd.exe
C:\>ping 192.168.100.95

Pinging 192.168.100.95 with 32 bytes of data:
Reply from 192.168.100.95: bytes=32 time=1ms TTL=100
Reply from 192.168.100.95: bytes=32 time=1ms TTL=100
Reply from 192.168.100.95: bytes=32 time=1ms TTL=100
Reply from 192.168.100.95: bytes=32 time<1ms TTL=100

Ping statistics for 192.168.100.95:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>

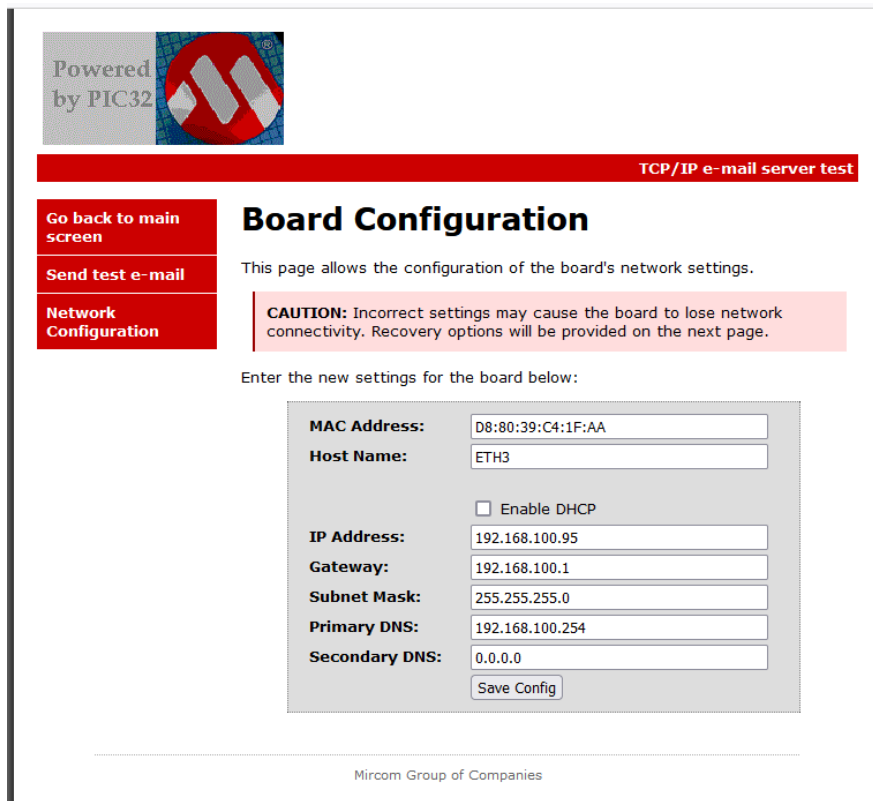
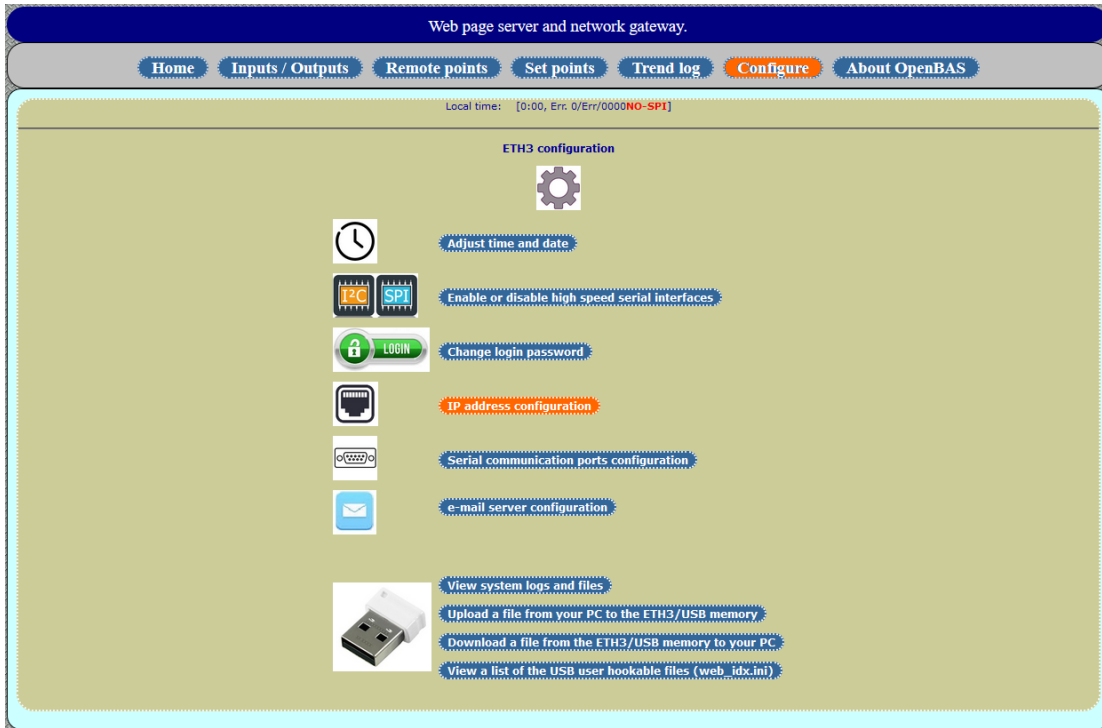
```



## View and modify the IP settings using the web browser.

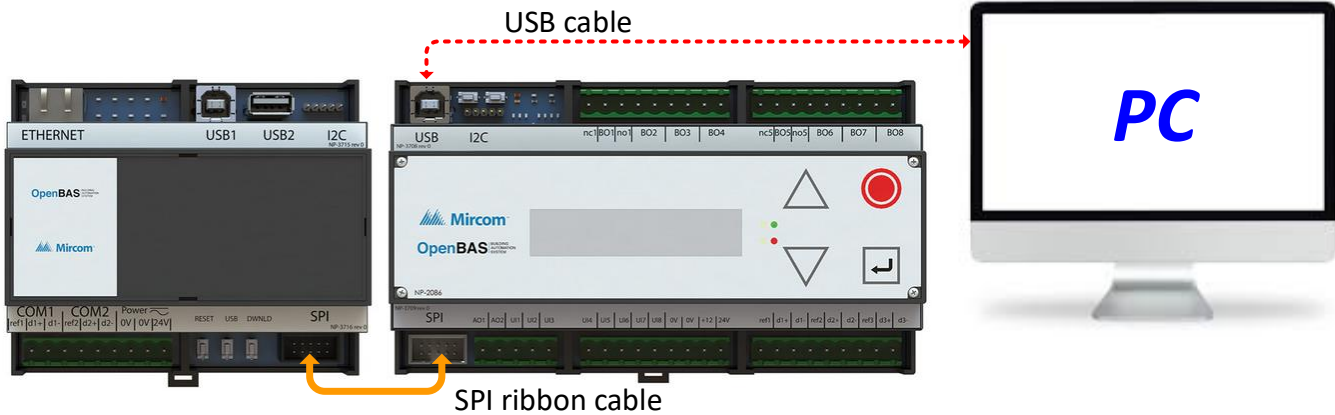
Once you know the IP address of the controller and have checked with the **PING** command line tool that the network connection is working, you can log in to the web page server using any standard web browser.

The default credentials for connecting are **USERNAME: admin** **PASSWORD: mircom** this can be changed from the Configure menu as well. To change the IP settings, select the IP address configuration link as shown below.



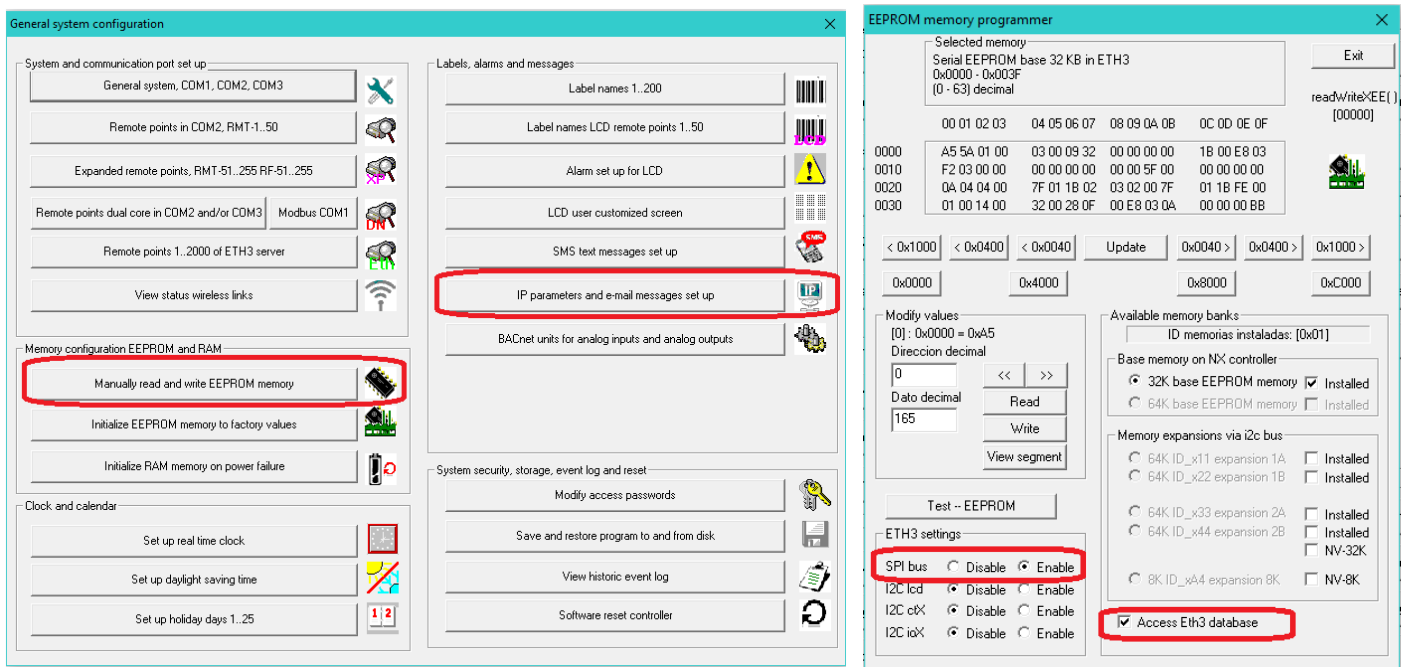
## View and modify the IP settings using the configuration software tool.

When an NX10 controller is connected to the ETH3 using the SPI connector then changes can be made to the IP configuration from within the configurator software. The PC connects to the USB port of the NX10 controller.



For this to work the **SPI port** has to be enabled on the **ETH3**, this can be done from two different points:

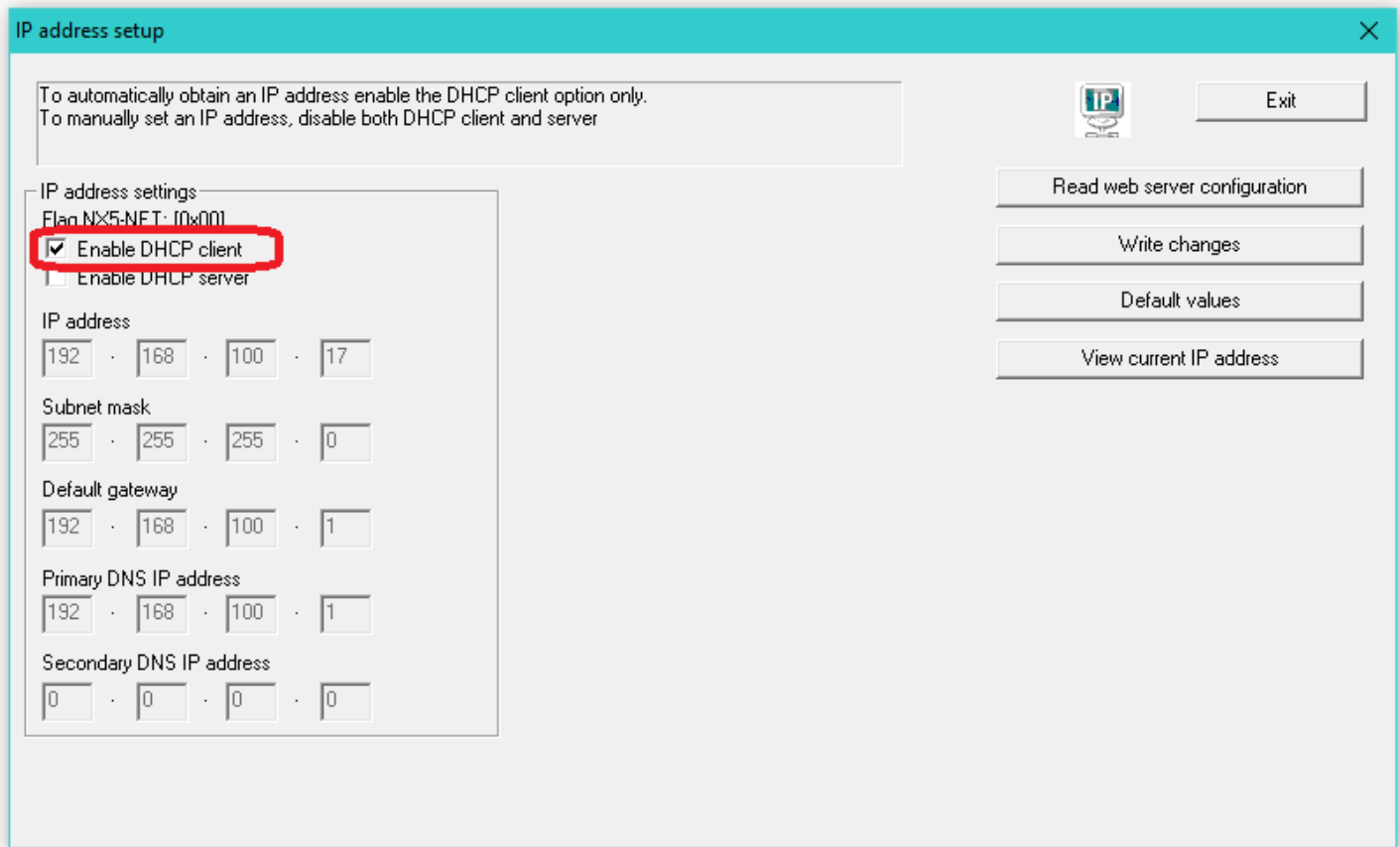
- From within the web browser in the **configuration** menu in the **Enable or disable high speed serial ports** (see the image on the previous page for viewing and modifying the IP settings using the web browser).
- From the configurator software in the **General system configuration** in the **Manually read and write EEPROM memory** highlighted below.



Make sure the ETH3 database checkbox is enabled (**connection should be over IP**) and make sure the SPI BUS is enabled as shown on the right side of the image above.

When the SPI bus is active you can now select the **IP parameters and e-mail setup** button while the software is connected via the USB cable to the NX10 controller attached to the SPI bus of the ETH3 as shown on the image on the next page.

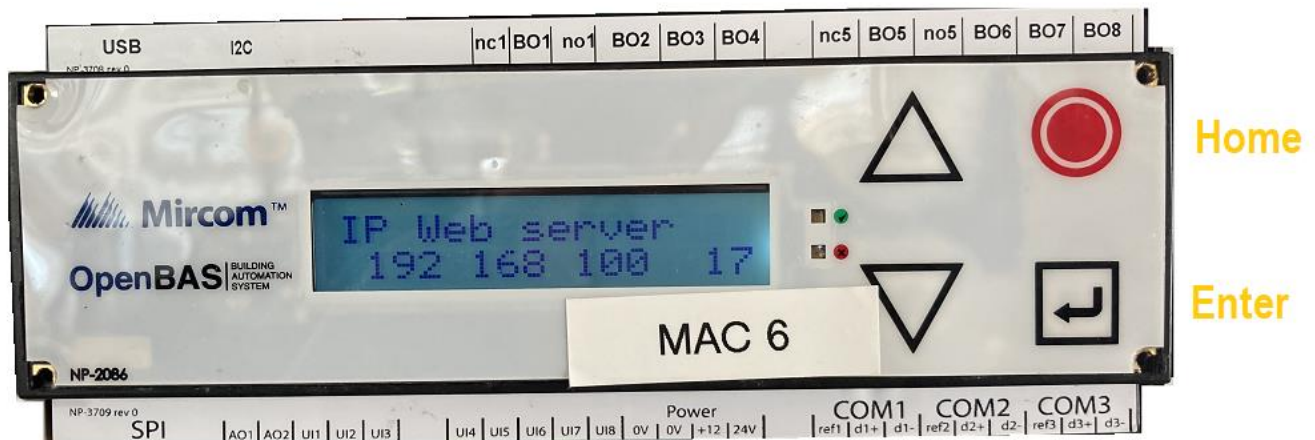
Now select the **IP parameters and e-mail setup** button and modify the IP as needed. Note that the IP fields are only enabled when the **Enable DHCP client** checkbox is unchecked.



### View the current IP address on the LCD of an NX10L or NX10D controller attached via the SPI bus.

The **NX10L** and **NX10D** controllers have an LCD display that when connected to the ETH3 thru the SPI bus show the current IP address that the ETH3 uses to connect to the network.

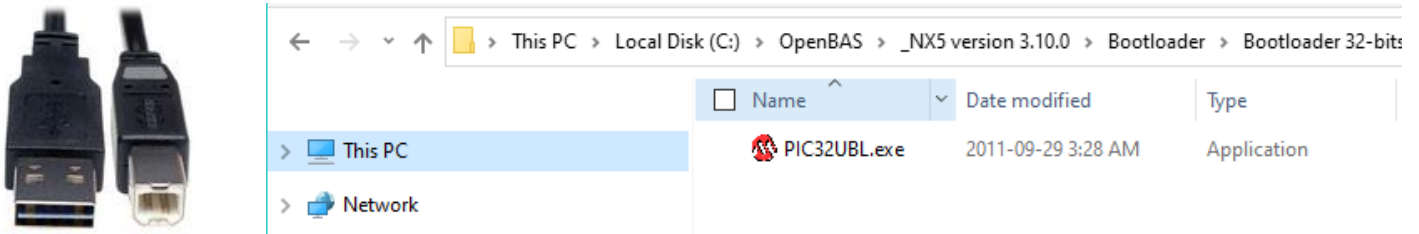
Press the **HOME** button to show the controller's current time (you might need to push it twice if a user custom screen is enabled), then press **ENTER** to see the current IP address of the ETH3 that is attached to its SPI bus.



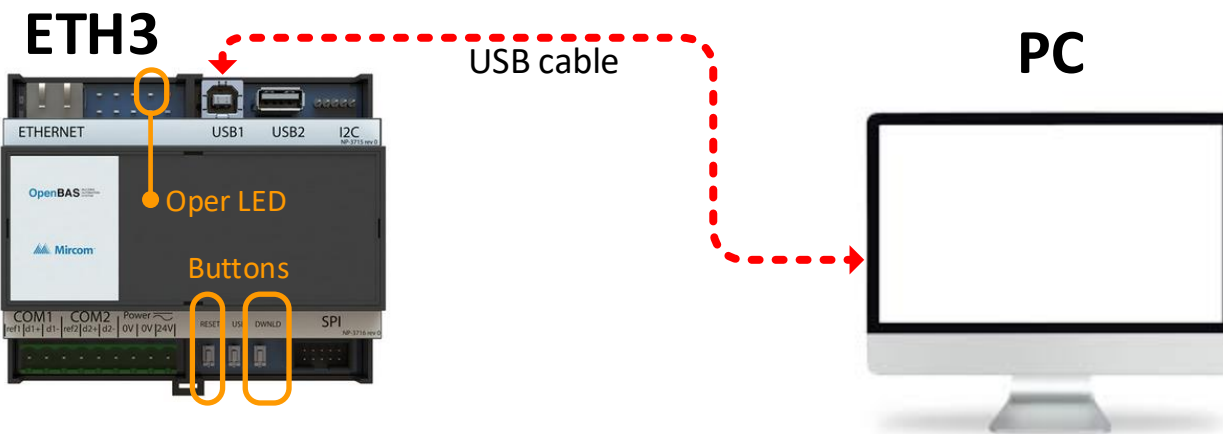
## Upgrading the firmware to use the new features.

When manufactured the ETH3 is loaded with the latest firmware available at that time. However, over time as new features become available and when the ETH3 is put in service on the field a firmware upgrade allows these new features to be used. All what is needed to upgrade the ETH3's firmware is:

A USB A-MALE to B-MALE cable and the **32-bit bootloader** that comes with the installation of System Design Studio.

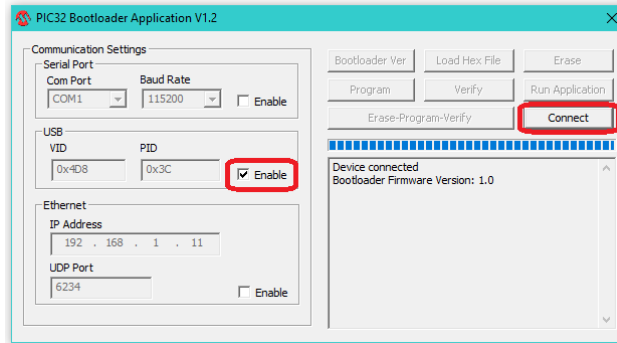


- ⊗ The first thing to do is to connect the **ETH3** and the **PC** using the USB cable, attach to any available USB port of your PC and to USB1 on the ETH3, make sure nothing is connected to the USB2 connector while upgrading the firmware.

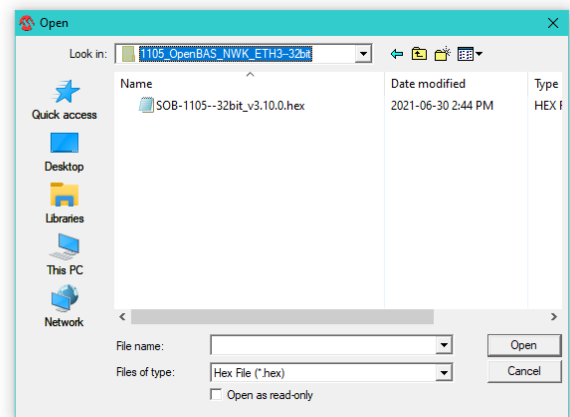
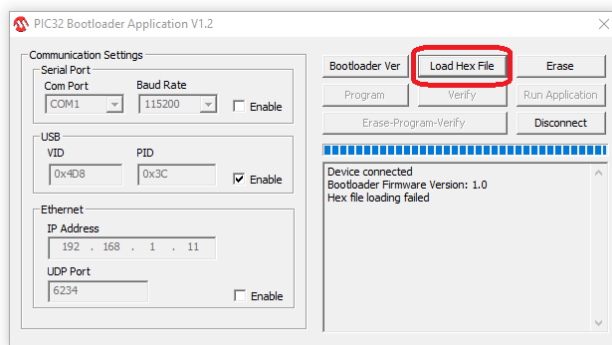


- ⊗ Press and hold the **DWNLD** button and at the same time press the **RESET** button for 1 second. This will place the ETH3 into **bootloader mode**, the **OPER** yellow LED will blink two times per second, all other LED's with the exception of POWER will be off.
- ⊗ Run the 32-bit Bootloader program **PIC32UBL.exe** located in the folder **C:\OpenBAS\\_NX5 version 3.10.0\Bootloader\Bootloader 32-bits** the version number might change as new software is released, always use the newest one available if more than one version is installed on your PC.

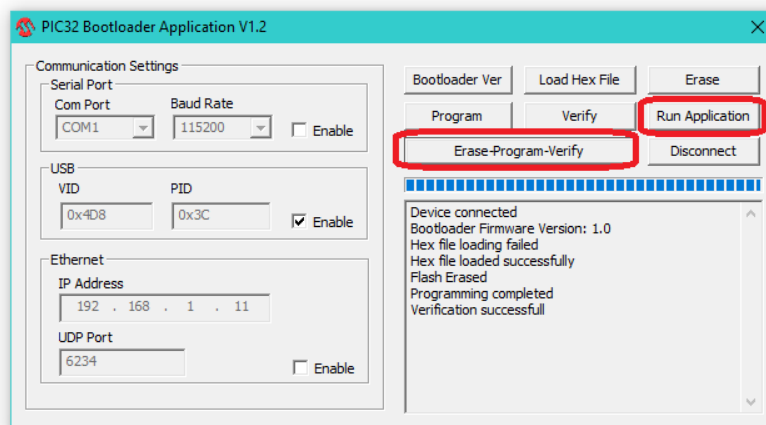
- ⊗ Select USB connection method by clicking the [X] Enable checkbox as shown in the image below and press the **CONNECT** button, if all goes well a message indicating the device is connected will show.



- ⊗ Click the **LOAD HEX** button and in the dialog search and select the **SOB-1105--32bit\_v3.10.0.hex** file located on the path: **C:\OpenBAS\NX5 version 3.10.0\Bootloader\\_HEXfiles\OpenBAS\1105\_OpenBAS\_NWK\_ETH3--32bit** again as mentioned before the version number might be different as new versions become available. Always use the newest version available.



- ⊗ To program the ETH3 press the **Erase-Program-Verify** button and make sure no errors are present and finally when the programming has ended and the “Verification successful” is displayed as shown below, press the **RUN APPLICATION** button to return the ETH3 to the **operational mode**.

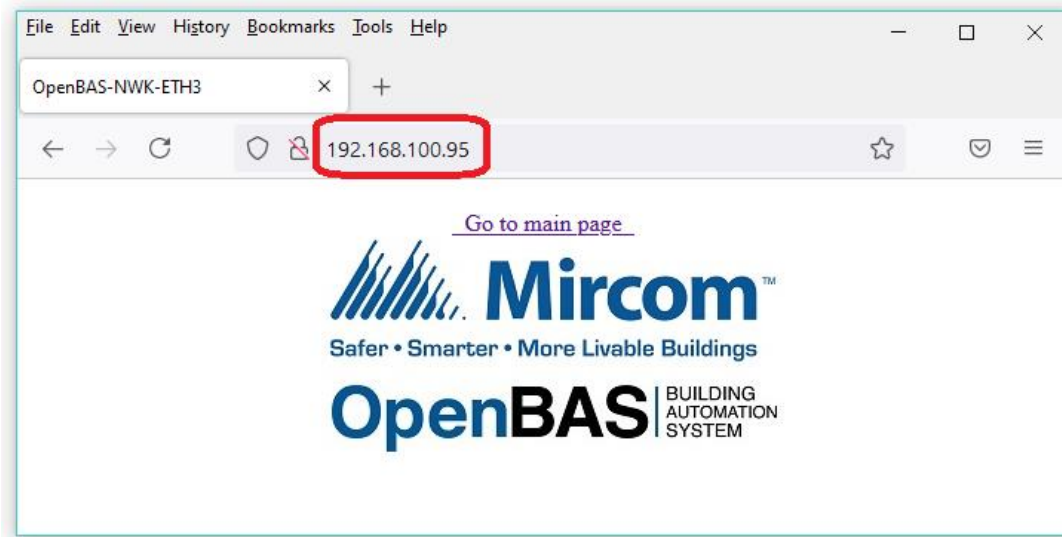


- ⊗ At this point the **ETH3** firmware is ready and only the internal web pages in the internal flash memory need to be updated as well to match the version number and enable any new feature that was added to the web pages. See the next section for instructions on how to update the web pages.

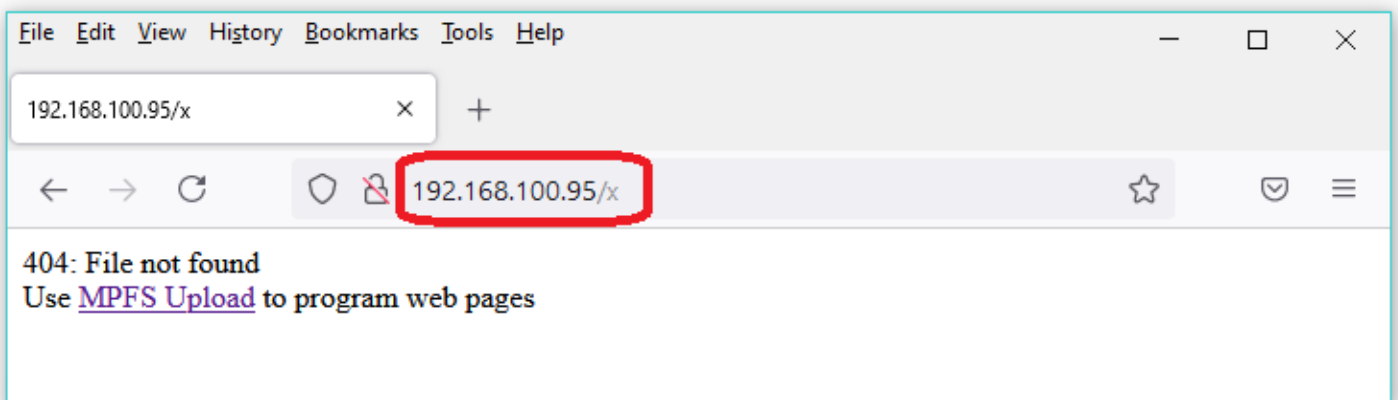
## Using the web page server and updating the internal flash web pages.

After upgrading the firmware, it is a good practice to also update the internal web pages stored in the ETH3's flash memory, to do this you should be able to connect with the via IP using any browser. See the sections for setting up the IP address in the previous sections if you still don't know the ETH3's IP address.

- ⊗ First try connecting to the IP to be updates by typing the IP address in the URL of your web browser.

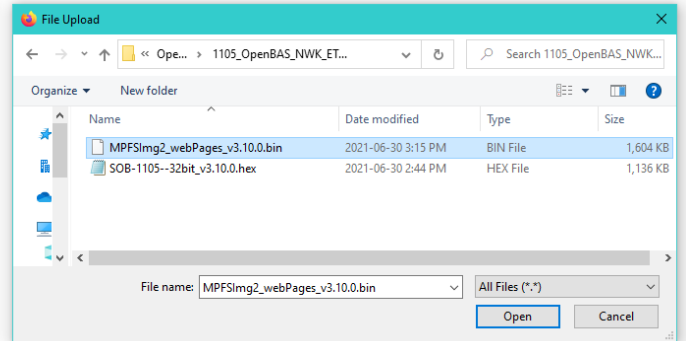
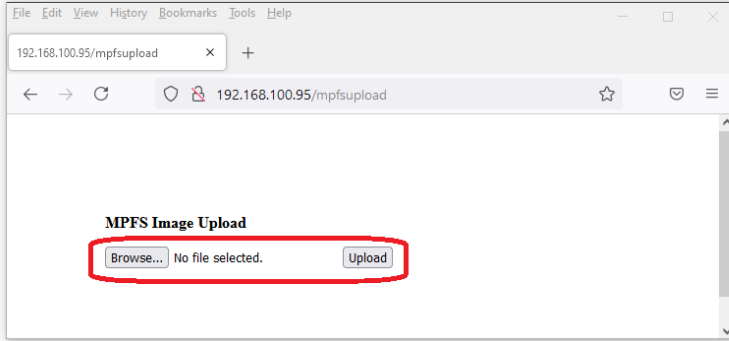


- ⊗ Now go again to the URL and add an 'X' to the IP address after a forward slash as shown below. When the message showing you that the file was not found, click on the **MPFS UPLOAD** link to select the image file located in the same folder you selected for the HEX file used to load the firmware.

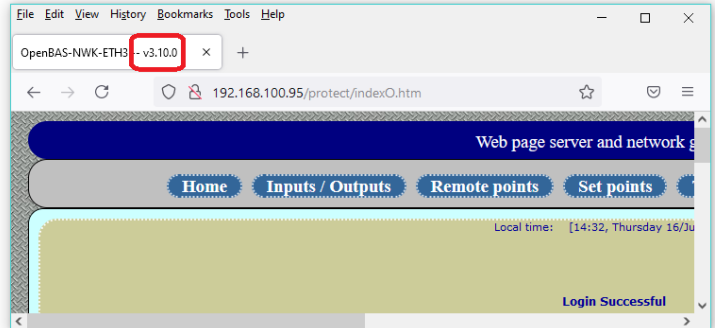
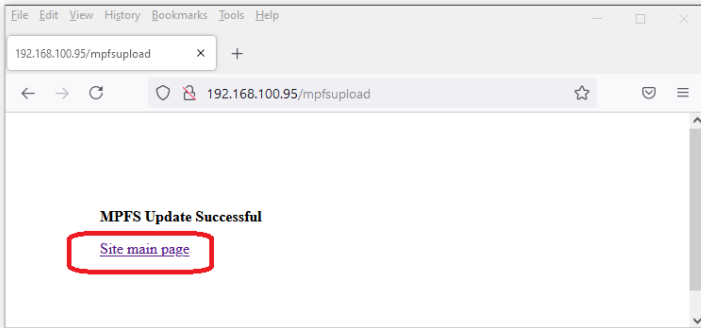


- ⊗ Use the BROWSE button as shown on the image on the next page and locate the flash image file: **MPFSImg2\_webPages\_v3.10.0.bin** is located in same folder where the firmware is located in the path: **C:\OpenBAS\NX5 version 3.10.0\Bootloader\HEXfiles\OpenBAS\1105\_OpenBAS\_NWK\_ETH3--32bit** again as mentioned before the version number might be different as new versions become available. Always use the newest version available.

- ⊗ After selecting the **BIN** image file then click the **UPLOAD** button and wait for about 1 minute for the new updated web pages to be transferred to the ETH3.



- ⊗ After the programming is complete you can follow the link See Main Page, if needed you will be asked for login credentials. The default credentials for connecting are **USERNAME: admin** **PASSWORD: mircom**



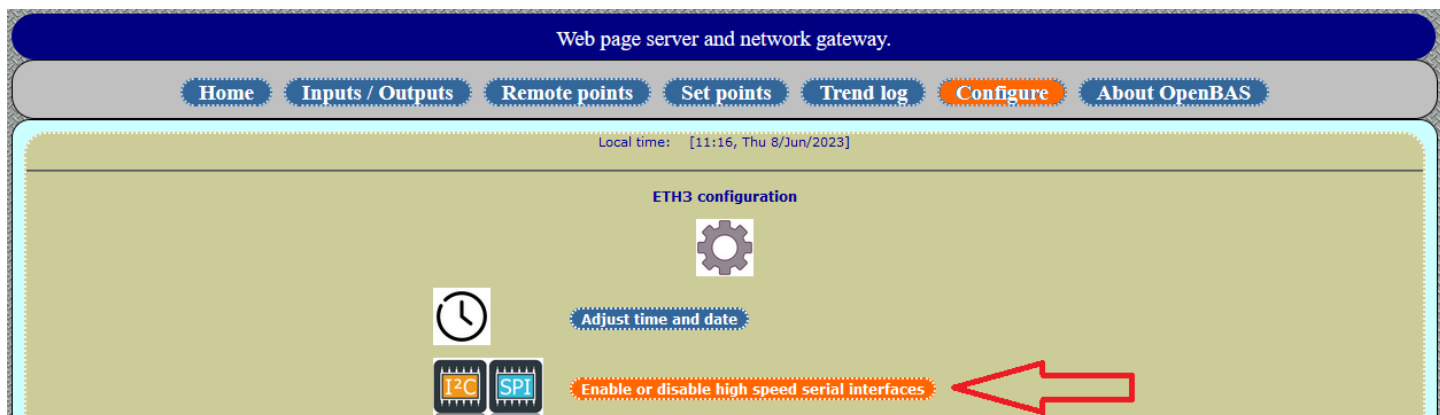
- ⊗ When you finally connect, and the web server shows up you will be able to verify that the new version is correct in the tab of the **HOME** web page as highlighted above.

## Enabling and disabling the SPI port.

The SPI port of the NX should only be enabled when an NX controller is physically attached to it using the flat 10 pin connector, otherwise erratic communication will stall the configuration as the SPI port will be looking for it.

Therefore, if it is not installed the SPI port should be disabled (default after factory configuration from version 3.14.0 and onwards).

If the NX::SPI controller is installed the setting should be enabled. There are two ways to enable the SPI setting. The first being the web server in the configure tab..



From this page select the appropriate setting. Once enabled the ETH3 will search for the NX::SPI controller and after few seconds activate it.



The second method is using the configurator software in the EEPROM memory section as shown on the next page.



Enabling or disabling the SPI bus from within the configurator's EEPROM memory programmer dialog.

Make sure not to change other settings as this might affect the operation of the ETH3 gateway.

The screenshot shows the 'EEPROM memory programmer' window. At the top, it displays 'Selected memory: Serial EEPROM base 32 KB in ETH3, 0x0000 - 0x003F (0 - 63) decimal'. Below this is a memory map table:

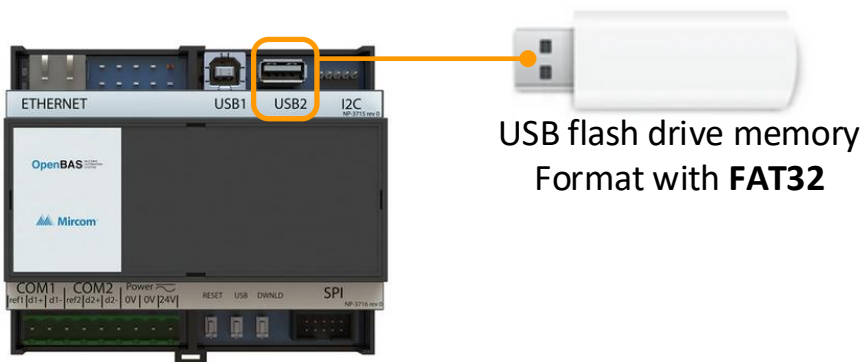
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	A5	5A	01	00	03	01	05	33	00	00	00	00	09	00	E8	03
0010	F1	03	FF	00	00	00	00	00	00	00	00	00	00	00	00	00
0020	0F	04	04	00	7F	01	1B	02	05	04	00	7F	01	1B	FE	00
0030	01	00	14	00	32	00	28	0F	00	E8	03	0A	00	00	00	1F

Navigation buttons include '< 0x1000', '< 0x0400', '< 0x0040', 'Update', '0x0040 >', '0x0400 >', '0x1000 >', '0x0000', '0x4000', '0x8000', and '0xC000'. The 'Modify values' section shows '[0]: 0x0000 = 0xA5', with a 'Decimal address' of 0 and 'Decimal data' of 165. The 'Available memory banks' section shows 'ID memorias instaladas: [0x01]' and 'Base memory on NX controller' with '32K base EEPROM memory' checked and 'Installed'. The 'Memory expansions via i2c bus' section lists several expansion options, all unchecked. The 'ETH3 settings' section, highlighted by a red arrow, has 'SPI bus' set to 'Enable', 'I2C lcd' set to 'Disable', 'I2C ctX' set to 'Disable', and 'I2C ioX' set to 'Disable'. Other settings include 'Test -- EEPROM' and 'Access Eth3 database' checked.

---

## Using a USB memory to host user customized web pages.

The web pages loaded in the previous section reside in the 2MB internal flash memory of the ETH3. But also, the user can create customized web pages and store them in a USB memory that can be installed in the USB2 terminal on the ETH3.



You can insert any USB flash memory, but it is preferred to store an empty or formatted USB memory because loading one that has a great number of files and subdirectories could stall the ETH3's file system.

Make sure the format used is the **FAT32** that is the file system supported by the ETH3.

This USB memory can store unlimited pages however the ETH3 can link only 50 of them, the full instructions on how to use this web pages and the CGI (Common Gateway Interface) to create dynamic web pages that update with the values of the database and also can use Java Script and Ajax are discussed in the following document:

- ❖ ETH3 **CGI** (common gateway interface) user manual for creating your custom web pages.
  - [http://www.rikmed.com/OpenBAS/Programming/ETH3\\_CGI\\_userManual.pdf](http://www.rikmed.com/OpenBAS/Programming/ETH3_CGI_userManual.pdf)

As this document is very large it will not be included in this user guide.

When the USB flash memory is installed unlimited data trending of up to ten different variables from the database can be enabled. See next section for details on this feature.

On the next two pages the CGI syntax and mapping table are extracted from the above-mentioned document for your quick reference

## CGI tag syntax for the ETH3 web pages

To correctly create tags that the ETH3 can understand they must adhere to the following format

### %TYPE-INDEX-FLAGS%

Where the **'%** percent character at the beginning and the end of the tag, are the opening and closing delimiters.

If you want to add the percent character in your HTML file alone there are two ways to do it:

- Create an empty CGI tag with two percent characters **%%**
- Or embed an HTML character code **&#37;**; where the 37 corresponds to ASCII code for the **'%** character, this way any character between 0...255 can be embedded into an HTML page.

Following is an ASCII character code and its codes that can be used in HTML:

SPECIAL CHARACTERS								
CHARACTER	NUMBER CODE	NAME CODE	CHARACTER	NUMBER CODE	NAME CODE	CHARACTER	NUMBER CODE	NAME CODE
"	&#34;	&quot;	ı	&#191;	&iquest;	à	&#224;	&agrave;
&	&#38;	&amp;	À	&#192;	&Agrave;	á	&#225;	&aacute;
<	&#60;	&lt;	Á	&#193;	&Aacute;	â	&#226;	&acirc;
>	&#62;	&gt;	Â	&#194;	&Acirc;	ã	&#227;	&atilde;
ı	&#161;	&iexcl;	Ã	&#195;	&Atilde;	ä	&#228;	&auml;
¢	&#162;	&cent;	Ä	&#196;	&Auml;	å	&#229;	&aring;
£	&#163;	&pound;	Å	&#197;	&Aring;	æ	&#230;	&aelig;
¤	&#164;	&curren;	Æ	&#198;	&AElig;	ç	&#231;	&ccedil;
¥	&#165;	&yen;	Ç	&#199;	&Ccedil;	è	&#232;	&egrave;
	&#166;	&brvbar;	È	&#200;	&Egrave;	é	&#233;	&eacute;
§	&#167;	&sect;	É	&#201;	&Eacute;	ê	&#234;	&ecirc;
¨	&#168;	&uml;	Ê	&#202;	&Ecirc;	ë	&#235;	&euuml;
©	&#169;	&copy;	Ë	&#203;	&Euml;	ì	&#236;	&igrave;
ª	&#170;	&ordf;	Ì	&#204;	&Igrave;	í	&#237;	&iacute;
«	&#171;	&laquo;	Í	&#205;	&Iacute;	î	&#238;	&icirc;
¬	&#172;	&not;	Î	&#206;	&Icirc;	ı	&#239;	&iuml;
®	&#174;	&reg;	Ï	&#207;	&Iuml;	ð	&#240;	&eth;
¯	&#175;	&macr;	Ð	&#208;	&ETH;	ñ	&#241;	&ntilde;
°	&#176;	&deg;	Ñ	&#209;	&Ntilde;	ò	&#242;	&ograve;
±	&#177;	&plusmn;	Ò	&#210;	&Ograve;	ó	&#243;	&oacute;
²	&#178;	&sup2;	Ó	&#211;	&Oacute;	ô	&#244;	&ocirc;
³	&#179;	&sup3;	Ô	&#212;	&Ocirc;	õ	&#245;	&otilde;
´	&#180;	&acute;	Õ	&#213;	&Otilde;	ö	&#246;	&ouml;
µ	&#181;	&micro;	Ö	&#214;	&Ouml;	÷	&#247;	&divide;
¶	&#182;	&para;	×	&#215;	&times;	ø	&#248;	&oslash;
·	&#183;	&middot;	Ø	&#216;	&Oslash;	ù	&#249;	&ugrave;
¸	&#184;	&cedil;	Ù	&#217;	&Ugrave;	ú	&#250;	&uacute;
¹	&#185;	&sup1;	Ú	&#218;	&Uacute;	û	&#251;	&ucirc;
º	&#186;	&ordm;	Û	&#219;	&Ucirc;	ü	&#252;	&uuml;
»	&#187;	&raquo;	Ü	&#220;	&Uuml;	ý	&#253;	&yacute;
¼	&#188;	&frac14;	Ý	&#221;	&Yacute;	þ	&#254;	&thorn;
½	&#189;	&frac12;	Þ	&#222;	&THORN;	ÿ	&#255;	&yuml;
¾	&#190;	&frac34;	ß	&#223;	&szlig;			

Following the opening **'%** character is the type of the variable to be dynamically converted by the **CGI**, on the following pages the codes for each object in the database are described:

Table of **TYPE** on the database highlighted in yellow, it must be at two-character code telling the CGI which variable type is being addressed:

	Variable type	TAG	Low range	High range	Flag name	Flag value defaults to 3 decimals	Flag value 0, 1, 2, 3 decimals	Flag status override active	Flag description	Flag ON/OFF	Flag Apagado/ Encendido	Flag Abierto/ Cerrado	Flag Normal/ Alarma	Flag Graphing	Flag Scheduling	Flag System clock
<b>HARDWARE (NX)</b>	Analog inputs	AI	1	40	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Binary inputs	BI	1	40	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Analog outputs	AO	1	10	fN	fV	ff0.. ff3	fOVS	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Binary outputs	BO	1	40	fN	fV	ff0.. ff3	fOVS	fD	fbON	fbAE	fbAC	fbNA	-	-	-
<b>SOFTWARE (NX)</b>	Lighting groups	BO	41	60	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Floats 32 bits EEPROM	DF	1	100	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Words 16 bits EEPROM	DI	1	100	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Bytes 8 bits EEPROM	DB	1	100	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	System timers	TM	1	16	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	RES_FLT result float register RAM 32 bits	RF	1	255	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	RES_BIT result bit RAM 1 bit	RB	1	255	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Remote points via field busses	RP	1	255	fN	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Schedules	HR	1	200	-	-	-	-	fD	-	-	-	-	-	see HR format	-
	Graph trend points	GR	1	16	-	-	-	-	fD	-	-	-	-	see GR ranges	-	-
	System real clock time	HR	1	-	-	-	-	-	-	-	-	-	-	-	-	fTM
<b>ETH3</b>	Remote points fieldbus ETH3 via COM1	RP	1	1000	-	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Remote points fieldbus ETH3 via COM2	RP	1001	2000	-	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Remote points IP	RP	2001	2255	-	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Set points 32 bits	DI	1001	1100	-	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Set points 16 bits	DB	1001	1100	-	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Set points 18 bits	HR	1001	1100	-	fV	ff0.. ff3	-	fD	fbON	fbAE	fbAC	fbNA	-	-	-
	Schedules	HR	1	200	-	-	-	-	fD	-	-	-	-	-	see HR format	-
	Hook to USB web files	WF	0	50	fN	-	-	-	fD	-	-	-	-	-	-	-

Following the **TYPE** and separated with a hyphen is the **INDEX** or object number, which must be between the ranges shown in the table above in the columns: **LOW RANGE** and **HIGH RANGE** inclusive.

Finally, after the **INDEX** and also separated by a hyphen, follows the **FLAG** field with the different flags shown in the table in all the sections labeled as **FLAGS**.

Finally, to close the CGI tag the % character must finish the tag, no spaces are allowed inside the CGI tags.

For examples on using the graph sample points refer to the source code of the web pages in the flash files stored in the directory:

**C:\OpenBAS\ETH3\_WebPagesUser\ETH3\_flashWebPages\protect\graf\_\*.cgi**

To print the real time clock use for example the CGI tag:

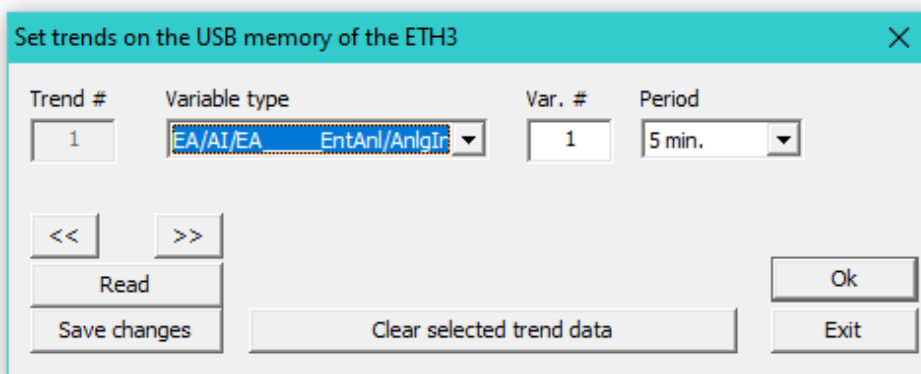
**%TM-1-fTM%**

For additional examples refer to the rest of the source file in the same directory and for user examples for the files in the user web pages located in:

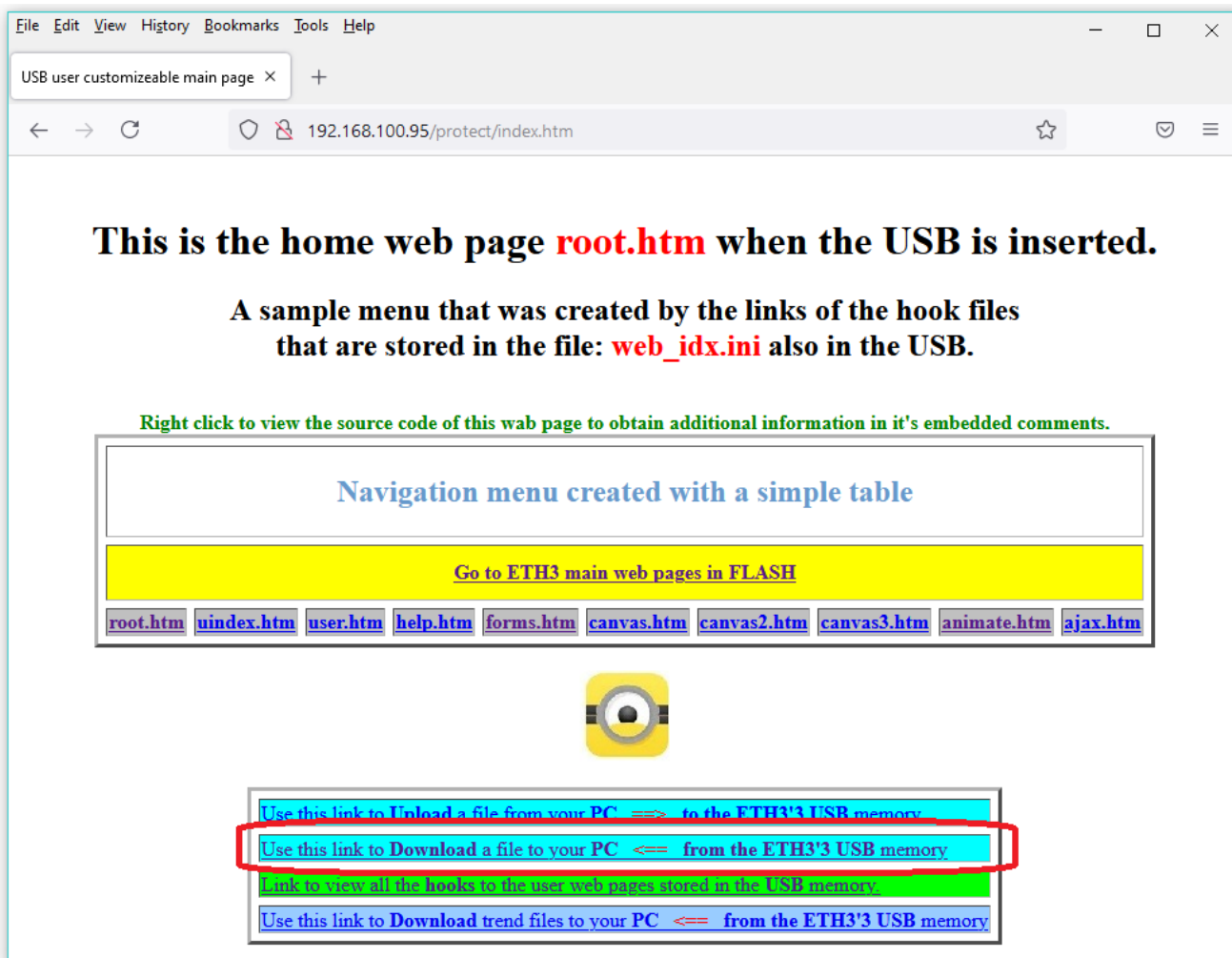
**C:\OpenBAS\ETH3\_WebPagesUser\\_OpenBAS\\*.\***

# Unlimited data trending on the USB flash memory

When the USB flash memory is installed unlimited data trending of up to ten different variables from the database can be enabled. IN the MFC tool select trends while the ETH3 is scoped, and the following dialog allows you to configure each one of the ten USB trends.

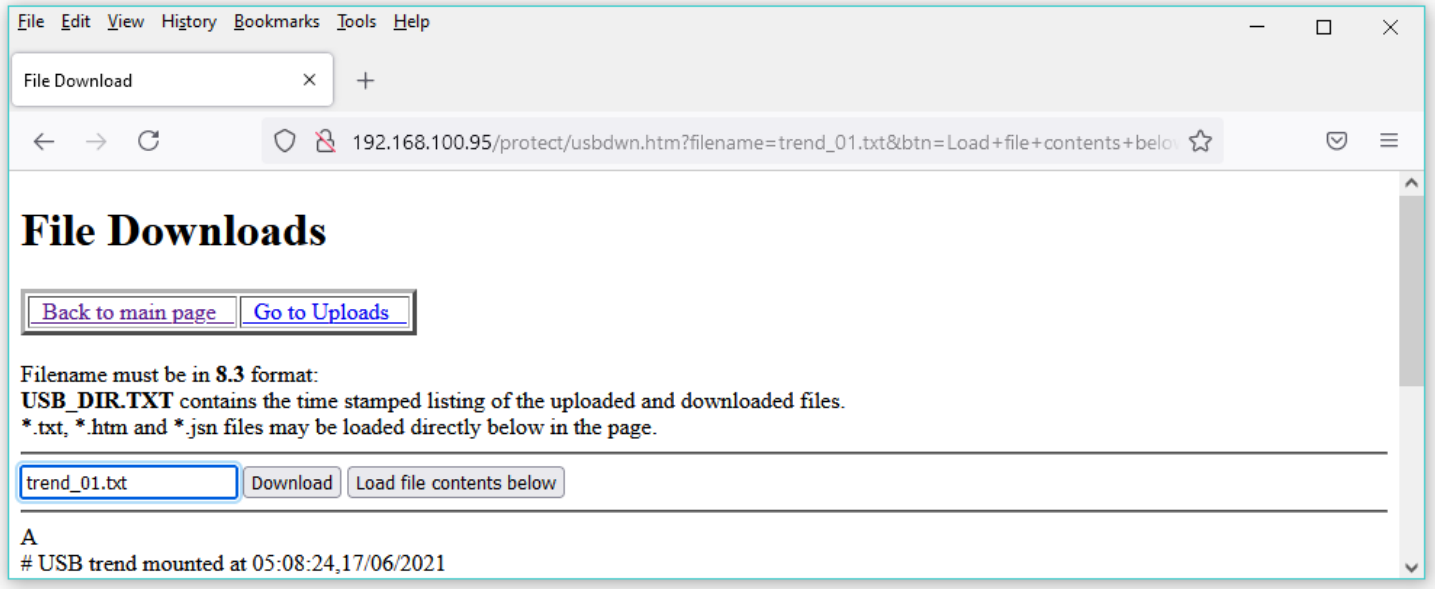


To download the data stored in the USB simply select the option shown below using a web browser.



There are ten files that the ETH3 updates with the period selected on the dialog shown on the previous page.

These ten files are named `trend_01.txt` thru `trend_10.txt` simply type the file name of the desired trend and press the DOWNLOAD button for the browser to download the file to your computer. If you just want to view the file contents in the browser select the LOAD CONTENTS BELOW button.



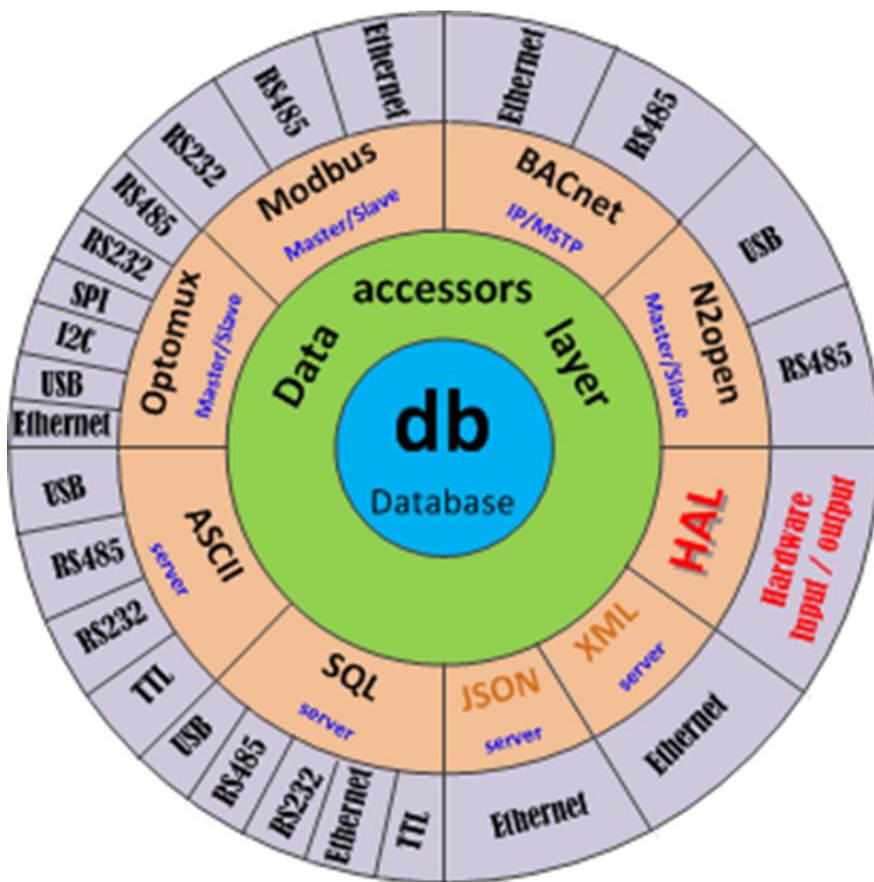
# Using the ETH3 as an Ethernet protocol converter BACnet/IP & Modbus-TCP

The ETH3's database, which is actually comprised of two databases is briefly explained below. On following sections of this user guide a detailed description of these two databases is provided.

- **Local ETH3 database**, is the database stored internally on the ETH3 including remote points managed by its own two field buses COM1 and COM2.
- **Remote SPI-NX database**, is the database of the NX controller connected over the high-speed SPI bus. This database includes any remote points that the NX has connected over any of its own three field buses COM1, COM2 and COM3 if it has a dual core installed.

Same as the NX controllers, the ETH3 has a layered database access, thus any object in the database is read or written using the **data accessor layer**. Thus, any object can be read or written over any protocol in any of the serial fieldbuses or via the IP protocols.

Even the internal logic of the PLC that the user creates accesses the database via this **data accessor layer**, so anything modified by the PLC is immediately available and propagated to any master protocol to its slaves without any user intervention.



- Database access is secured by a layer of data accessors
  - Protocols and the hardware abstraction layer read and write the database through the accessor layer
  - Communication ports go through the protocols layers to talk to accessors
- Hardware inputs and outputs talk to the database through data accessors as well

## Know the internal database local to the ETH3.

All of the variables that the dual database the ETH3 can access can be used as operands in the instructions of its PLC ladder logic, provided that scoping is used. See the table for this dual data base access below.

Data base object type	Type	ETH3 remote database			ETH3 local database				
		Scope	SPI::NX		Scope	ETH3 (scoped)		Mapped internally as:	Alternate:
Analog inputs	Hardware	S.AI	1	40	E.AI				
Binary inputs		S.BI	1	40	E.BI				
Analog outputs		S.AO	1	10	E.AO	1	40	AO-2001..2040	ADF-101-140
Binary outputs		S.BO	1	60	E.BO				
Set point 32-bits	EEPROM	S.ADF	1	100	E.ADF	1	100	ADF-1001..1100	
Set point 16-bits		S.ADI	1	100	E.ADI	1	100	ADI-1001..1100	
Set point 8-bits		S.ADB	1	100	E.ADB	1	100	ADB-1001..1100	
Result bit 1-bit	RAM	S.RES_BIT	1	255	E.RES_BIT	1	255	RES_BIT-1001..1255	
Result float 32-bits		S.RES_FLT	1	255	E.RES_FLT	1	255	RES_FLT-1001..1255	
Timers		S.TMR	1	16	E.TMR	1	16	TMR-1001..1016	
Remote points NX/NG		S.RMT	1	255					
Remote points E.COM1					E.COM1.RMT	1	1000	AO-1001..2000	AO-1..255
Remote points E.COM2					E.COM2.RMT	1	1000	RMT-1001..2000	RMT-1..255
Remote points E.IP					E.IP.RMT	1	255	RES_FLT-1501..1755	AI-1..255
Clock					E.CLK	1	7	ADI-1201..1207	ADI-201-207

## Scoping mechanism for selecting between the ETH3 and the SPI::NX databases

Since the ETH3 has two databases that PLC1+PLC2 on the ETH3 can access, how do we select what we want to access?

This is where scoping comes into play, to simplify things a design decision was made, where all PLC instructions on the ETH3 can only access the ETH3 local database, outlined on red in the MFC tool dialog on the next page which contain:

### Local database ETH3 (note the E. prefix explained below)

- All the E.ADF, E.ADI, E.ADB, E.RES\_BIT, E.RES\_FLT, E.TMR registers.
- Remote points from its E.COM1, E.COM2 and E.IP client.

And only the **OUTPUT ASSIGN** instruction that has been renamed as **TRANSFER INSTRUCTION** for this use, can read and write registers from and to the Remote SPI::NX database, shown with the blue dotted arrow on the next page.

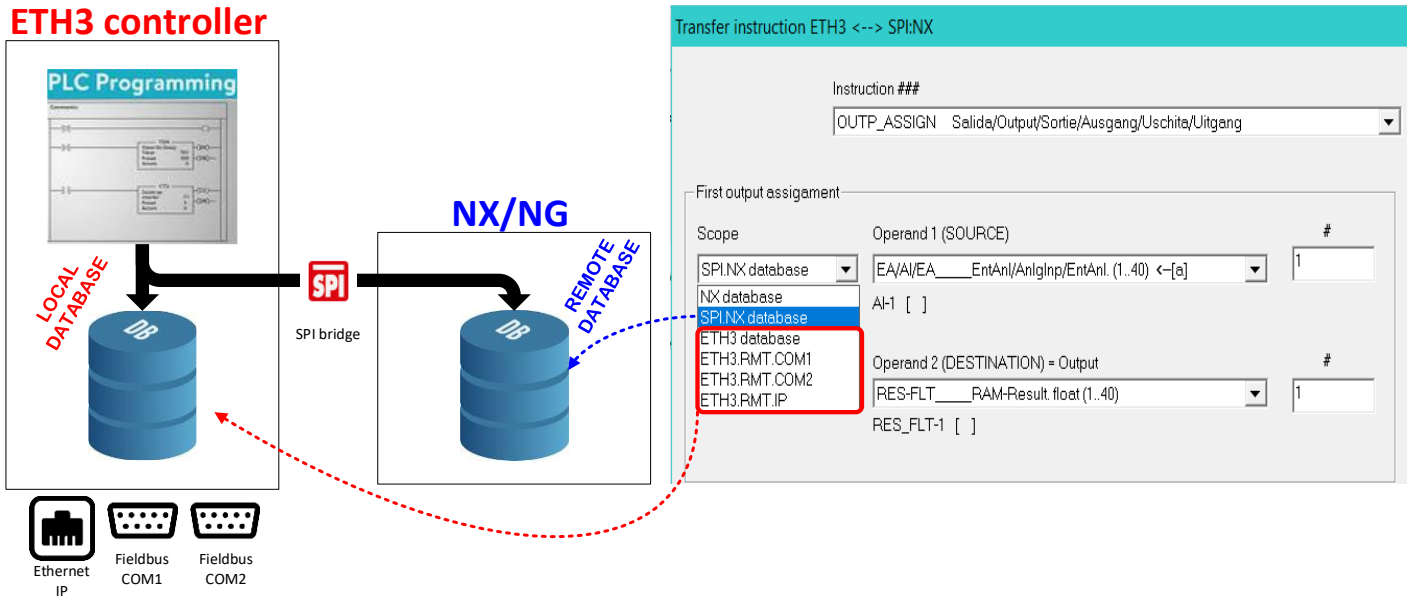
By convention the following syntax for accessing the database used on the scripts when support is added to the ETH3 will be using the following prefixes:

- Object\_#      **Un-scoped objects** will generate a compiler error.
- S.Object\_#    **SPI::NX scoped objects** should be preceded by an “S” followed by the object type and #
- E.Object\_#    **ETH3 scoped objects** should be preceded by an “E.” followed by the object type and #

The **ETH3.SCOPE** script keyword marks the script to be a target to run on the ETH3, un-scoped variables generate a compiler error.



The scoping dialog shown below using the **TRANSFER INSTRUCTION** (named OUTPUT ASSIGNMENT INSTRUCTION when used on an **NX** controller context instead of the **ETH3**).



The term **LOCAL database** means that all the registers are on the internal memories of the **ETH3** so they can be accessed quite fast, even the COM1, COM2 and IP client remote points, as once they are polled by their respective masters, the values of this remote points reside on the **ETH3**'s memories and can be read or written by the PLC and any other application or protocol needing them.

Whereas the **REMOTE database** that comes from the **NX/NG** and is accessed via the **SPI bridge**, which the PLC must share with all other protocols and applications using it and it is slower. Therefore a **50-point cache** has been added to somehow relief the pressure put on the SPI bus by the PLC instructions executing faster than the SPI bridge can supply.

For this scoping to work seamlessly the software scoping hides the complexities of remapping that happen on the PLC operands.

On the **NX/NG** databases the addressing of every individual type is limited to an eight-bit object number, so that only registers from 1 to 255 can be used.

To break this limitation and allow the PLCs of the **ETH3** to access more than 255 registers of each type, the internal addressing for the operands in the **ETH** for the **OUTPUT ASSIGN/TRANSFER** instruction has been increased to 11 bits, thus allowing to map registers in the range from 1 up to 2048.

We can see that if a linear addressing were used it would be hard to read by humans, so to help with this the configurator software uses **scoping** for both its source and destination operands to help with the translation.

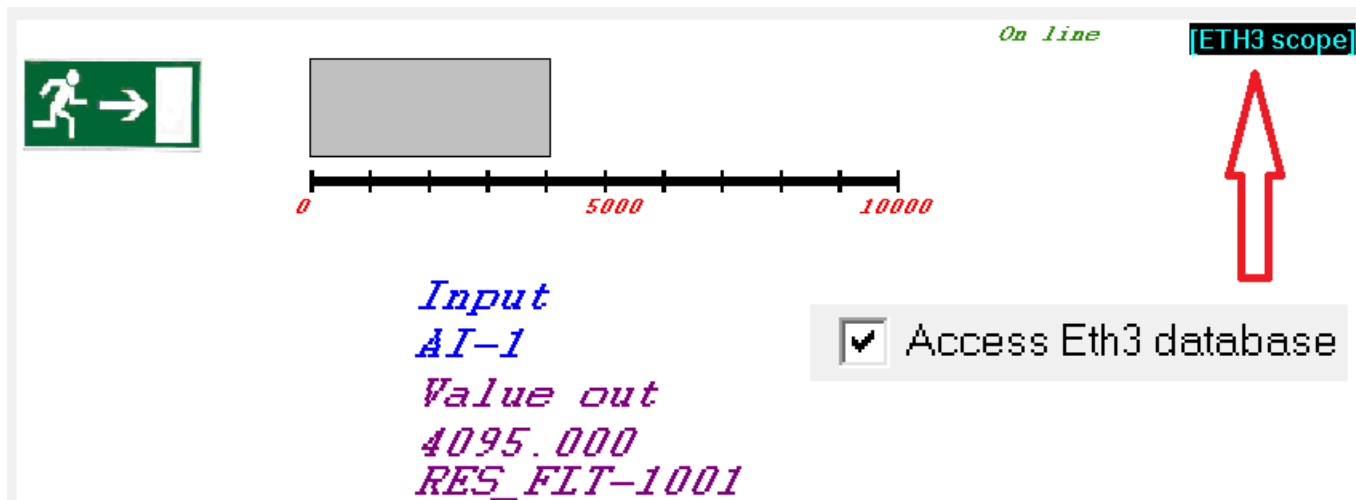
From the table in the previous page we can clearly see that all **SPI:NX** registers are less than 255 and all **ETH3** registers are greater than 1000, so internally everything is clear for the PLC.

**To be clear:** Only the **TRANSFER INSTRUCTION** can read and write the **SPI::NX REMOTE DATABASE** into the PLC. All other PLC instructions on the **ETH3** refer to the **ETH3's LOCAL DATABASE** only.

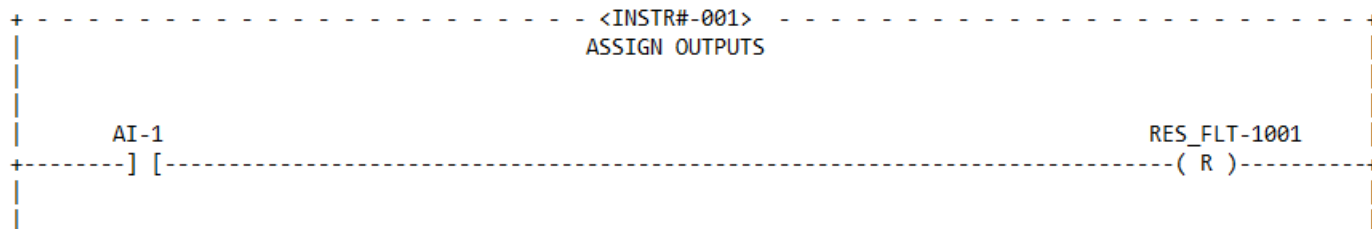
If for example you select that the source operand is the analog **AI-1 of the SPI::NX** and the destination is **RES\_FLT-1 on the ETH3**, when you select OK you will have the instruction write the registers for the **TRANSFER INSTRUCTION** as shown on the next figure.

First of all, take note that because the ACCESS ETH3 DATABASE check box is marked, we are dealing with the PLCs of the ETH3 and not with those of the NX/NG that is attached on the SPI bus.

Then also note how the source operand which is the AI-1 coming out from the SPI::NX is a number less than 255, then take a look at the destination register which is the RES\_FLT-1 of the ETH3 and due to this, it is remapped into the RES\_FLT-1001 register inside of the PLC instruction.



When we generate the documentation, this is exactly what we see.

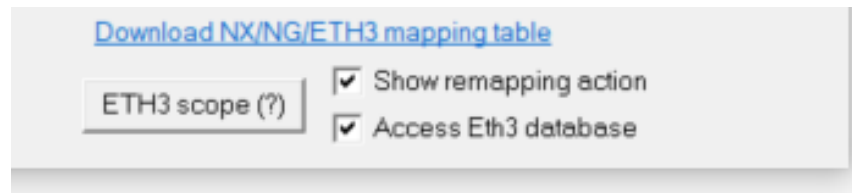


Note that all the other NON-TRANSFER instructions on the ETH3 PLCs, refer to the operands using 8-bit addressing. Therefore, only the first 255 registers of each type can be directly addressed. Any object 255 above should be first transferred to an ETH3's RES\_FLT register for use with the **TRANSFER INSTRUCTION**.

Also, the first 255 remote points of **E.COM1**, **E.COM2** and **E.IP** can be used directly as an operand on any NON-TRANSFER instruction by using the ALTERNATE column of the table shown two pages back, as it is clear that these instructions can only access the LOCAL ETH3 DATABASE so there is no confusion as to which registers to access. The same applies to the real time clock and calendar **E.CLK** and the I2C front mounted expansions of the ETH3 mapped into **E.AO**.

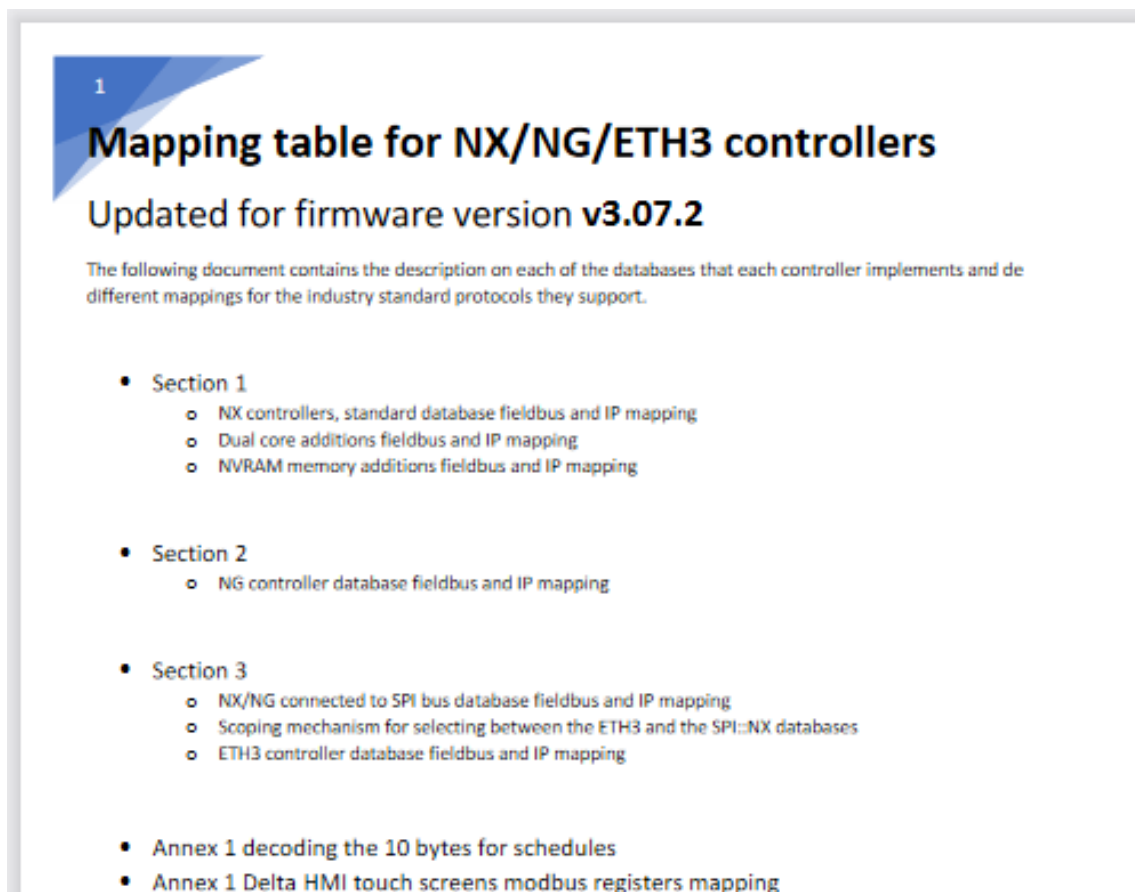
Newly added to the configurator on the **OUTPUT ASSIGN/TRANSFER INSTRUCTION** is a help button and a link to the mapping table for if you need help while creating your PLC instructions or planning a protocol conversion interface with a 3<sup>rd</sup> party controller.

ETH3 scoping help button and option check boxes.



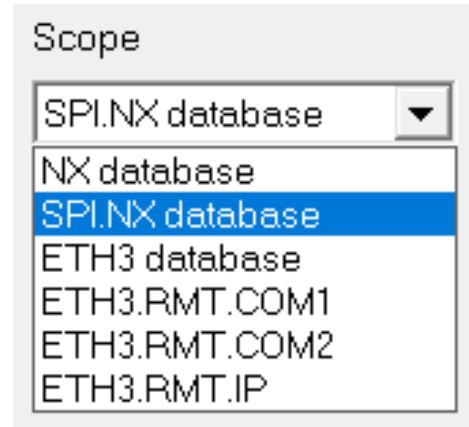
On the bottom right corner of the dialog, you will see the following UI elements:

- An **ETH3 scope (?)** help button that will pop up a help dialog.
- A **Show remapping action** checkbox that when checked “*provides assistance*” every time you select a database scope, can be unchecked if you are annoyed by the help or already have the mapping set in your head.
- An **Access ETH3 database** checkbox that when checked accesses the two ETH3’s PLCs otherwise if there is an NX connected over the SPI bus it accesses the NX’s three PLCs.
- A **Download NX/NG/ETH3 mapping table** link, that downloads the most recent available update of the mapping table document from the internet as shown below:



Pressing the **ETH3 scope (?)** help button pops-up a dialog that contains an explanation of the scoping of the ETH3 database and a short table showing the relationship of the scoped and un-scoped registers, remember that in the ETH3 PLC only the OUTPUT ASSIGN/TRANSFER INSTRUCTION uses internally the un-scoped ranges, all other PLC instructions use the ETH3 scoped ranges automatically, that is the reason you will not see the scoping field depicted below in all the other PLC instructions.

ETH3 db types	RANGES (scoped ranges)	OUTPUT ASSIGN instruction (unscoped ranges)
AI	None	None
AO	1..40	AO-2001..2040 (i2c expansions)
BI	None	None
BO	None	None
ADF	1..100	ADF-1001..1100
ADI	1..100	ADI-1001..1100
ADB	1..100	ADB-1001..1100
RES_BIT	1..255	RES_BIT-1001..1255
RES_FLT	1..255	RES_FLT-1001..1255
TMR	1..16	TMR.1001..1016
RMT.COM1	1..1000	AO-1001..2000 Alternate: AO-1..255
RMT.COM2	1001..2000	RMT-1001..2000 Alternate: RMT-1..255
RMT.IP	1..255	RES_FLT-1501..1750 Alternate: AI-1..255
RTCC	Time/Date	ADI-201..210



From the scoping field shown above on the right, the first option NX DATABASE is not available when you have activated the ACCESS ETH3 DATABASE checkbox, so only the option for accessing the **SXI.NX database** and **ETH3.XXXX** scopes are available for you.

This can be better explained by an example, here we use an ADD instruction and add three ETH3's RES-FLT registers 1, 2, 3 and place the result on RES\_FLT-4. As we can see both on the configuration dialog and on the ON-LINE screen because we have checked the ACCESS ETH3 DATABASE checkbox, **ETH3 scope** is automatically selected and displayed on the top right section of the instruction canvas.

On line **ETH3 scope**

LOGIC / MATH / COMP.

Instruction [2] result FLOAT

ADD R\_FLT = OP1 + OP2 + OP3 + OP4

Result RES\_FLT 4

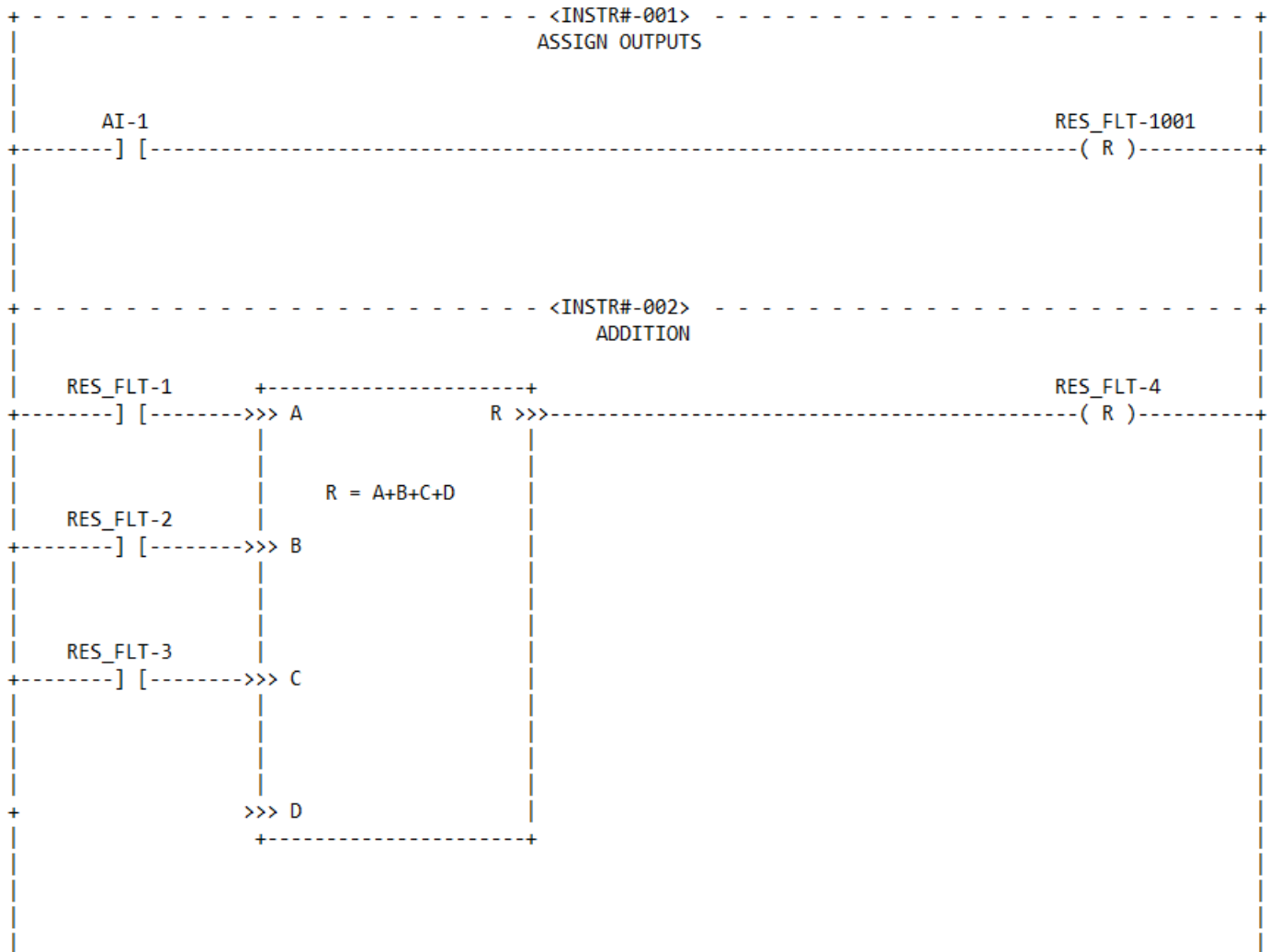
RES\_BIT-1.20 Lighting groups 1.20  
 RES\_BIT-21.36 Timers 1.16  
 RES\_BIT-37.38.39 Fixed timers 1,2,4 seconds  
 RES\_BIT-40.219 User free  
 RES\_BIT-247 Online as slave COM1  
 RES\_BIT-248 Online as slave COM2  
 RES\_BIT-249 Fixed timer 1 minute  
 RES\_BIT-250 Local override PB1  
 RES\_BIT-251.253 Power ON timer +1, +5, +10 seconds  
 RES\_BIT-254 LED green  
 RES\_BIT-255 LED red

Orerand 1: RES-FLT\_\_RAM-Result. float (1..40) 1  
 Orerand 2: RES-FLT\_\_RAM-Result. float (1..40) 2  
 Orerand 3: RES-FLT\_\_RAM-Result. float (1..40) 3  
 Orerand 4: NULL 1

Copy Paste Paste ++ Variable Name  Access Eth3 database

Now if we generate the documentation on this second instruction we added, it can be seen that on instruction <001> the ASSIGN OUTPUT instruction refers to ETH3 variables with register numbers > 1000 and SPI::NX registers if the number is <= 255.

While on the ADD instruction on PLC instruction <002> all registers default to **ETH3 scope** so their numbering will always be in the 1-255 range and directly addresses the registers of the ETH3



## Access the expanded remote database of a companion NX controller via the high-speed SPI bus.

As explained on the next section, when the PLC tries to access variables from the SPI::NX remote database instead of the ETH3's local database it must go thru the SPI bus to fetch the current value, on doing this it incurs in a speed penalty for having to wait for the NX to reply back with the data thru a shared bus, therefore to avoid stalling the PLC waiting for the data, a **50 point cache has been added** where the PLC only waits if the value is not already cached.

Thus when creating your PLC programs try to reduce the number of variables the PLC accesses from the remote SPI::NX database to improve latency and execution speed of the PLCs.

There is a diagnostic tool in the MFC configurator software to help you see how this cache is working out. Select the STATISTICS button on the main screen while connected to an NX controller connected to the ETH3 via the SPI interface to get the following screen:

The screenshot displays two windows from the MFC configurator software. The 'Comm ports statistics' window on the left shows various communication metrics for different ports (USB, SPI, COM1, COM2) and includes a 'Metasys' section with checkboxes for 'COS test Metasys', 'ACK', 'COS init.', and 'All'. A red box highlights the 'ETH3 SPI::NX bridge cache' button, with a yellow arrow pointing to the 'ETH3 SPI bridge cache statistics' window on the right. This second window provides detailed cache and PLC performance data.

General cache and PLC info	
Cache miss pause:	25 ms
Cache miss wait:	0 ms
PLC ms act.:	1
PLC ms max.:	57
PLC loop/s.:	107
PLC block/s.:	857

Read cache	
Expire K:	300 ms.
Size:	0
Hits:	0
Miss:	0

Write cache	
Expire K:	200 ms.
Size:	0
Hits:	0
Miss:	0

On this dialog, important performance information on the SPI BUS cache usage is provided. Take care that the SIZE parameter of either the write or read caches do not exceed 50 or degraded PLC execution could result.

Timing of the performance of the PLC execution is provided in the GENERAL CACHE AND PLC INFO section where the:

- **PLC ms act** Is the current time to execute the current instruction.
- **PLC ms max** Is the maximum time any instruction has taken to execute when waiting for SPI::NX data.
- **PLC loop/s** Provides information how many times per second the user PLC program is scanned/executed.
- **PLC block/s** Provides information on how many instructions per second the PLC executes

The CLEAR STATISTICS button resets the statistics and maximum waits and re-loads new statistic values.

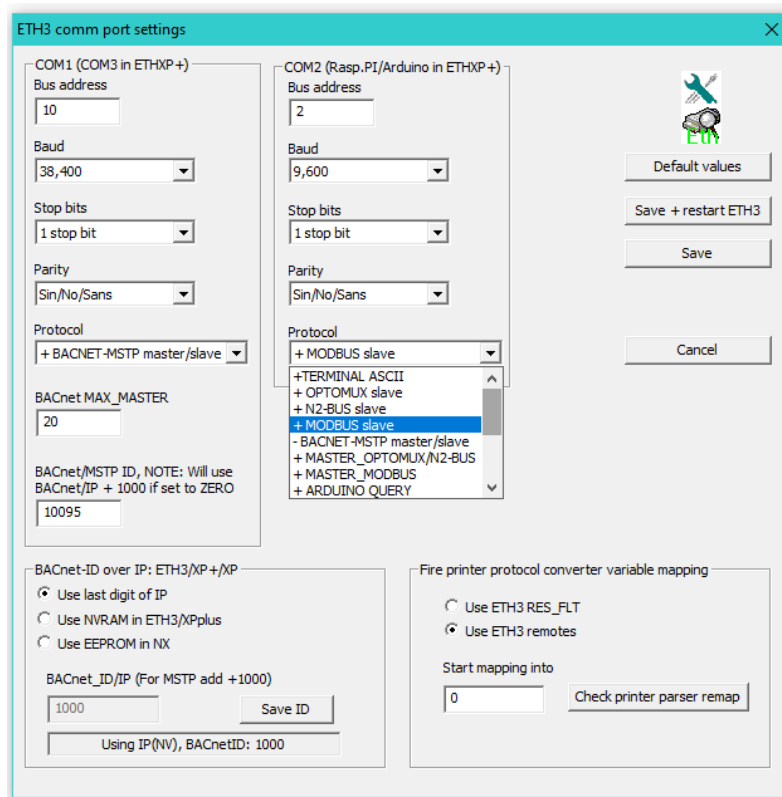
# Setting the ETH3's field buses as masters in BACnet, Modbus or Optomux.

There are different ways to set the protocols of the fieldbuses of the ETH3, the first one is using the MFC configurator software, to do this first connect using an IP address with an ETH3, it can be either a stand alone ETH3 or have a SPI::NX attached. The procedure is still the same as this setting only pertains to the ETH3's field buses COM1 and COM2.

When clicking on either the VIEW STATUS or the CONFIGURE main screen buttons you will find the following UI element:



When clicking it the dialog for setting the ETH3's COM1 and OCM2 remote points will pop up and click in the button labeled: **CONFIGURATION OF COM 1 AND COM2 PORTS OF THE ETH3.**



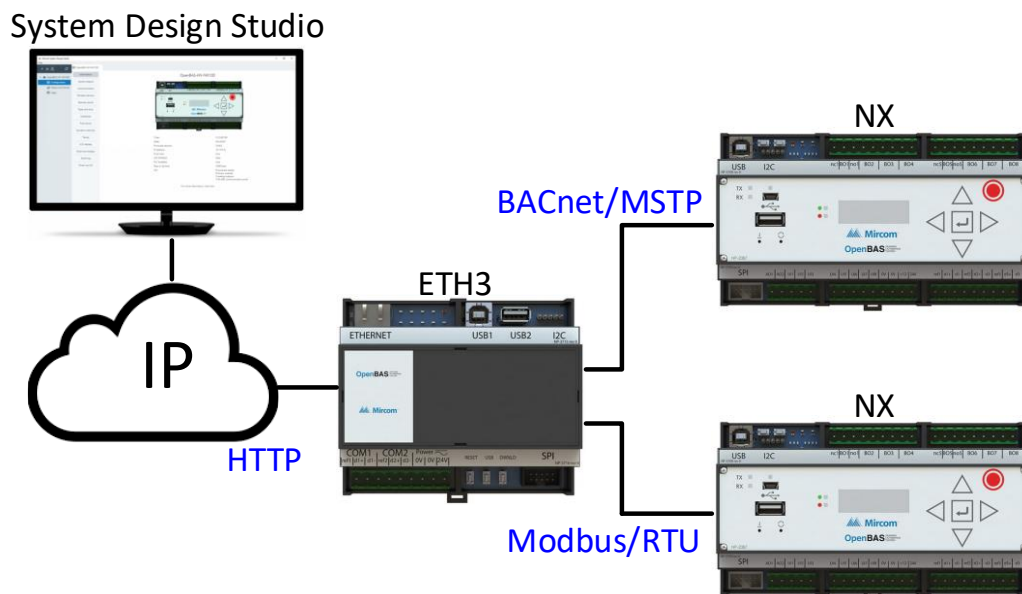
On this dialog the supported Master or Slave protocols of each COM port can be chosen. If the protocol has a plus sign [+] prefixing the protocol, it is supported by the firmware, if it has a minus sign [-] instead, it is not supported. Please be sure that the ETH3 has the most recent firmware installed to obtain the most protocols supported and bug fixes.

Note that BACnet/MSTP is only supported in COM1, and additional settings are needed. Also, in this dialog the BACnet/IP-ID setting can be entered as well as some additional fields for when a Mircom fire panel parser is used to convert the information sent over the printer port to all the protocols that the ETH3 supports.

## Accessing slaves for programming on the fieldbuses on any protocol using bridging and tunneling.

When installing OpenBAS-NX controllers on the field busses, the ETH3 acts now not only as a gateway but also as a protocol converter using bridging and tunneling so that the user is able to program these NC controllers over its COM1 and COM2 fieldbuses without any regard of which protocol they are set to.

The following figure depicts this, as two slaves are connected to COM1 and COM2 field buses using two different industry standard protocols and can even have 3<sup>rd</sup> party native controllers and still the SDS configurator software can program them seamlessly.



The three master protocols that can be used when connecting slaves are In **COM1** are:

- **Optomux/N2-OPEN** Bridging is used to program slaves via IP.
- **Modbus-RTU** Tunneling employing Modbus-MEI is used to program slaves via IP.
- **BACnet/MSTP** Tunneling employing BACnet proprietary frames is used to program slaves via IP.

The two master protocols that can be used when connecting slaves are In **COM2** are:

- **Optomux/N2-OPEN** Bridging is used to program slaves via IP.
- **Modbus-RTU** Tunneling employing Modbus-MEI is used to program slaves via IP.

The next four pages provide an insight into the technicalities of using these bridging and tunneling features. You don't need to understand any of these complexities to use bridging and tunneling as both SDS and the MFC tool rely on the ETH3's capabilities to do all this transparently so neither the user nor the configurator programs need to be aware of this.

**Note that to use this tunneling/bridging feature the ETH3 as well as all the slave controllers on the field busses must have their firmware upgraded to v3.10.0.**



First to convey an Optomux message using HTTP, **SDS** generates a URL that provides information on how the message should be routed, four different URL encodings allow any of the following routes to be used internally on the ETH3:

URL	Use
<b>opto22.htm</b>	ETH3/SPI_NX bridging
<b>com1_e.htm</b>	ETH3/COM1 bridging or tunneling
<b>com2_e.htm</b>	ETH3/COM2 bridging or tunneling
<b>eth_22.htm</b>	ETH3 internal database

This is a typical URL that is requesting to send an Optomux request to an NX controller that is connected in **COM1** that is configured to use a BACnet/MSTP network using MAC 'X06':

[http://192.168.100.95/com1\\_e.htm?id=39&cmd=>061E7300000066\r&btn=Enviar](http://192.168.100.95/com1_e.htm?id=39&cmd=>061E7300000066\r&btn=Enviar)

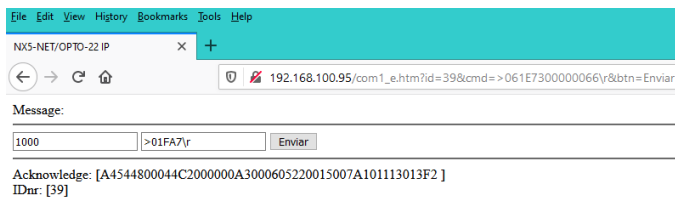
The **URL encoding** is shown below as provided on the LOG file for IP communication, highlighting both the request and the response or acknowledge:

```

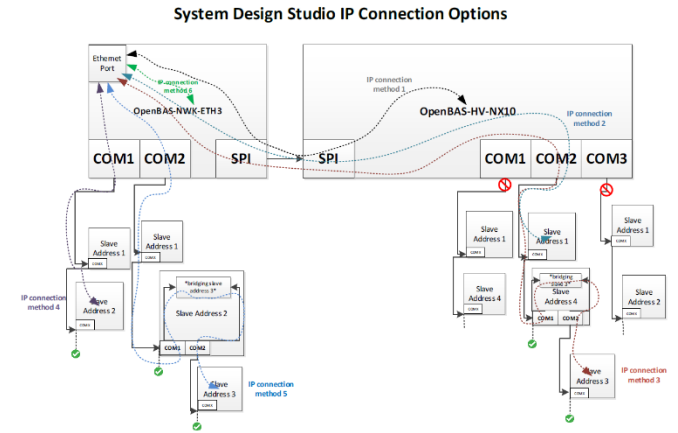
-----
[TIME: 06:39:02.184 DATE: 03/03/2021]
gIP_sequenceIDnr:39
HTML Message: 'http://192.168.100.95/com1_e.htm?id=39&cmd=>061E7300000066\r&btn=Enviar'
HTML Acknowledge:
<!doctype html>
<meta http-equiv="refresh" content="60">
<html>
  <head>
    <title> NX5-NET/OPTO-22 IP</title>
  </head>
  <body>
    Message:<br>
    <form method="get" action="com1_e.htm">
      <input type="text" name="id" value="1000">
      <input type="text" name="cmd" value=">01FA7\r">
      <input type="submit" name="btn" value="Enviar">
    </form>
    <br>
    Acknowledge: [A4544700044C1C00000A8000805220015007E1011130130E
  ]<br>
  IDnr: [39]
</body>
</html>

```

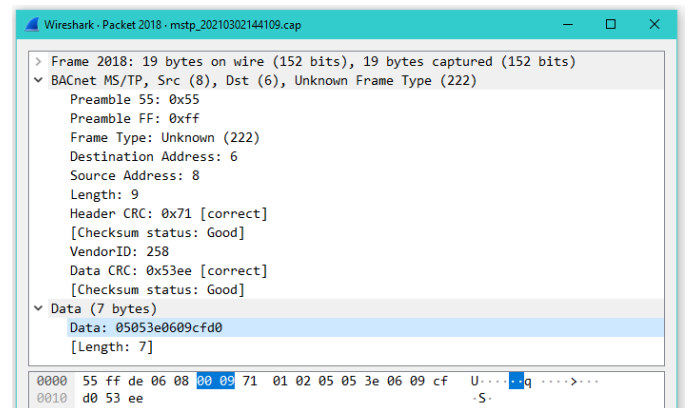
The nice thing is that you can also use any browser to do BACnet or Modbus tunneling without any effort, simply copy and paste the URL and place it in any browser's URL field and you will automatically get an Optomux response with the acknowledge encoded as shown below by the ETH3's integrated CGI (**Common Gateway Interface**):



The following diagram taken from **SDS v1.1.0** manual, shows the internal routings according to the URL:



Using Wireshark network analyzer software, we can easily decode the information present on the BACnet/MSTP network as depicted below:



As we can see, the BACnet/MSTP tunneling employs the proprietary MSTP frame **222**, thus if we dissect the MSTP frame we will have the following:

The 0x55, 0xFF preamble followed by the frame type 222.

If we read the BACnet standard on section **9.3 MS/TP Frame Format** we find that:

*“Frame Types 128 through 255 are available to vendors for proprietary (non-BACnet) frames. Use of proprietary frames might allow a Brand-X controller, for example, to send proprietary frames to other Brand-X controllers that do not implement BACnet while using the same medium to send BACnet frames to a Brand-Y panel that does implement BACnet.*

*Token, Poll For Master, and Reply To Poll For Master frames shall be understood by all MS/TP master nodes.”*

Then follows the Destination and source MAC addresses followed by the payload's data length (if any, otherwise it is set to zero) and finally the header's CRC checksum.

Then after the header, comes the DATA section, that ahead or it contains a WORD indicating the Vendor ID of that Frame, any of the following can be used for NX controllers as taken form the ASHRAE’s BACnet site:

<http://www.bacnet.org/VendorID/index.html>

505	Mircom Group of Companies	Jason Falbo	25 Interchange Way Toronto, ON L4K 5W3 Canada
828	SAMDAV	Ricardo Galnares	Cerrada de Lerdo 6-E Distrito Federal, CP D.F. 10580 Mexico

The DATA section contains the Optomux message that uses the same compression method that Modbus **MEI (Modbus Encapsulated Interface)** employs to allow code reuse in both the ETH3 and NX firmware and is explained below in detail:

If we take a look at the document:

**Modbus\_Application\_Protocol\_V1\_1a.pdf** that can be downloaded from the [www.modbus.org](http://www.modbus.org) web page we will find in the following section:

### 6.19 Function code 43 (0x2B) Encapsulated Interface Transport

*“Function Code 43 and its MEI Type 14 for Device Identification is one of two Encapsulated Interface Transport currently available in this Specification. The following function codes and MEI Types shall not be part of this published Specification and these function codes and MEI Types are specifically reserved: 43/0-12 and 43/15-255.*

*The MODBUS Encapsulated Interface (MEI)Transport is a mechanism for tunneling service requests and method invocations, as well as their returns, inside MODBUS PDUs.*

*The primary feature of the MEI Transport is the encapsulation of method invocations or service requests that are part of a defined interface as well as method invocation returns or service responses.”*

This is a typical Modbus MEI packet:

#### Request

Function code	1 Byte	0x2B
MEI Type*	1 Byte	0x0E
MEI type specific data	n Bytes	

\* MEI = MODBUS Encapsulated Interface

#### Response

Function code	1 Byte	0x2B
MEI Type	1 byte	0x0E
MEI type specific data	n Bytes	

#### Error

Function code	1 Byte	0xAB : Fc 0x2B + 0x80
MEI Type	1 Byte	0x0E
Exception code	1 Byte	01, 02, 03, 04

Because both **BACnet** and **Modbus** employ binary data as compared to **Optomux** that uses ASCII encoded plain text, to save space and make the tunneled data shorter and faster to be conveyed, an **encoding / decoding** is implemented by the ETH3 that **compresses / decompresses** the tunneled Optomux message before being placed into the BACnet or Modbus frames and is as follows:

In modbus the data part of the tunneled Optomux message is compressed if only hexadecimal characters ‘0’ thru ‘F’ are present on the message, the first character of the message used as the START flag: ‘>’ is sent uncompressed, the ‘\r’ END character is encoded as either a 0xD character on the lower nibble of the last byte and then the last byte will be encoded as 0x00 or if the number of characters are even, then it will be encoded as a 0x0D byte at the end.

Below is a **Modbus-MEI** encoded compressed packet.

```

*** modbus MEI COMPRESSED encoding examples ***
* Address      01 // Header
* FunctionCode 2B // Header
* SubCode     64 // Header
* Size        05 // Header, size: compressed: BIT_7 = FALSE + n,
* Size        05 // Header, size: compressed: BIT_7 = FALSE + n ||
* Data: '>'    3E // Data, 1st character always uncompressed
* Data: '0'1' 01 // Data, Encoded data compressed
* Data: 'F'A'  FA // Data, Encoded data compressed
* Data: '7'\r' 7D // Data, Encoded data compressed
* Data: '\Null\r' 00 // Data, Null 0x00 || 0xD0
* CRC         xx // CRC lo
* CRC         xx // CRC hi
***
  
```

For messages less than 3 bytes or that have other ASCII characters outside of the ‘0’ thru ‘F’ range that are non-valid Optomux characters, they are sent uncompressed as shown below.

```

*** modbus MEI UNCOMPRESSED encoding examples. NOTE:
* Address      01 // Header messages less than 3 characters
* FunctionCode 2B // Header will be sent uncompressed
* SubCode     64 // Header
* Size        01 // Header, size: compressed: BIT_7 = FALSE + n ||
* Data: 'A'    41 // Data, Encoded data uncompressed
* CRC         xx // CRC lo
* CRC         xx // CRC hi
***
  
```

This encoding example for Modbus/RTU also applies to BACnet/MSTP using the proprietary frame type 222 (0xDE) and the DATA occupies the rest of the packet.

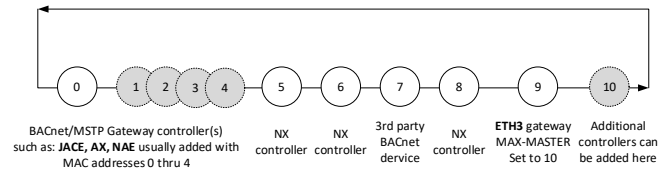
Both protocols use a 16-bit CRC to verify the integrity of the packet, and the tunneled message also contains its own CRC thus the data's integrity is double-fold verified upon arrival as both CRC's are checked.

### BACnet/MSTP networking tips:

To improve the operation of (any) BACnet/MSTP network, the following points have to be followed, not doing so will degrade the bandwidth and thus the performance of the network:

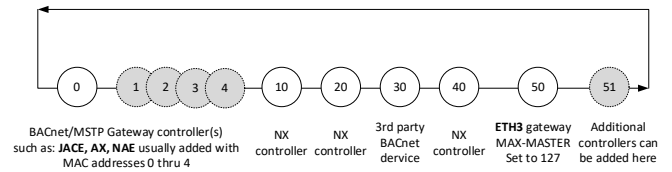
- MAC addresses 0-4 are usually reserved for bus Gateways, remember all **BACnet/MSTP** (Master-Slave-Token-Passing) devices are both true **Masters** and **Slaves**. When a MAC address holds the **Token** it becomes the Master and all other peers are then Slaves, hence it is a true multi-master network. Only devices with "A" functionality, normally used as Gateways employ the largest part of the bandwidth for polling and reading or writing data.
- All other (non-gateway or router) devices can use MAC addresses 5 thru 127.
- Leave **NO** gaps or spaces between MAC addresses to avoid waste maintenance **Poll-For-Master** frames that in average stall the network with a dead time around 20-80 milli seconds for each frame sent when the polled device does not exist or is off-line. Each 50 times that the token is received by any MSTP device on the network, it must do a maintenance poll-for-master cycle looking for added devices to heal broken token sequences. These are completely avoided by using contiguous addresses.
- If possible, assign **ETH3's** fields bus port **COM1** to use the **last MAC address** and set its **MAX-MASTER** to its own MAC address + 1, to still allow for more devices to be added on top of it if when necessary and wasting no unnecessary time.

The following diagram shows a typical good networking scenario.



Here the lower MAC addresses that are usually assigned to gateways in the range 0 to 4, ensure they start the token sequence ahead of any other devices that are present. Then a **no-gap** MAC addressing scheme from 5 thru 8 is assigned. At the end place the ETH3 whose MAC address is set to 9 and has its **MAX MASTER** set to 10 thus it will only look up to address 10 before reverting back to 0 to search for other devices and pass the token.

A poor network addressing scheme is shown below, where gaps exist between the MAC addresses.



Here, for example MAC address 10 will periodically every 50 tokens start a maintenance **Poll-For-Master** sequence looking for MAC addresses: 11, 12, 13, 14, 15, 16, 17, 18 and 19 before finally finding MAC address 20.

The same wasteful sequence will again happen for MAC addresses 20, 30, 40 and 50. Even worst for this last one address, as its MAC address is set to 127 which is the maximum allowed MAC address for MSTP devices and every 50 tokens received, it will look for addresses from 51 and all the way thru 127 and then fold back to MAC address 0 all the way to 49 until it finds some device that is present.

**And this will repeat forever, for every single node every 50 times the token is received by it!**

Once again, but now in big size, the golden rules for MSTP networks are:

- ⊗ **Leave NO-GAPS in MAC addresses.**
- ⊗ **Set MAX MASTER of the last device on the network to its own MAC address + 1.**
- ⊗ **Follow good RS485 wiring practices using EOL and BIAS resistors to properly polarize the bus and avoid reflections that will corrupt data.**

The following **Wireshark** MSTP capture shows this wasteful sequence in great detail, where each **Poll-For-Master** frame kills around 50-80 milli seconds of network bandwidth in the field bus.

Frame #	Timing	Difference
3	0.014991	
4	0.094945	0.079954
6	0.110937	
7	0.158909	0.047972
10	0.174900	
11	0.228720	0.053820
14	0.238863	
15	0.286836	0.047973
18	0.302827	
19	0.350799	0.047972

Maximum	0.080
Minimum	0.048
Average	0.056

Image showing the **Poll-For-Master** sequence of different MAC addresses with GAPS.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0x08	0x09	BACnet	8	BACnet MS/TP Poll For Master
2	0.000000	0x08	0x04	BACnet	8	BACnet MS/TP Token
3	0.014991	0x04	0x05	BACnet	8	BACnet MS/TP Poll For Master
4	0.094945	0x04	0x06	BACnet	8	BACnet MS/TP Token
5	0.110937	0x06	0x08	BACnet	8	BACnet MS/TP Token
6	0.110937	0x08	0x0a	BACnet	8	BACnet MS/TP Poll For Master
7	0.158909	0x08	0x04	BACnet	8	BACnet MS/TP Token
8	0.158909	0x04	0x06	BACnet	8	BACnet MS/TP Token
9	0.174900	0x06	0x08	BACnet	8	BACnet MS/TP Token
10	0.174900	0x08	0x0b	BACnet	8	BACnet MS/TP Poll For Master
11	0.228720	0x08	0x04	BACnet	8	BACnet MS/TP Token
12	0.228720	0x04	0x06	BACnet	8	BACnet MS/TP Token
13	0.238863	0x06	0x08	BACnet	8	BACnet MS/TP Token
14	0.238863	0x08	0x0c	BACnet	8	BACnet MS/TP Poll For Master
15	0.286836	0x08	0x04	BACnet	8	BACnet MS/TP Token
16	0.286836	0x04	0x06	BACnet	8	BACnet MS/TP Token
17	0.302827	0x06	0x08	BACnet	8	BACnet MS/TP Token
18	0.302827	0x08	0x0d	BACnet	8	BACnet MS/TP Poll For Master
19	0.350799	0x08	0x04	BACnet	8	BACnet MS/TP Token
20	0.350799	0x04	0x06	BACnet	8	BACnet MS/TP Token
21	0.366789	0x06	0x08	BACnet	8	BACnet MS/TP Token
22	0.366789	0x08	0x0e	BACnet	8	BACnet MS/TP Poll For Master

By using a good MAC addressing scheme you will ensure a fast and responsive BACnet/MSTP network.

In this other example we can see that when MAC address 8 is taken off-line, the Poll-For-Master searching for the next MAC address heals the token cycle by polling from MAC addresses 9 thru 20 and then folds back from 0 all the way up until it finds MAC address 4.

At that time, the node with the MAC address 4 replies with the **Reply-To-Poll-For-Master** frame and then finally the token sequence can start again.

No.	Time	Source	Destination	Protocol	Length	Info
317	3.357142	0x06	0x08	BACnet	8	BACnet MS/TP Token
318	3.358084	0x08	0x04	BACnet	8	BACnet MS/TP Token
319	3.358084	0x04	0x06	BACnet	8	BACnet MS/TP Token
320	3.374075	0x06	0x07	BACnet	8	BACnet MS/TP Poll For Master
321	3.422047	0x06	0x08	BACnet	8	BACnet MS/TP Token
322	3.438038	0x08	0x04	BACnet	8	BACnet MS/TP Token
323	3.438038	0x04	0x06	BACnet	8	BACnet MS/TP Token
324	3.454029	0x06	0x08	BACnet	8	BACnet MS/TP Token
325	3.501028	0x06	0x08	BACnet	8	BACnet MS/TP Token
326	3.549009	0x06	0x09	BACnet	8	BACnet MS/TP Poll For Master
327	3.597006	0x06	0x0a	BACnet	8	BACnet MS/TP Poll For Master
328	3.644976	0x06	0x0b	BACnet	8	BACnet MS/TP Poll For Master
329	3.692949	0x06	0x0c	BACnet	8	BACnet MS/TP Poll For Master
330	3.724932	0x06	0x0d	BACnet	8	BACnet MS/TP Poll For Master
331	3.772904	0x06	0x0e	BACnet	8	BACnet MS/TP Poll For Master
332	3.820876	0x06	0x0f	BACnet	8	BACnet MS/TP Poll For Master
333	3.868849	0x06	0x10	BACnet	8	BACnet MS/TP Poll For Master
334	3.916790	0x06	0x11	BACnet	8	BACnet MS/TP Poll For Master
335	3.964793	0x06	0x12	BACnet	8	BACnet MS/TP Poll For Master
336	4.012766	0x06	0x13	BACnet	8	BACnet MS/TP Poll For Master
337	4.060680	0x06	0x14	BACnet	8	BACnet MS/TP Poll For Master
338	4.108710	0x06	0x00	BACnet	8	BACnet MS/TP Poll For Master
339	4.156684	0x06	0x01	BACnet	8	BACnet MS/TP Poll For Master
340	4.204656	0x06	0x02	BACnet	8	BACnet MS/TP Poll For Master
341	4.252628	0x06	0x03	BACnet	8	BACnet MS/TP Poll For Master
342	4.300606	0x06	0x04	BACnet	8	BACnet MS/TP Poll For Master
343	4.316592	0x04	0x06	BACnet	8	BACnet MS/TP Reply To Poll For Master
344	4.317565	0x06	0x04	BACnet	8	BACnet MS/TP Token
345	4.332583	0x04	0x06	BACnet	8	BACnet MS/TP Token
346	4.333555	0x06	0x04	BACnet	8	BACnet MS/TP Token

The whole process took from frames 326 thru 343 and spanned from 3.549 to 4.316 seconds = **767 milli seconds** therefore, to traverse those 17 nodes to heal the network, an average **Poll-For-Master** time of 45.5 milli seconds was used or better say wasted.

If **MAX MASTER** had been set to **127** instead of **20**, the time to heal a broken network would have taken instead:

$$\begin{aligned}
 &9 \text{ thru } 127 + 0 \text{ thru } 4 \\
 &= 122 \text{ nodes} \\
 &= \mathbf{5.504 \text{ seconds.}}
 \end{aligned}$$

## Adding remote points both manually and using BACnet discovery.

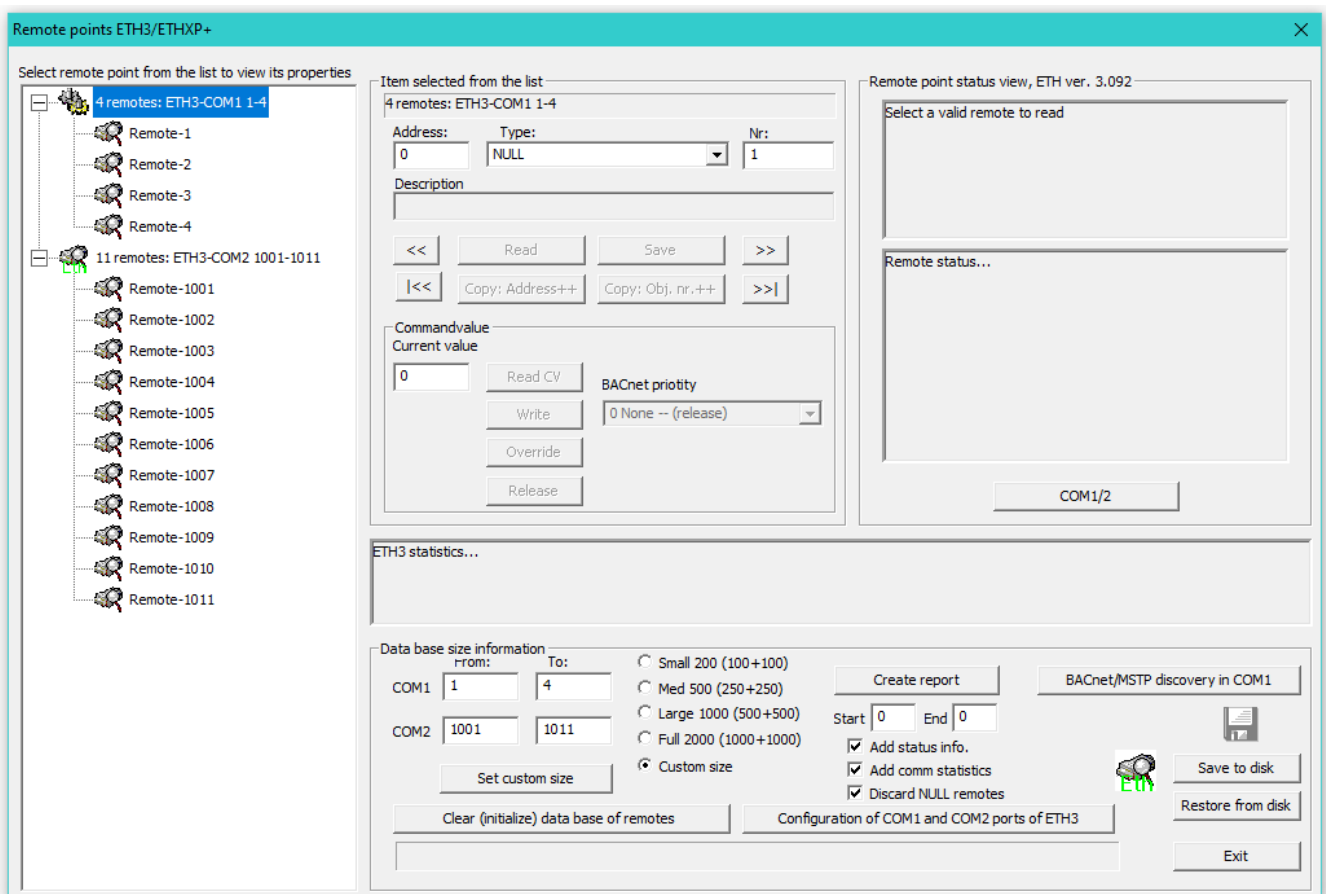
Now that you have two field buses on the ETH3, and each one can have 1000 remote points how do you add them? On future versions of the script compiler support will be added so that the remote points can be added programmatically. But in the meantime, there are ways to add the remote points the master protocols mentioned in the previous section.

For Optomux/N2-Open and modbus-RTU the only solution is to add them manually. For BACnet/MSTP this option is also available, but the ETH3 supports full device and point discovery, thus adding one or even multiple controllers and their points is a breeze.

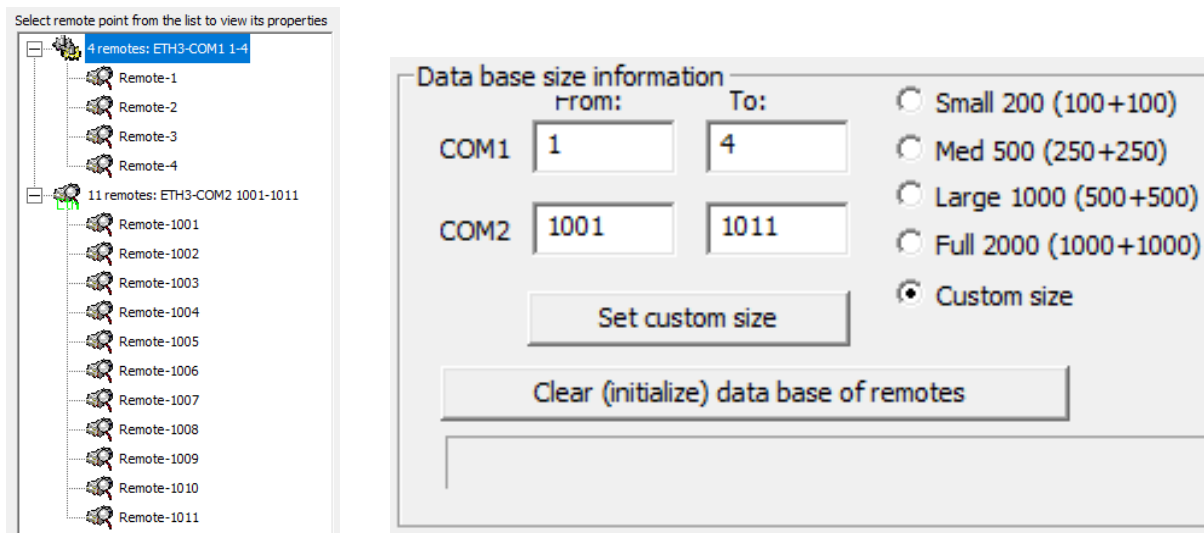
But before going on this newly added feature let's go to the basics. To get into the ETH3's remote points using the MFC tool (**support for SDS is coming soon to relief this job and abstract and ease the remote point's settings**) first click on either the VIEW STATUS or the CONFIGURE main screen buttons you will find the following UI element:



This will open the ETH3's remote dialog, and as we can see, it is a heavily UI loaded dialog, therefore on the next pages we will be providing an in-depth description of each section.



First to the left you find a tree pane where each COM1 and COM2 has the remote point elements that are first set by the user on the “Database size information” section depicted to the right.



First let’s remember that the 2000 remote points available on the ETH3, they are split in two, so then the remote points 1 thru 1000 are assigned to COM1 and 1001 thru 2000 are assigned to COM2.

The first thing to do is decide how many will be used for remote points, on the screenshot above you can see that COM1 has assigned four remote points 1 thru 4 and COM2 has eleven 1001 thru 1001.

All the remaining non used remote points will not be scanned or polled by the master protocol thus can be freely used for any other thing in the user program.

Notice to the right some pre-set options to size the remote’s database size to SMALL, MEDIUM, LARGE or FULL size as well as the CUSTOM SIZE used when you need a specific even asymmetric size for COM1 and COM2.

The CLEAR DATABASE button will clear the contents of the remote points, make sure you have saved any modified database to disk before using this option to avoid losing your unsaved changes.

On the next page more details on this dialog is provided.

As shown on the screenshot below, when you select a remote point from the tree pane on the left you will notice that the configuration for that specific point is shown in the ITEMS SELECTED FROM THE LIST section.

Also, the current value and the status (ON-LINE or OK, OFFLINE or WRITE MODE) is shown in the REMOTE POINT STATUS section, the point's current configuration is also shown here.

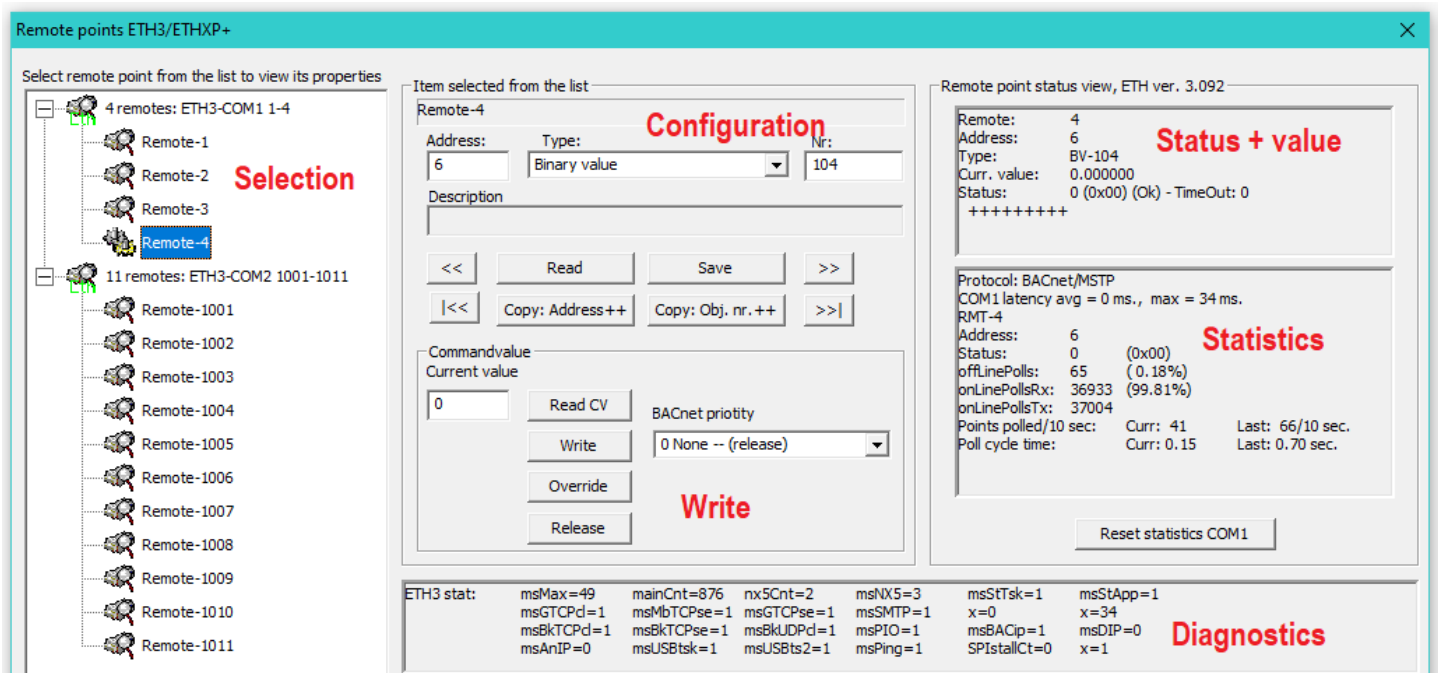
IN the STATISTICS section the point's communication performance can be viewed and analyzed, especially important is the ON LINE POLLS RX as this parameter shows how good or bad is the communication with this object. Anything below a 98-99% indicates there is a bus problem, either caused by:

- Poor wiring.
- Bus overload.
- Wrong or missing bus polarization and/or end of line resistors.
- Any driver on the bus wis defective or partially shorted.
- Noise introduced into the data lines caused by poor or defective shielding or ground loops.

The RESET STATISTICS COM1 or COM2 button clears the statistics of all the remote points of that specific COM port.

The COMMAND VALUE section allows modifying the current value by either a simple write or an override if allowed by the object or protocol, BACnet offers additionally prioritized commanding.

Finally the in the DIAGNOSTICS section some performance index statistic counters are provided to aid debugging the master poll engines as well as the internal timing of different components of the ETH3's stacks.

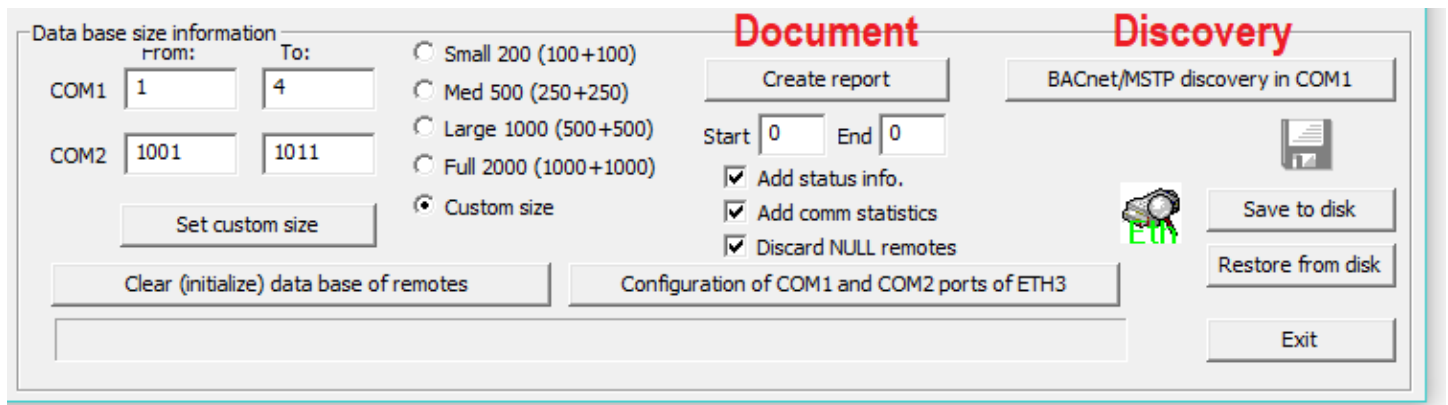


You can add remote points by filling in the CONFIGURATION section and pressing SAVE, Note that the object TYPE and number range gets updated according on the master protocol selected on that specific COM port.

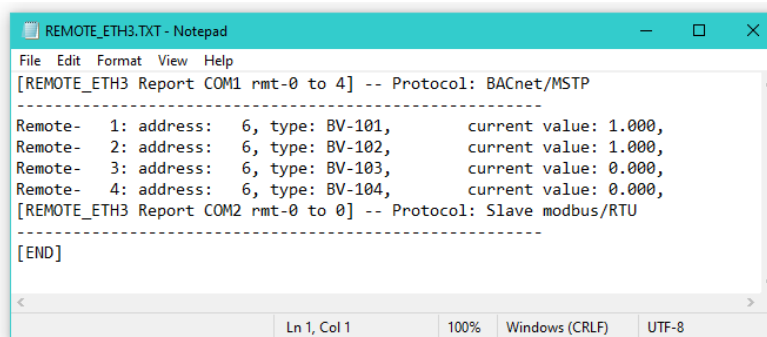
On the next page additional information on how to document your remote points configuration and current status as well as using the BACnet discovery feature added on v3.10 is provided.

The CREATE REPORT button offers the option to “Document” not only the configuration but also the current value, status, and communication statistics of the remote points.

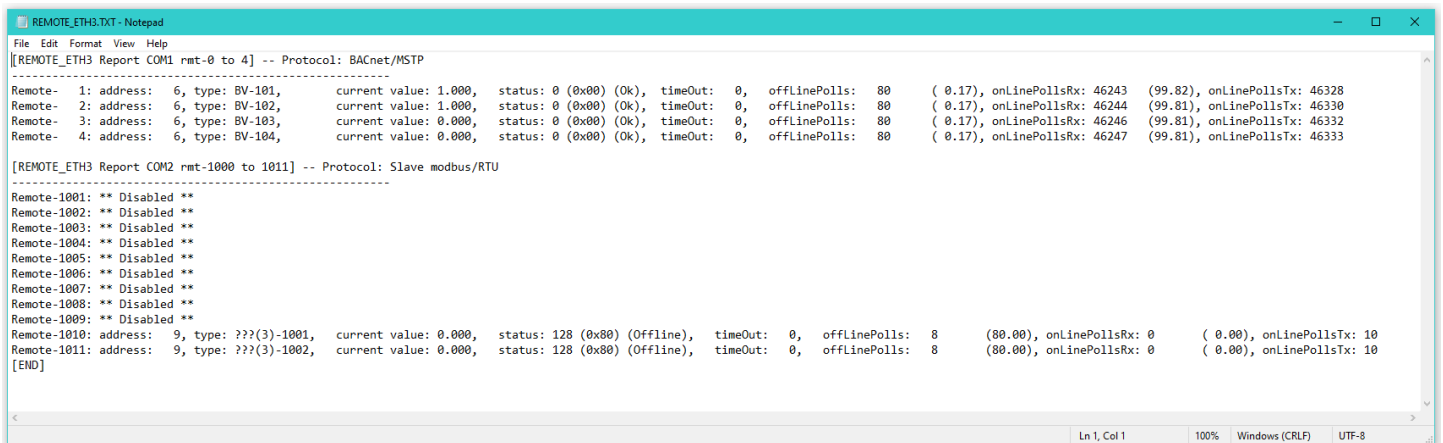
The START and END fields which by default are both set to ZERO indicate to document all the active points of the remote points database, manually selecting START and END range will only document that specific range.



A small, customized report showing only remotes: START=1 thru END=4 without STATUS and COMM STATISTICS (both options unchecked) is shown below.



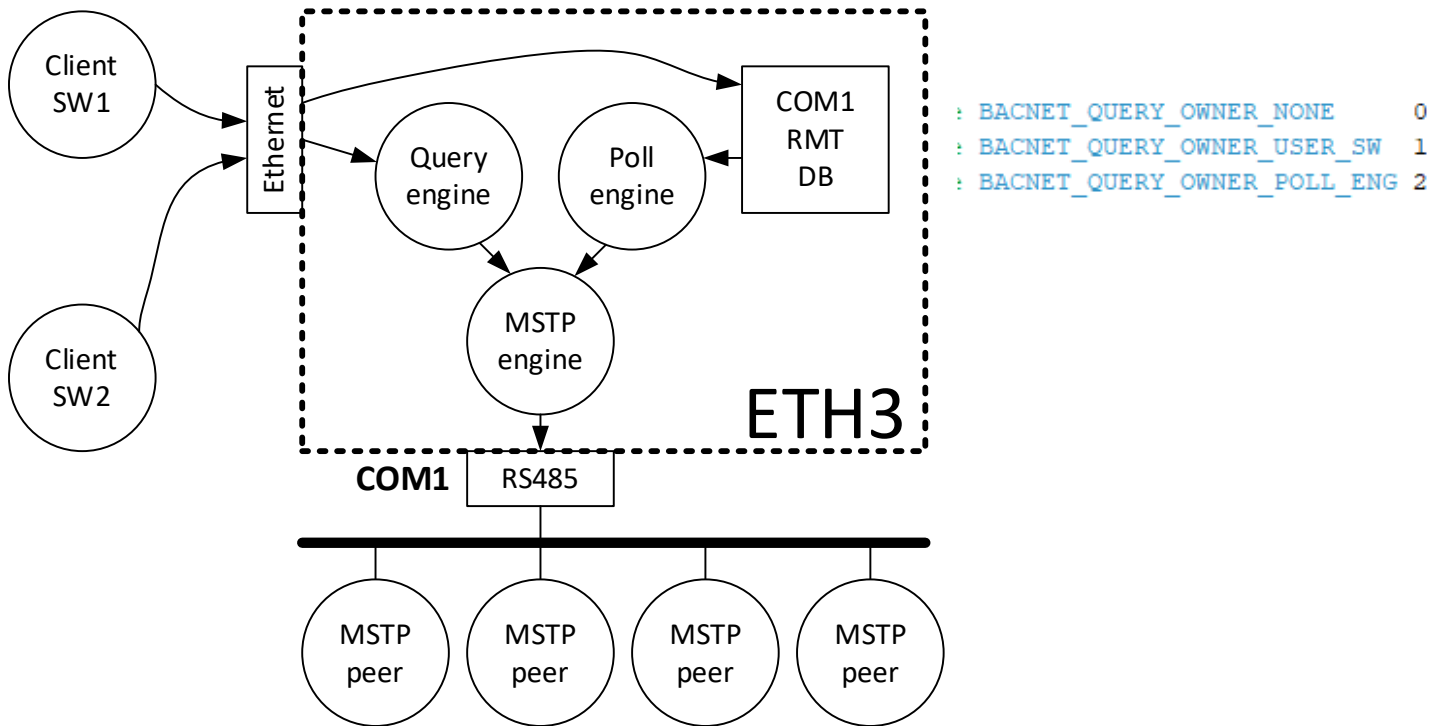
Whereas a full COM1 with 4 remotes and COM 2 with 11 remotes including STATUS and COMM STATISTICS information is shown below.



By pressing the BACnet/MSTP discovery in COM1 button a dialog described on the next page pops up..



A **Query Engine** has been integrated into the **ETH3** that handles not only the communication between with the client software(s) that want to query MSTP devices, but also interacts and synchronizes with the **Poll Engine** that uses the remote points database that exist in the **ETH3** to poll for read/write up to 1000 remote points on COM1.



As depicted on the image above, the two query engines interact with the MSTP engine previously described to send the queued messages while the MSTP is on the **USE\_TOKEN** state to the MSTP peer devices on the MSTP network.

Let's remember that an MSTP network can contain up to **128 nodes** in the addressing range of **0 and up to 127 inclusive** on what is called the **MAC** (Media Access Control) or aka. Device or Node **address**.

Also addresses in the range **128 and 254 inclusive** can be on the same network but behave as slave only devices which do not participate in the token passing mechanism and thus only respond to a query when addresses by master nodes in the 0 to 127 address range that hold the token.

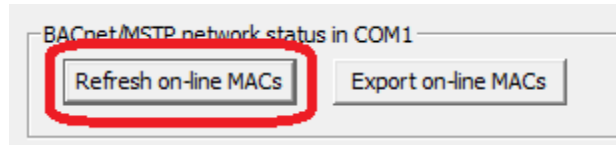
Finally, the MAC address **255** is defined as a **BROADCAST** address intended to address any device node on the network. This broadcast address is usually used to send broadcast messages such as the WHO-IS and WHO-HAS messages to discover devices or who owns a specific set of objects, as well as the TIME and DATE synchronization messages.

BACnet has a pair of services named **Who-Is and I-Am Services** as described on section 16.10 of the standard. With these services clients can discover over the network who is out there. The services come on three flavors:

- **Broadcast** Who-Is that is intended to discover **all devices** on a network.
- **Multicast** Who-Is that is intended to discover **a list of devices within a range** on a network.
- **Unicast** Who-is that is intended to discover a **single device** on the network.

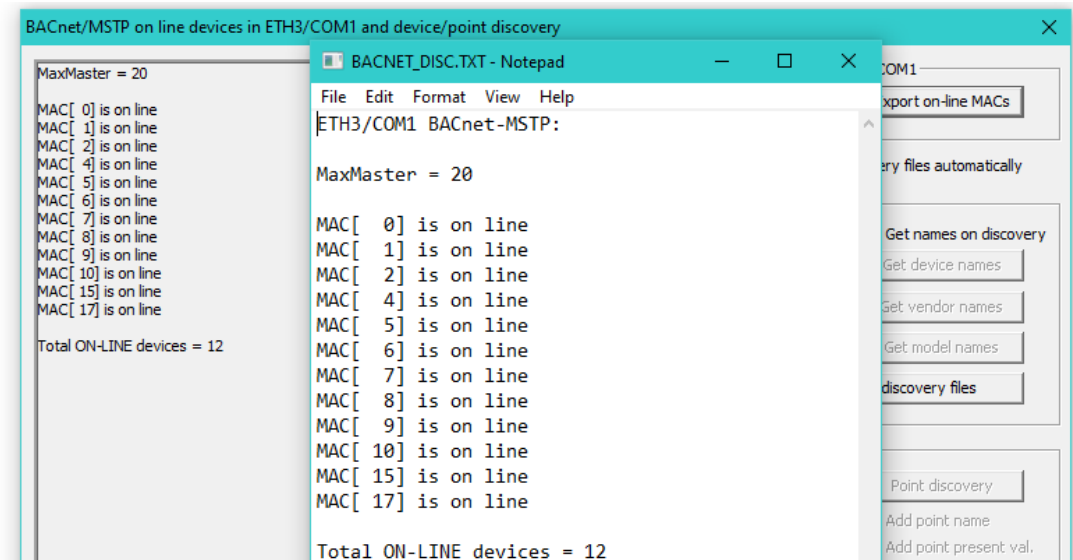
All messages are sent broadcast so any device on the network and possibly subnetworks can receive the request and reply with an I-Am message when appropriate. In the table below we can see the service primitive for the Who-Is service.

Adding devices with DEVICE discovery.

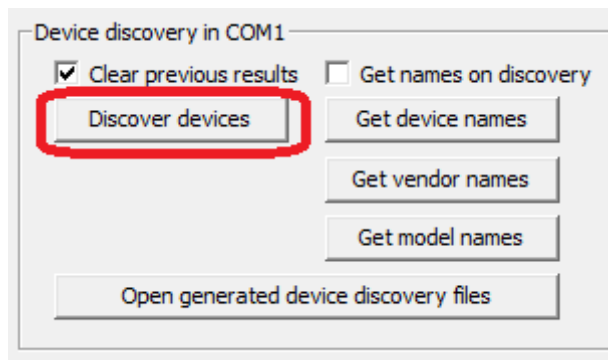


The **ETH3** besides supporting the device discovery service, also implements a **live MSTP on-line** monitoring system for the Polling and Queuing engines to avoid accessing Off-Line devices that are not present on the network.

To do this as the ETH3's receive state machine receives every single MSTP frame on the network, it keeps track each time a node sends a token, thus it keeps an array updated with the status of all of the possible 128 nodes that can be present on the network. This array is loaded with a 15 second timeout timer that is reloaded every time a node sends a token and decrements down to zero if no token for that specific node is detected. On the next image we can see both the data displayed and the generated file shown on the screen.



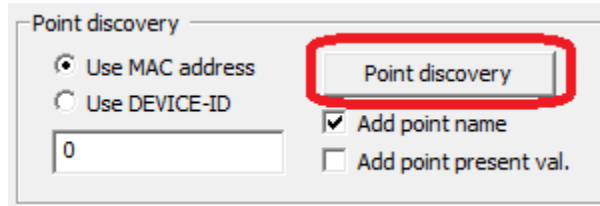
To discover devices on the BACnet/MSTP network their properties such as DEVICE, VENDOR and MODEL names use the DEVICE DISCOVERY section.



Once all devices have been discovered on the network the next section to discover the points is enabled as shown on the next page.

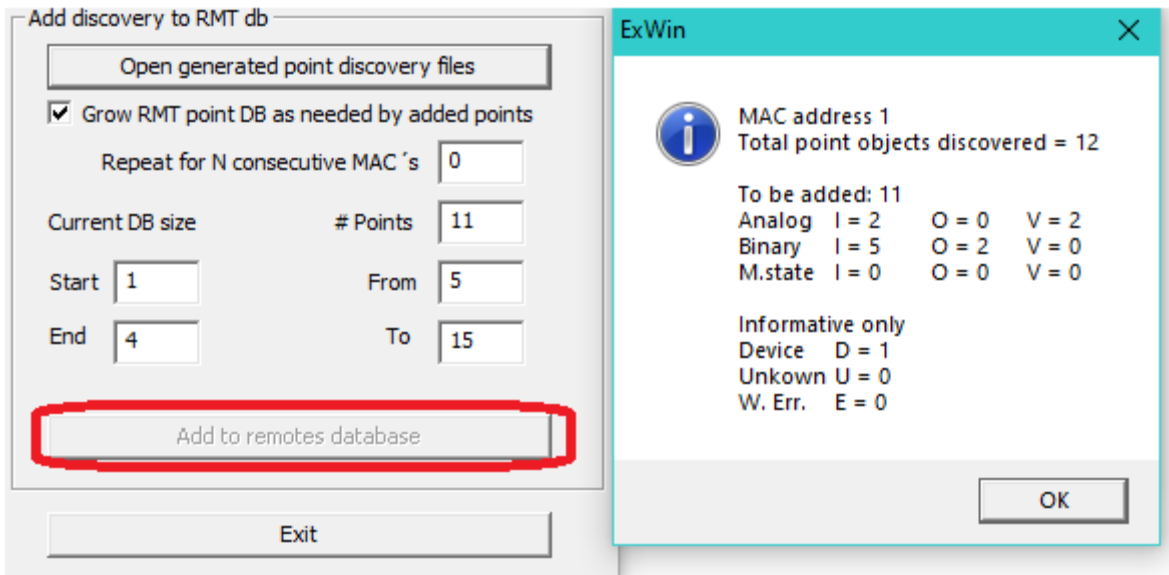
The POINT discovery is a very important part of BACnet as it allows to know how many points and of which each one of them is. It is however a complex and large process as every BACnet device discovered on the network can have up to **4,194,304 instances** of each one of the **30 different object types** defined by BACnet.

Therefore, each BACnet device can have as much as **125,829,120 objects** and as each object has in average **20 required properties plus 20 optional properties**, then doing some basic math we end up with **5,033,164,800** different individual properties that can be read or be written!



To do the point discovery either use the MAC address or the DEVICE ID that was obtained doing the DEVICE DISCOVERY on the last step, remember a file is created with this information and can always be reviewed.

To speed up discovery you can UNCHECK the ADD POINT NAME and CURRENT VALUE as only the configuration of each point is obtained, this is important when discovering a large number of objects.

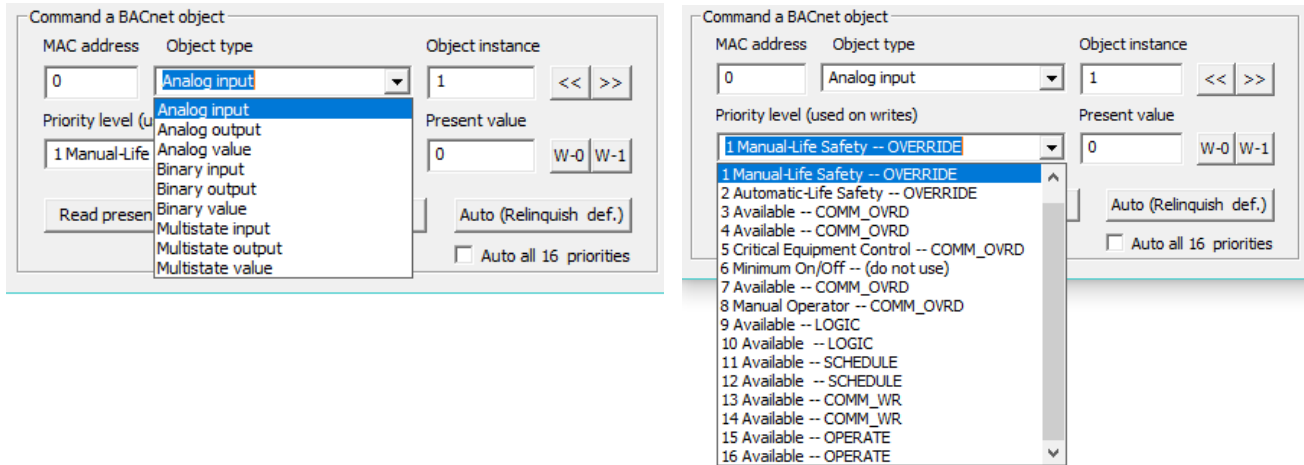


Once the discovery is finished a file for each specific MAC or DEVICE-ID is created that is used next to automatically add the remote points to the database.

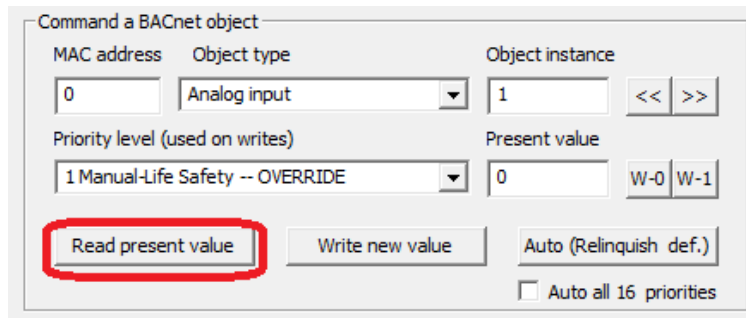
Open the selected file and if the GROW RMT POINT AS NEEDED checkbox is selected the points will be added to the remote database and it will be grown automatically.

Also, a REPEAT FOR N CONSECUTIVE MAC field is available that can be optionally set to a number between 1 thru 10 to duplicate the same remote points with automatically incremented MAC addresses. This is handy when adding similarly configured controllers on the network to the remote points database.

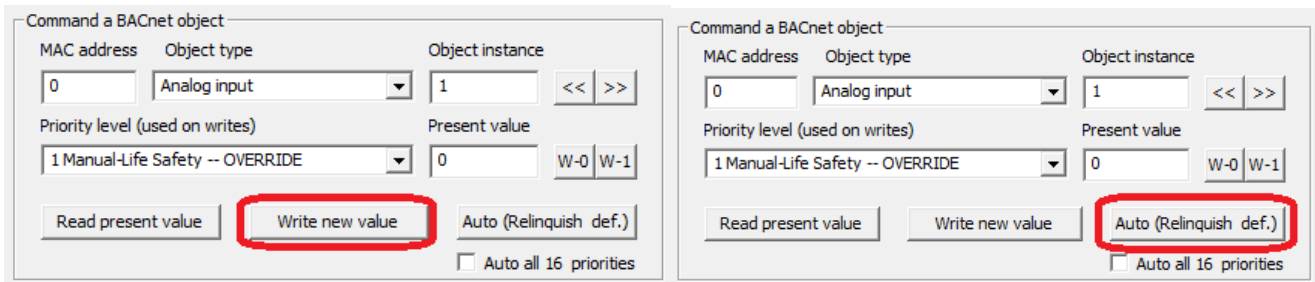
Finally In the discovery points dialog a feature to read /write any point available on the MSTP network even if it has not been mapped into the remote points is available. This is handy to test any points of any device from either OpenBAS or 3<sup>rd</sup> part that can be read or written correctly using BACnet priorities.



The **Read present value** button is used to read any object's present value.

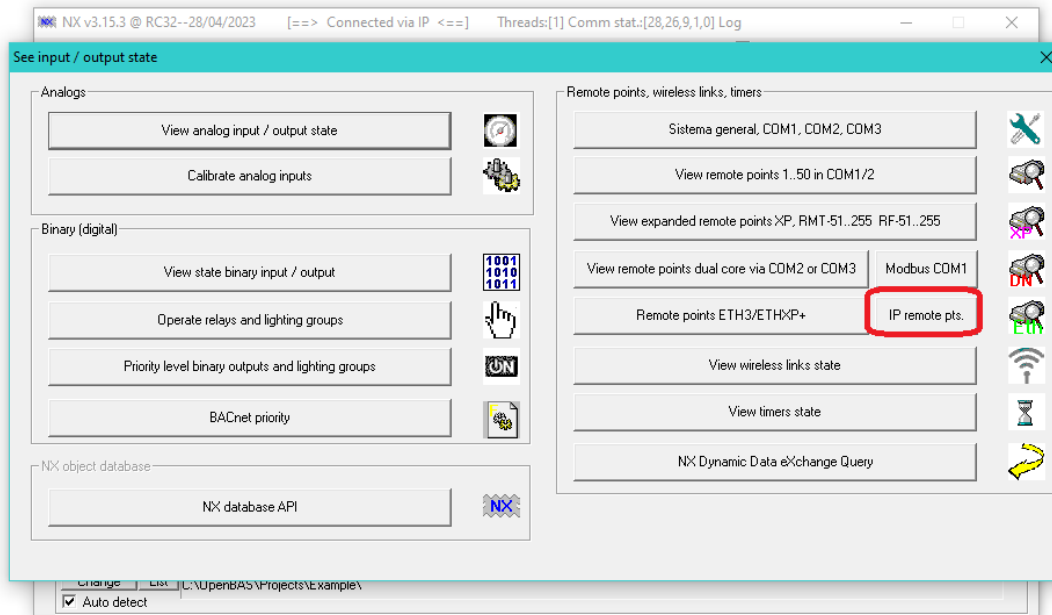


The **Write new value** and **Auto (or relinquish default)** buttons either send a priority command or release it.

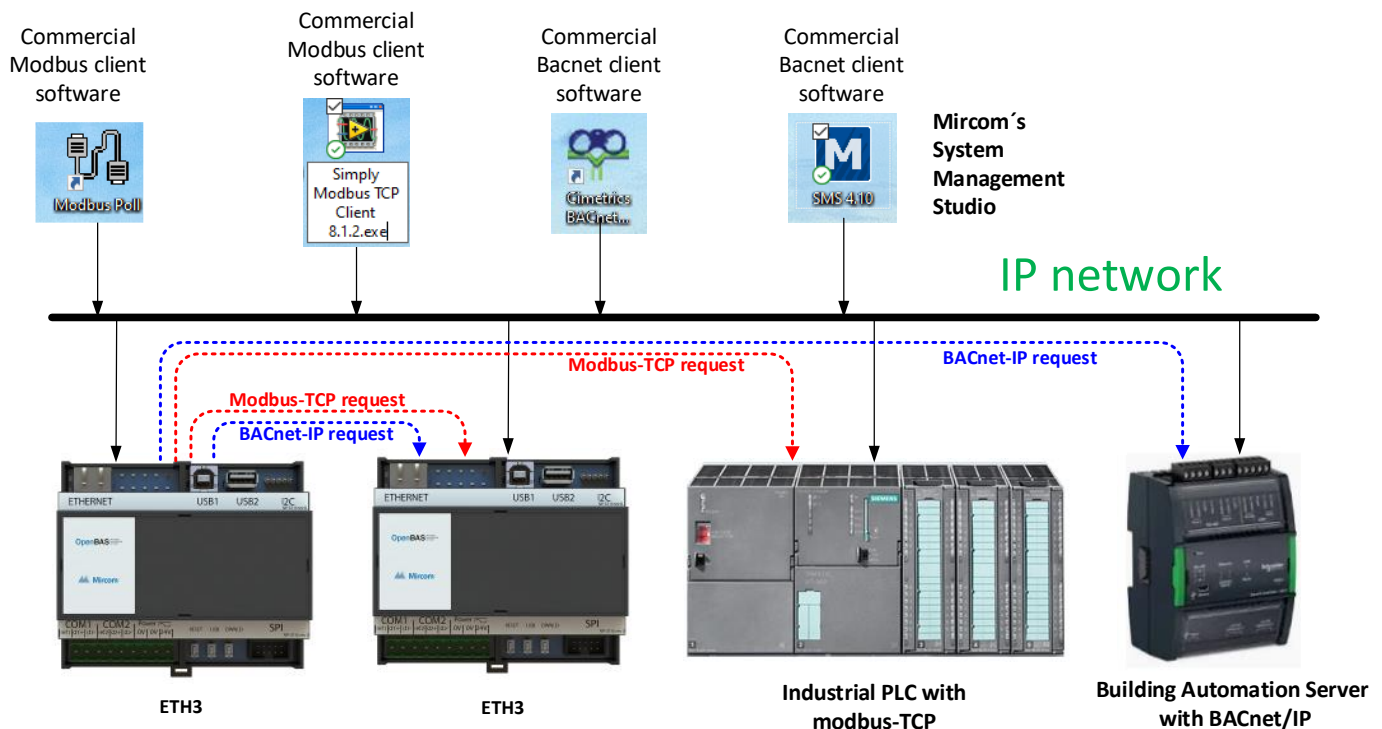


# Adding IP remote points using Modbus/TCP and BACnet/IP clients.

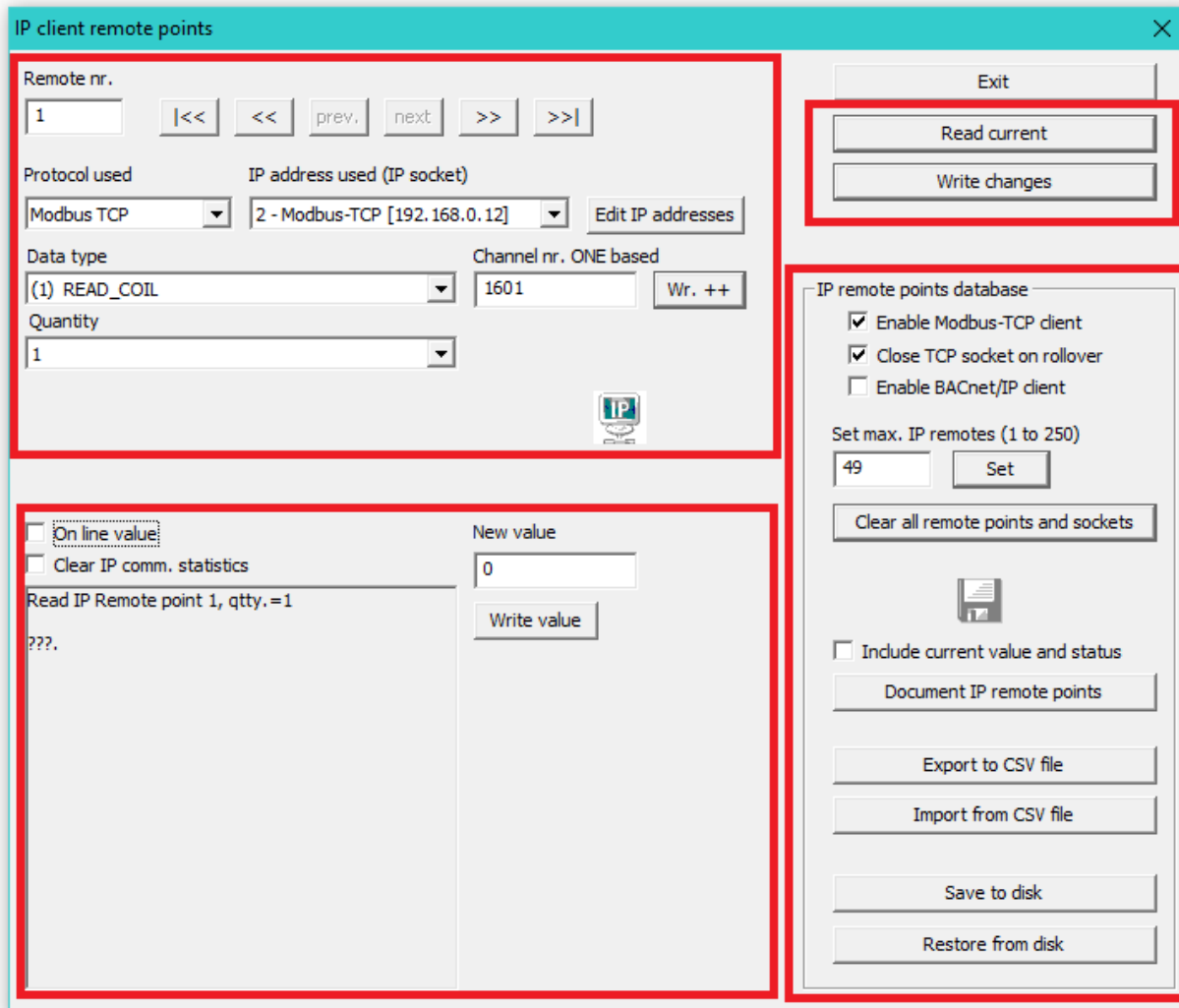
On version 3.15.3 support was added to be able to poll IP controllers over BACnet/IP and Modbus/TCP. Up to 250 remote points can be added as follows. Thus now the ETH3 besides being a BACnet/IP and Modbus-TCP server that can answer queries to other client software, can also generate its own IP request to read and write data from other controllers over IP.



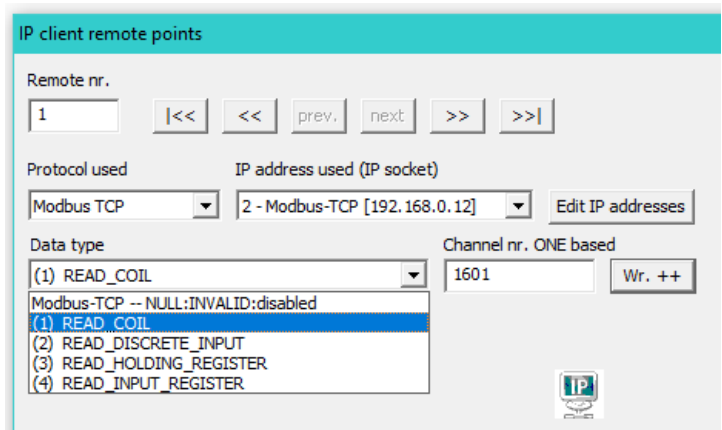
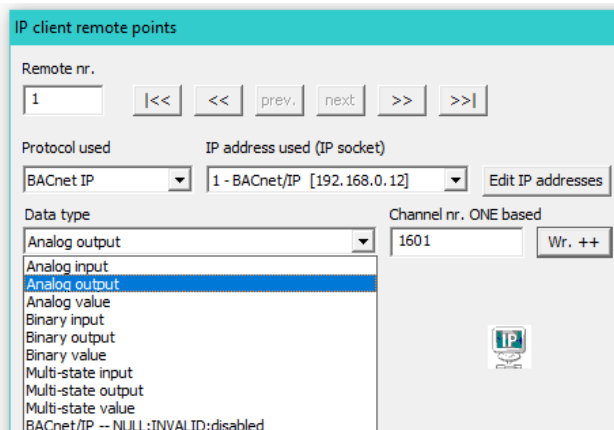
As depicted below, the ETH3 controllers can now be “**servers and clients**” simultaneously in both of the IP protocols: “**Modbus-TCP and BACnet/IP**” therefore they can respond to queries placed by other IP client software as well as place their own queries to “**read and write data**” from other IP controllers using those two mainstream protocols.



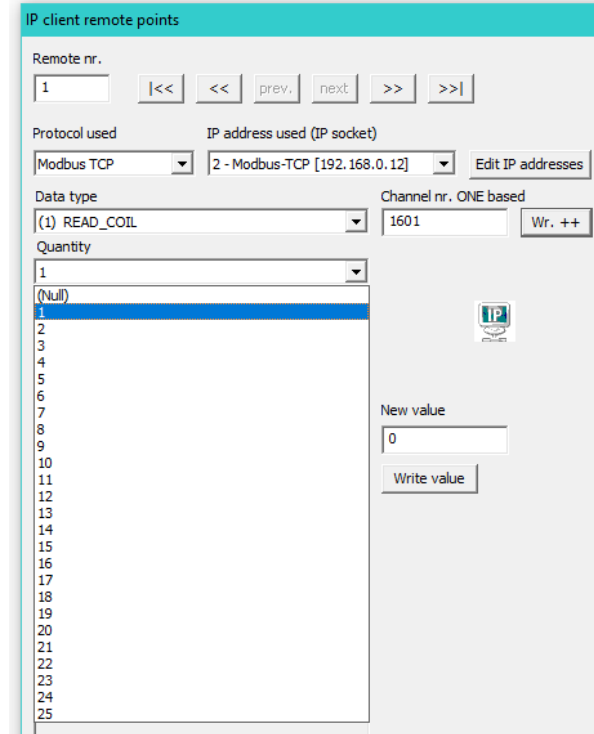
The MFC configurator tool now offers a module to configure the IP polling engine and this same APIs will be added to System Design Studio soon to ease taking advantage of this new feature in a more user friendly interface.



Up to **250 IP remote points** can be added to poll controllers from **20 different IP addresses** on any of the two before mentioned protocols supporting the following datatypes:

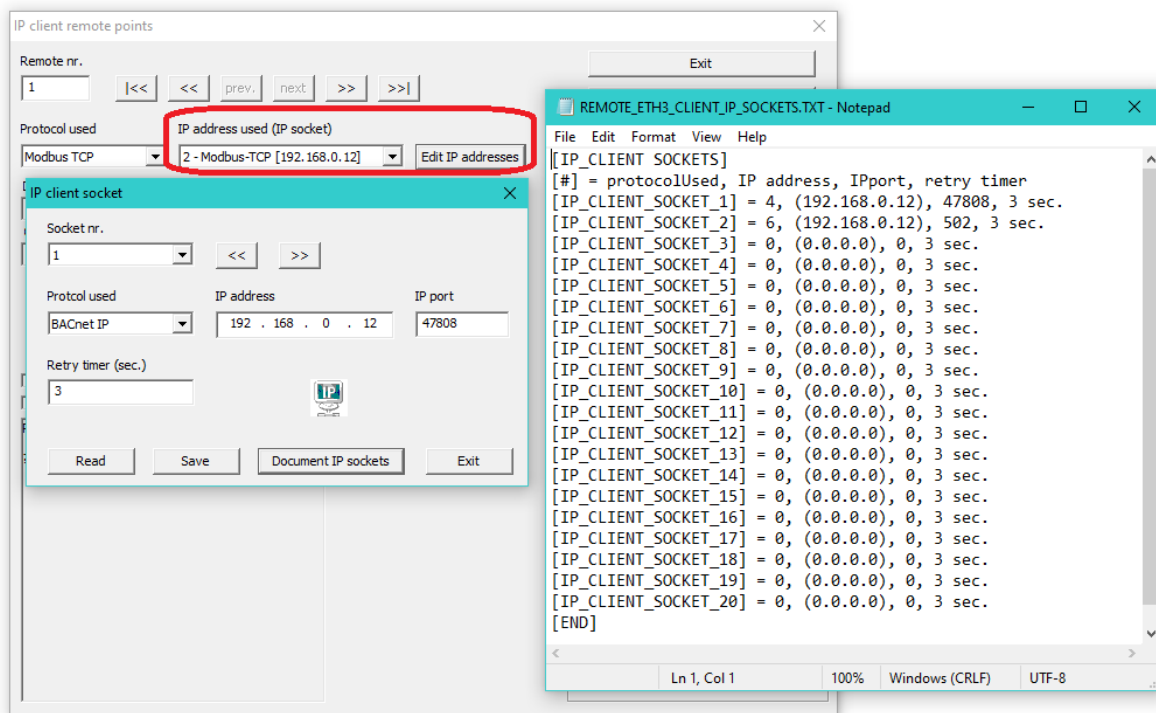


Modbus TCP has an added feature to create queries to get multiple registers from 1 and up to 25 registers in a single query to speed up the IP data polling.



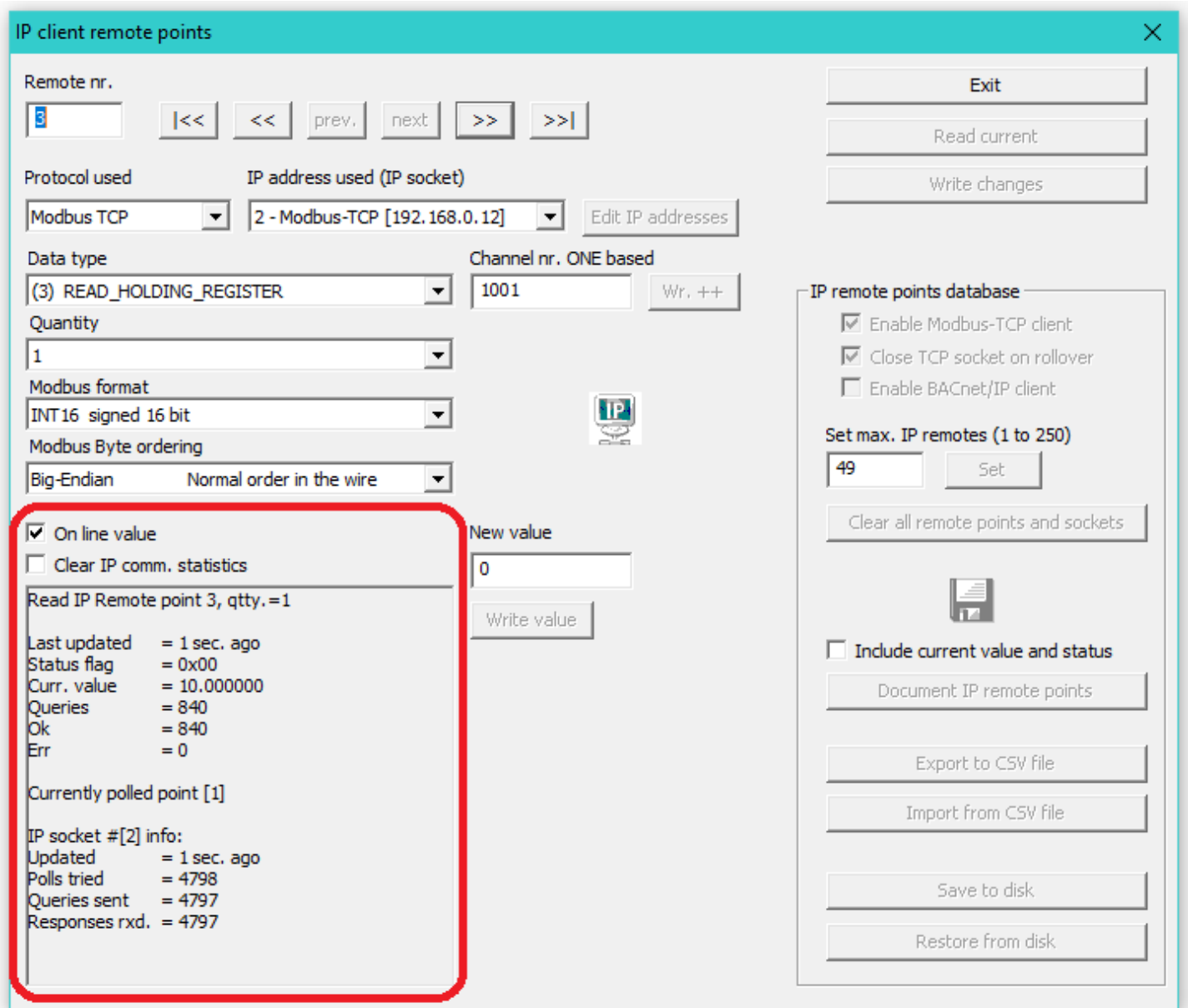
As mentioned before a combination of up to 20 different IP addresses that can be added as an IP socket that includes the IP address and the IP port to access the different IP protocols are supported.

These IP sockets can be added and modified by the user easily and saved to the ETH3 live.



A dialog to add up to **250 IP remote points** enables easily adding any of the **Modbus-TCP** or **BACnet/IP** objects to poll in any order.

It is recommended however that Modbus-TCP point that are to be obtained from a same IP address are grouped together to enable reading them sequentially using the same open socket and reduce polling latency.



The information regarding the values and status as well as the IP health is displayed live.



All this information can be presented on files, saved and restored from and to the ETH3, as well as imported and exported to CSV files for easy configuration from Excel.

```

REMOTE_ETH3_CLIENT.TXT - Notepad
File Edit Format View Help
[List of active IP addresses to use for IP remote points polling]
[IP_CLIENT_SOCKET_1] BACnet/IP, (192.168.0.12 : 47808), 3 sec.
[IP_CLIENT_SOCKET_2] Modbus-TCP, (192.168.0.12 : 502), 3 sec.
[There are 2 valid IP addresses available for use]

[List of 49 IP remote points]
[#] IP[ socketUsed ], protocolUsed, dataType-#, (Updated, stat, value), (modbusQuantity, modbusFormat, modbusEndianess)
[IP_RMT_1], IP[ 2], Modbus-TCP, Coil-1601, (0 sec, st 0x00, 1.000000), (Qty: 1, 0, 0)
[IP_RMT_2], IP[ 2], Modbus-TCP, Coil-1602, (0 sec, st 0x00, 1.000000), (Qty: 1, 0, 0)
[IP_RMT_3], IP[ 2], Modbus-TCP, Hreg-1001, (1 sec, st 0x00, 10.000000), (Qty: 1, 1, 0)
[IP_RMT_4], IP[ 2], Modbus-TCP, Hreg-1002, (1 sec, st 0x00, 20.000000), (Qty: 1, 2, 1)
[IP_RMT_5], IP[ 2], Modbus-TCP, Ireg-1003, (1 sec, st 0x00, 30.000000), (Qty: 25, 3, 2) First of 25 consecutive registers
[IP_RMT_6], IP[ 2], Modbus-TCP, Ireg-1004, (2 sec, st 0x00, 40.000000), (Qty: 1, 3, 2) Next->2
[IP_RMT_7], IP[ 2], Modbus-TCP, Ireg-1005, (2 sec, st 0x00, 50.000000), (Qty: 1, 3, 2) Next->3
[IP_RMT_8], IP[ 2], Modbus-TCP, Ireg-1006, (3 sec, st 0x00, 601.000000), (Qty: 1, 3, 2) Next->4
[IP_RMT_9], IP[ 2], Modbus-TCP, Ireg-1007, (0 sec, st 0x00, 70.000000), (Qty: 1, 3, 2) Next->5
[IP_RMT_10], IP[ 2], Modbus-TCP, Ireg-1008, (1 sec, st 0x00, 80.000000), (Qty: 1, 3, 2) Next->6
[IP_RMT_11], IP[ 2], Modbus-TCP, Ireg-1009, (1 sec, st 0x00, 90.000000), (Qty: 1, 3, 2) Next->7
[IP_RMT_12], IP[ 2], Modbus-TCP, Ireg-1010, (2 sec, st 0x00, 100.000000), (Qty: 1, 3, 2) Next->8
[IP_RMT_13], IP[ 2], Modbus-TCP, Ireg-1011, (2 sec, st 0x00, 110.000000), (Qty: 1, 3, 2) Next->9
[IP_RMT_14], IP[ 2], Modbus-TCP, Ireg-1012, (2 sec, st 0x00, 120.000000), (Qty: 1, 3, 2) Next->10
[IP_RMT_15], IP[ 2], Modbus-TCP, Ireg-1013, (3 sec, st 0x00, 130.000000), (Qty: 1, 3, 2) Next->11
[IP_RMT_16], IP[ 2], Modbus-TCP, Ireg-1014, (0 sec, st 0x00, 140.000000), (Qty: 1, 3, 2) Next->12
[IP_RMT_17], IP[ 2], Modbus-TCP, Ireg-1015, (1 sec, st 0x00, 150.000000), (Qty: 1, 3, 2) Next->13
[IP_RMT_18], IP[ 2], Modbus-TCP, Ireg-1016, (1 sec, st 0x00, 160.000000), (Qty: 1, 3, 2) Next->14
[IP_RMT_19], IP[ 2], Modbus-TCP, Ireg-1017, (2 sec, st 0x00, 170.000000), (Qty: 1, 3, 2) Next->15
[IP_RMT_20], IP[ 2], Modbus-TCP, Ireg-1018, (2 sec, st 0x00, 180.000000), (Qty: 1, 3, 2) Next->16
[IP_RMT_21], IP[ 2], Modbus-TCP, Ireg-1019, (2 sec, st 0x00, 190.000000), (Qty: 1, 3, 2) Next->17
[IP_RMT_22], IP[ 2], Modbus-TCP, Ireg-1020, (3 sec, st 0x00, 200.000000), (Qty: 1, 3, 2) Next->18
[IP_RMT_23], IP[ 2], Modbus-TCP, Ireg-1021, (3 sec, st 0x00, 0.000000), (Qty: 1, 3, 2) Next->19
[IP_RMT_24], IP[ 2], Modbus-TCP, Ireg-1022, (1 sec, st 0x00, 0.000000), (Qty: 1, 3, 2) Next->20
[IP_RMT_25], IP[ 2], Modbus-TCP, Ireg-1023, (1 sec, st 0x00, 0.000000), (Qty: 1, 3, 2) Next->21
[IP_RMT_26], IP[ 2], Modbus-TCP, Ireg-1024, (1 sec, st 0x00, 0.000000), (Qty: 1, 3, 2) Next->22
[IP_RMT_27], IP[ 2], Modbus-TCP, Ireg-1025, (2 sec, st 0x00, 0.000000), (Qty: 1, 3, 2) Next->23
[IP_RMT_28], IP[ 2], Modbus-TCP, Ireg-1026, (2 sec, st 0x00, 0.000000), (Qty: 1, 3, 2) Next->24
[IP_RMT_29], IP[ 2], Modbus-TCP, Ireg-1027, (3 sec, st 0x00, 0.000000), (Qty: 1, 3, 2) Next->25
[IP_RMT_30], IP[ 2], Modbus-TCP, Coil-1601, (3 sec, st 0x00, 1.000000), (Qty: 20, 0, 0) First of 20 consecutive registers
[IP_RMT_31], IP[ 2], Modbus-TCP, Coil-1602, (4 sec, st 0x00, 1.000000), (Qty: 1, 0, 0) Next->2
[IP_RMT_32], IP[ 2], Modbus-TCP, Coil-1603, (0 sec, st 0x00, 1.000000), (Qty: 1, 0, 0) Next->3
  
```

Excel spreadsheet titled "REMOTE\_IP.CSV" showing configuration for IP remote points. The spreadsheet includes a header row with columns A through AD, and a data table with columns for ID, Protocol, IP Address, Port, and Timer. The data table lists 20 remote points, each with a unique ID, protocol (BACNET or MODBUS), IP address, port, and timer value. The spreadsheet also includes a detailed comment section explaining the configuration parameters and their values.

ID	Protocol	IP Address	Port	Timer
1	BACNET	192.168.0.12	47808	3
2	MODBUS	192.168.0.12	502	3
3	NONE	0.0.0.0	0	3
4	NONE	0.0.0.0	0	3
5	NONE	0.0.0.0	0	3
6	NONE	0.0.0.0	0	3
7	NONE	0.0.0.0	0	3
8	NONE	0.0.0.0	0	3
9	NONE	0.0.0.0	0	3
10	NONE	0.0.0.0	0	3
11	NONE	0.0.0.0	0	3
12	NONE	0.0.0.0	0	3
13	NONE	0.0.0.0	0	3
14	NONE	0.0.0.0	0	3
15	NONE	0.0.0.0	0	3
16	NONE	0.0.0.0	0	3
17	NONE	0.0.0.0	0	3
18	NONE	0.0.0.0	0	3
19	NONE	0.0.0.0	0	3
20	NONE	0.0.0.0	0	3

Comments section:

```

// Don't use IP addresses [0.0.0.0] or [255.255.255.255] except for the NONE/MULL protocol
// Protocol can be any of: MODBUS or BACNET or NONE or NULL
// Retry time must be between 3 and 250 seconds
// IP_SOCK_PROTOCOL, IP_ADDR, IP_PORT, RETRY_TIMER
// -----
// [START]
1 BACNET [192.168.0.12] 47808 3
2 MODBUS [192.168.0.12] 502 3
3 NONE [0.0.0.0] 0 3
4 NONE [0.0.0.0] 0 3
5 NONE [0.0.0.0] 0 3
6 NONE [0.0.0.0] 0 3
7 NONE [0.0.0.0] 0 3
8 NONE [0.0.0.0] 0 3
9 NONE [0.0.0.0] 0 3
10 NONE [0.0.0.0] 0 3
11 NONE [0.0.0.0] 0 3
12 NONE [0.0.0.0] 0 3
13 NONE [0.0.0.0] 0 3
14 NONE [0.0.0.0] 0 3
15 NONE [0.0.0.0] 0 3
16 NONE [0.0.0.0] 0 3
17 NONE [0.0.0.0] 0 3
18 NONE [0.0.0.0] 0 3
19 NONE [0.0.0.0] 0 3
20 NONE [0.0.0.0] 0 3
// [END]
// Now after the twenty sockets define the number of remote points that exist and will be polled
// k can be from 1 and up to 250
// The PROTOCOL can be any of: MODBUS or BACNET or NONE or NULL
// The IP_SOCKET_NR defines which one of the 20 IP addresses defined above will be used to poll the remote point
// k should be from 1 and up to 20 just make sure that the protocol matches the one defined in the IP socket
// For the DATA_TYPE depending on the protocol can be:
// For BACNET use any one of: AI AD AU BI BD BV MSI MSO MSV
// For MODBUS use any one of: Coil Dreg Hreg Ireg
// For CHANNEL_NR use the following range
// For BACNET use from 1 and up to 4194303
// For MODBUS use from 1 and up to 85335
// For QUANTITY use 1 as creating modbus multiple read is not supported by the CSV import feature yet
// For MULTIPLE_FLAG use 0
// For MODBUS_FORMAT use any of: UNIT16 INT16 INT32 INT64 UNIT64 FLOAT32 DOUBLE64
// For MODBUS_ENDIANESS use any of: AB_CD CD_AB BA_CD DC_BA for Big-Endian Little-endian Big-endian-byte-swapped Little-endian-byte-swapped
// These last three configuration parameters can be not included if the default values are desired
// IP_RM1_PROTOCOL, IP_SOCK_DATA_T, CHANNEL, QUANTIT, MULTIFL, MODBUS, MODBUS_ENDIANESS
// -----
// [Number of IP remote points]
49
// [START]
1 MODBUS 2 Coil 1601 1 0 UNIT16 AB_CD
2 MODBUS 2 Coil 1602 1 0 UNIT16 AB_CD
  
```

---

## Using the PLC to create powerful logic to interoperate between slave devices.

The ETH3 supports two PLC's with 400 instructions each. The operation is similar to the operation of NX controllers with the difference that the ETH3 has a dual database: It's own and if an NX is connected over the SPI bus it can also use what we will call the remote database in the PLC's instructions.

---

Use the schedules.

A

---

Add the i2c eight-channel current metering front mounted expander.

A

---

Add the i2c eight-channel binary input monitor front mounted expander.

A

---

Add the i2c seven-channel Input/Output front mounted expander.

A

## Using the e-mail generator.

Sometimes customers looking for special functions in small panels, such send emails or SMS messages or web pages, but the Mircom small panel has not included these special functions.

OpenBAS series is an option to give some of these special features to these small panels.

The OpenBAS-NWK-ETH3 comes with two RS-485 ports that can be configured for different protocols such ASCII, Modbus, BACnet, OPTOMUX 22, etc.

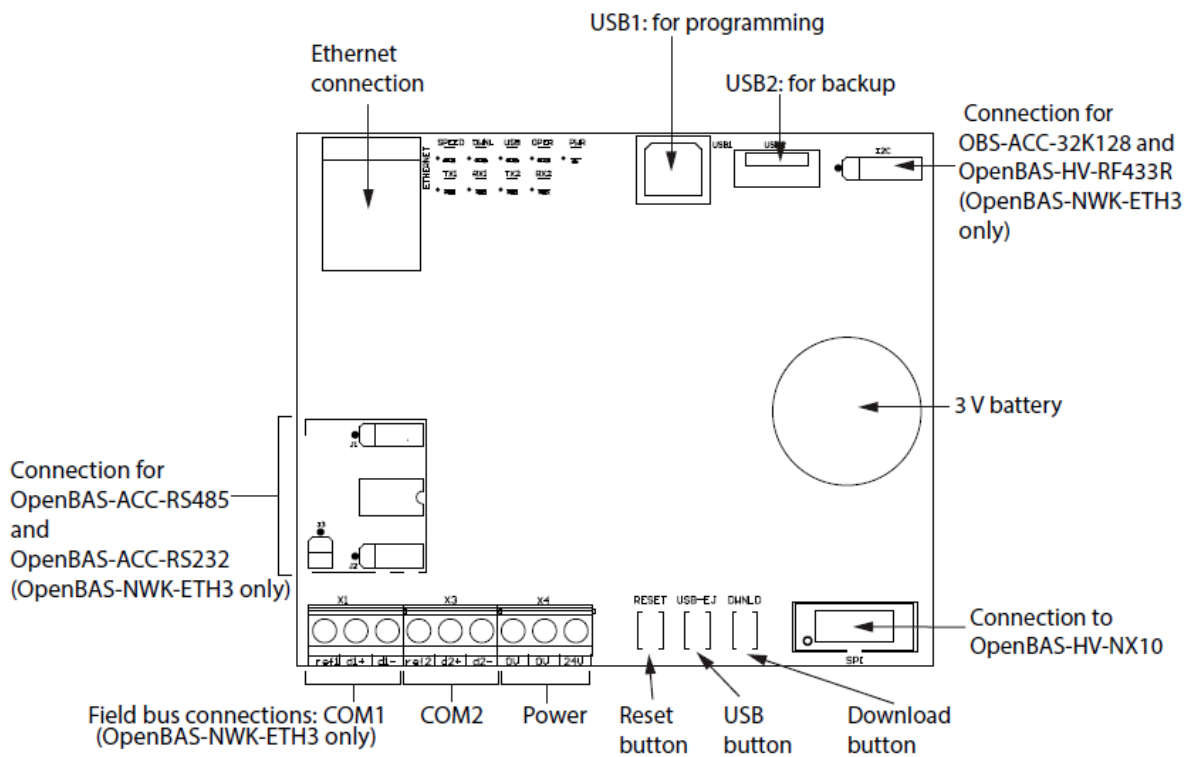
In this case, in this document we are going to describe how to use the OpenBAS-NWK-ETH3 as email server for the FX-3500 panel.

### What do you need to connect the FX-3500 to the OpenBAS-NWK-ETH3?

1. FX-3500 panel
2. OpenBAS-NWK-ETH3
3. OpenBAS-ACC.RS232 (or any 3<sup>rd</sup> party external RS232 to RS485 converter)

### Replacing the RS-485 driver for RS-232 driver on OpenBAS controller

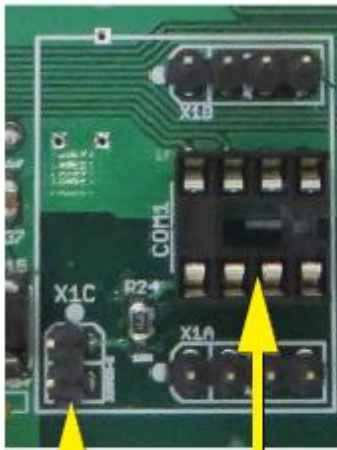
1. Localize the COM ports on the controller board



By default, COM1 and COM2 are configured as RS-485. COM1 can be changed to RS-232 or optically isolated RS-485 by installing the OpenBAS-ACC-RS232 or OpenBAS-ACC-RS485 converter.

## To install a communication converter

1. Open the jumper.
2. Remove the factory-installed RS-485 module.
3. Install the communication converter.



Jumper  
open

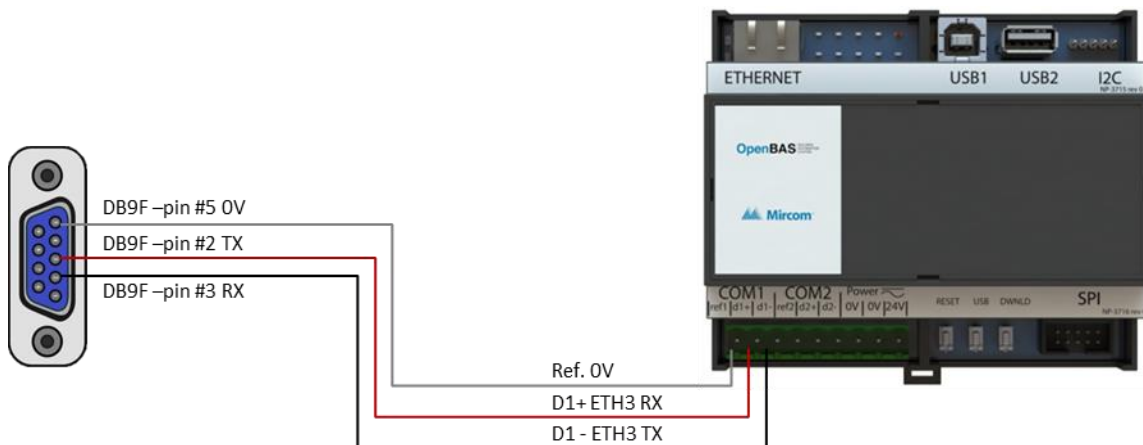
Factory-installed  
RS-485 module  
removed



OpenBAS-ACC-RS232

## How to Connect the FX-3500 panel to the OpenBAS-NWK-ETH3?

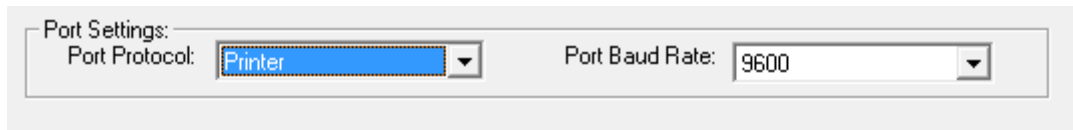
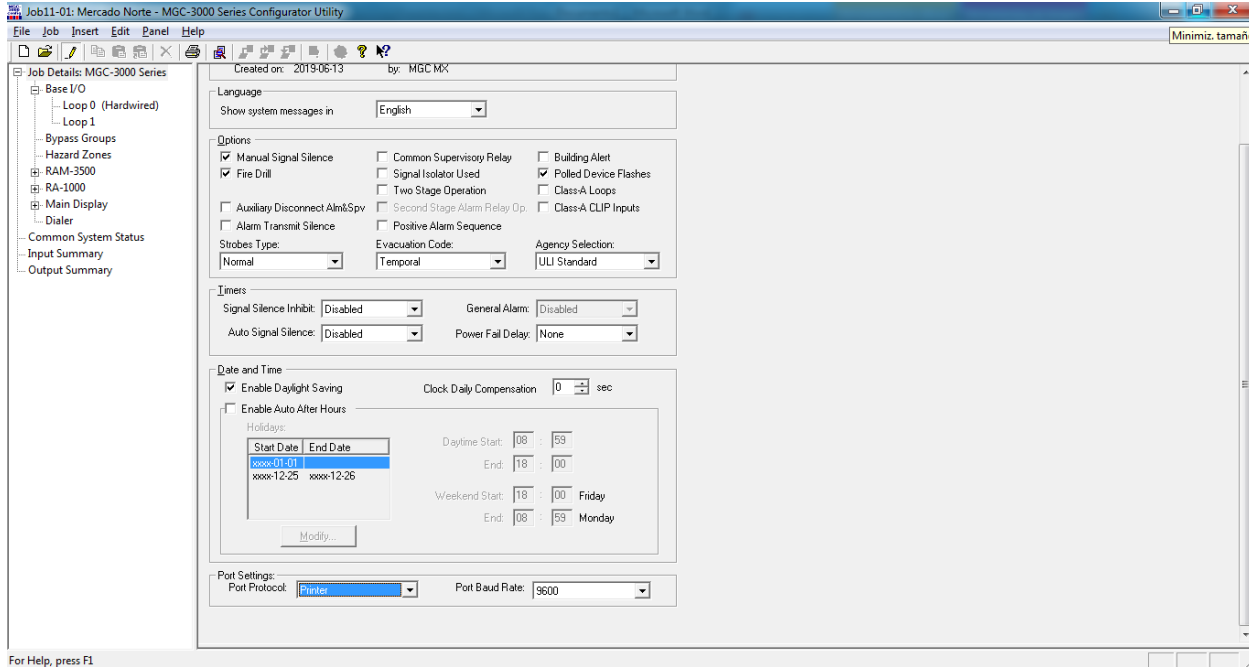
Localize the printer port on the panel, use the pin 2, 3 and 5 on the DB9F connector and wire direct to the Port COM1 on the ETH3 controller as shown on the diagram bellow.





## Setting up the printer port on FX-3500 Panel

Open the MGC-3000 Series Configuration Utility and active the printer port with the default values on the Port Settings section as shown below.



## Setting up the OpenBAS-NWK-ETH3

The first you must do, is configure the IP address of the ETH3 controller

To do it there are four different ways as explained in previous sections

1. Use the IPCONFIG.INI file of the USB memory inserted in the ETH3
2. Use the configuration tool if you have a NX controller connected in the SPI bus of the ETH3. You can see the current IP address in either the LCD display if the NX has one or in the configuration software.
3. Use the web page interface if you can already access the IP
4. Reset the ETH3 controller to the factory defaults so it uses the DHCP and uses a default IP address to connect to it and change the IP address to a new one.

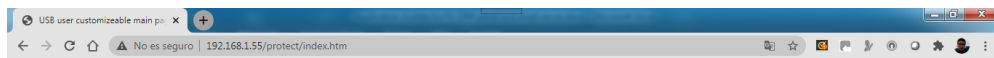
Open the web server page using the assigned IP address.



Click on Go to main page link if login credentials are requested use the data below.

- User: **admin**
- Password: **mircom**

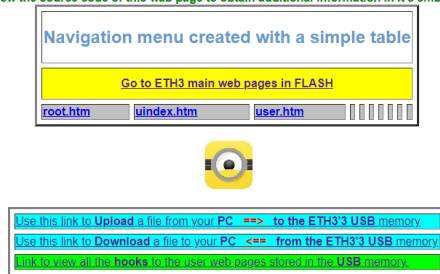
If you have a USB inserted on the ETH3 you will be redirected to the page shown below



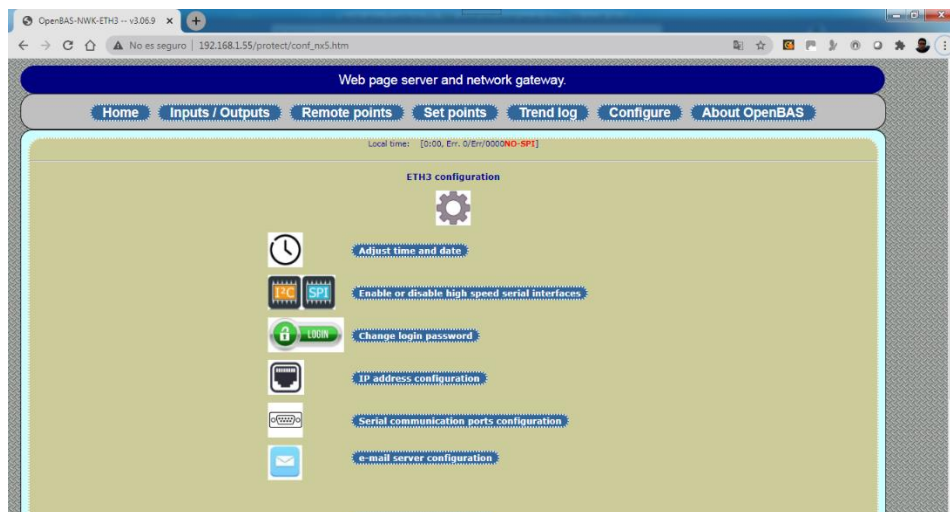
This is the home web page **root.htm** when the USB is inserted.

A sample menu that was created by the links of the hook files that are stored in the file: **web\_idx.ini** also in the USB.

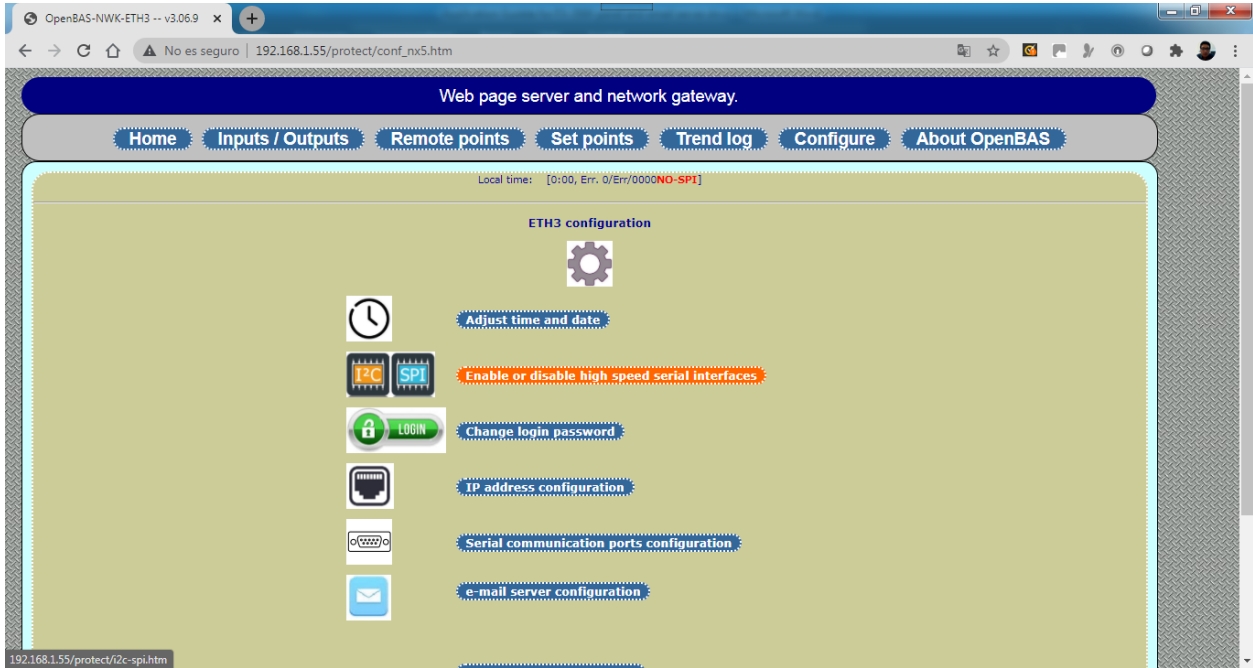
Right click to view the source code of this web page to obtain additional information in its embedded comments.



Click on the Go to main page option. Then, click on Configure menu.



As the ETH3 controller will work without NX controller you should disable the SPI bus. Click on Enable or disable high speed serial interfaces.



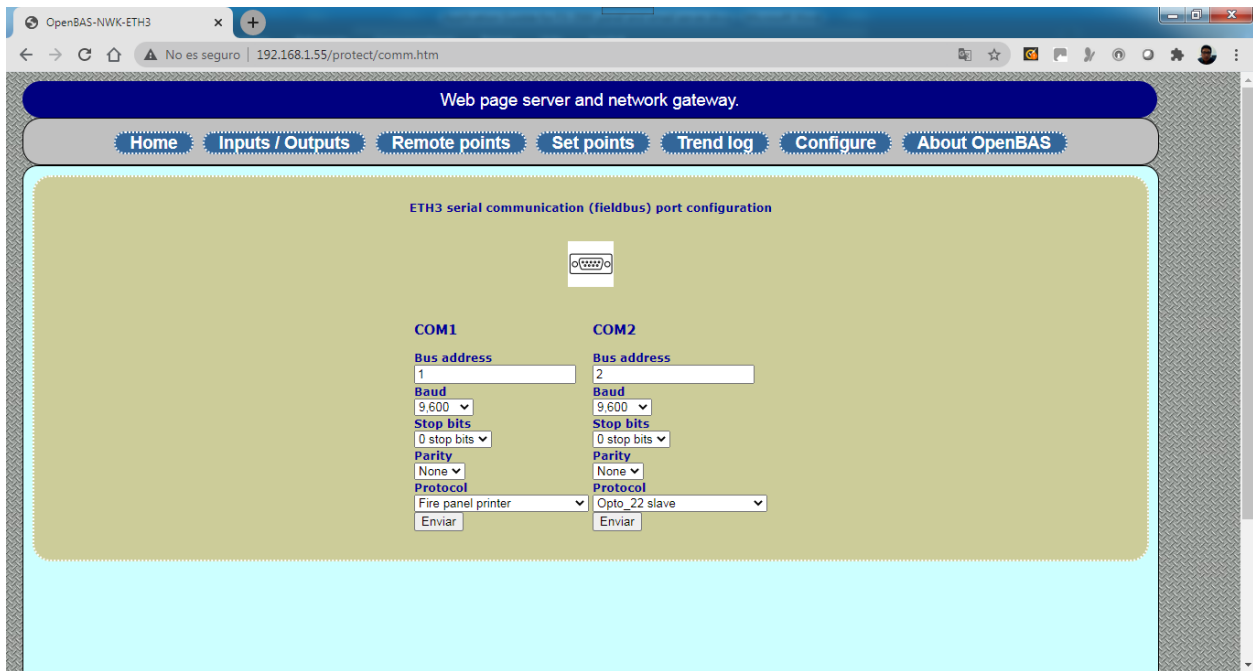
Click on the DISABLE SPI BUS button.



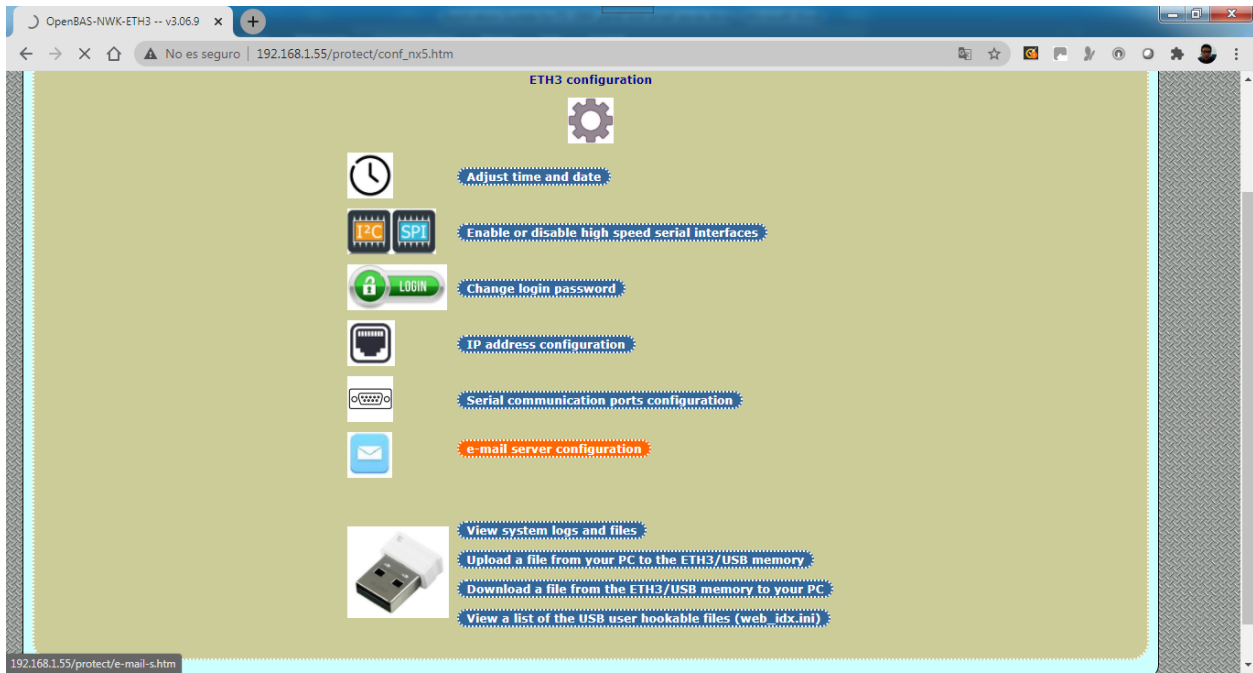
Return to the Configure page and select the Serial communication ports configuration.



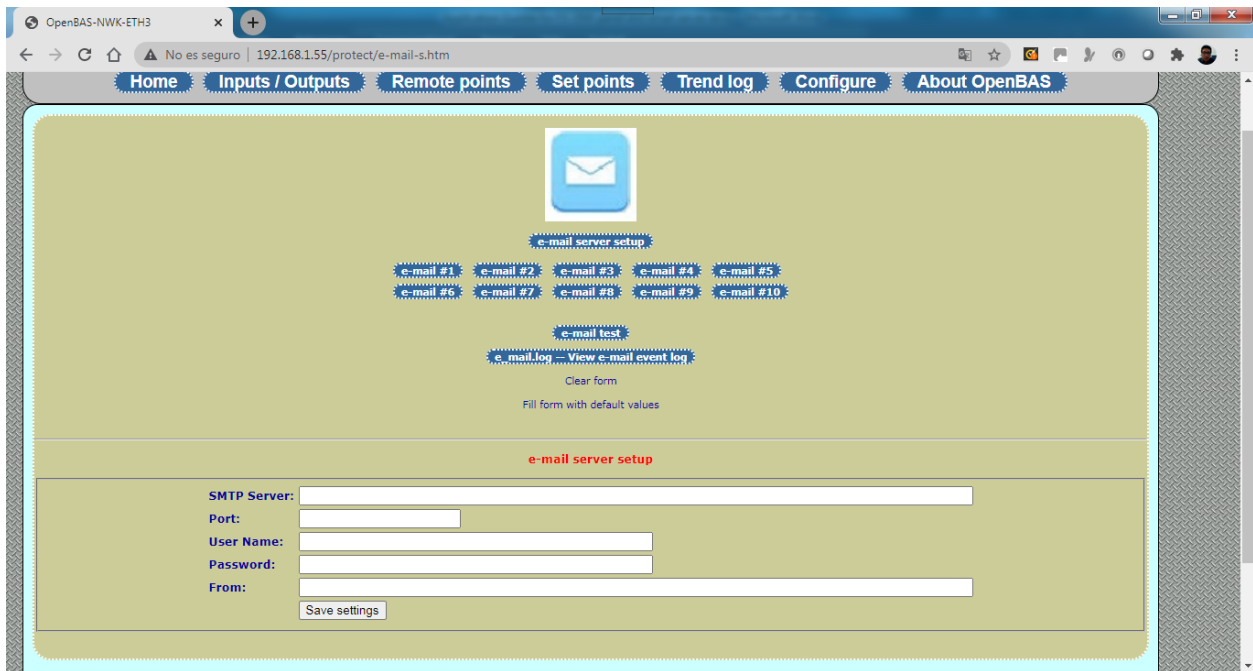
On the Serial communication ports configuration page, match the data on the COM1 port with the Fire Alarm Printer port configuration, and select the Fire panel printer option on the Protocol configuration, then click on Sent button.



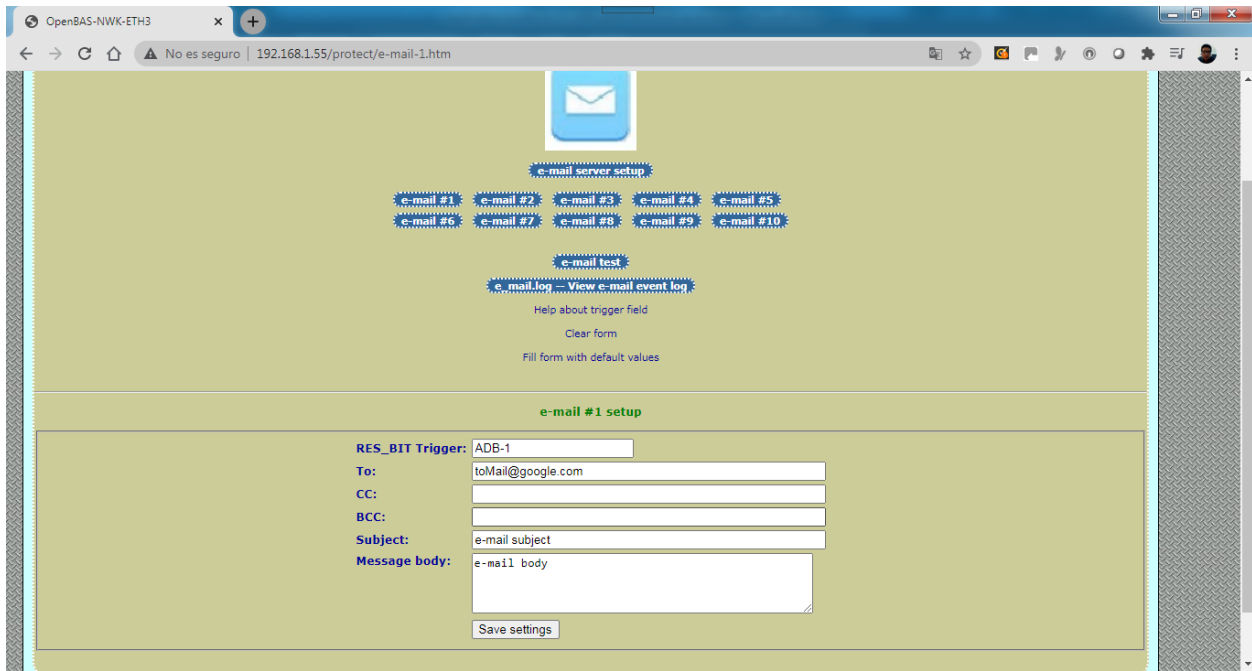
Return to the configuration page and click on the e-mail server configuration menu.



Fill the information requested according with your email server and click on the Save settings button.



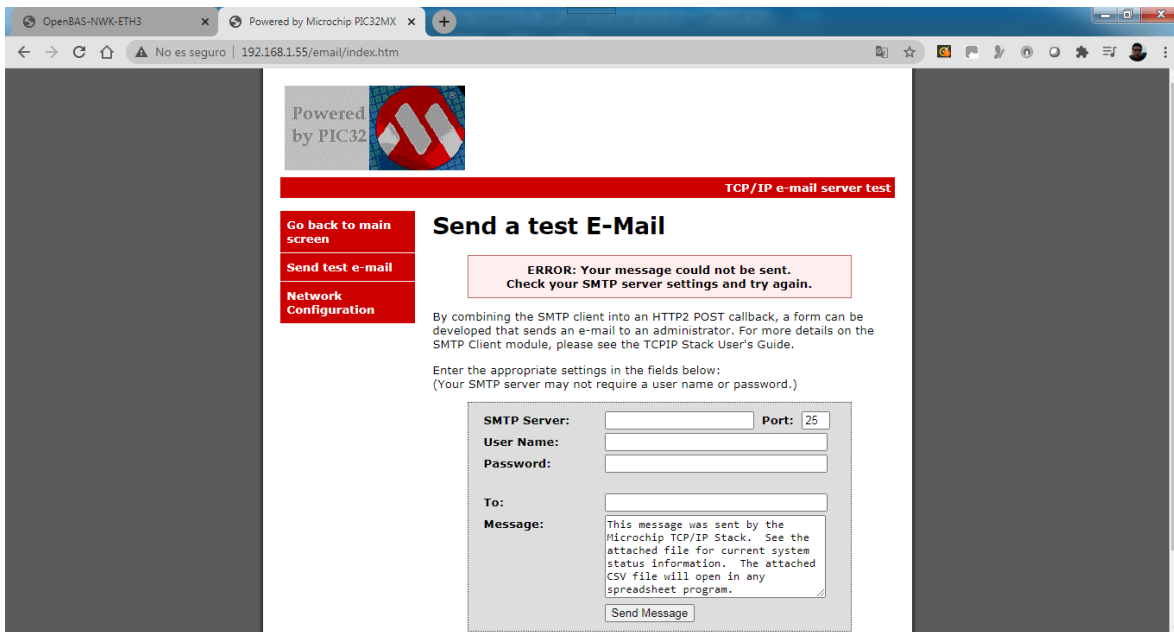
As you are using the COM1 port, you need configure the e-mail #1, the parameter configured on the RES\_BIT Trigger: must be configured as ADB-1, the other parameters should configure with the data you want to show on the e-mail, and then click on the save settings button.



On the e-mail server setup, you can use the option fill form with default values, for test propose is available an e-mail server, after testing you can change the configuration with the custom values.

#### How to test email server

1. Open in a new tab the e-mail test web page, fill with the e-mail server parameters and click on send message button. When click on the send message button you should receive an e-mail and the message You e-mail has been send. If something is wrong, you will receive an error message.

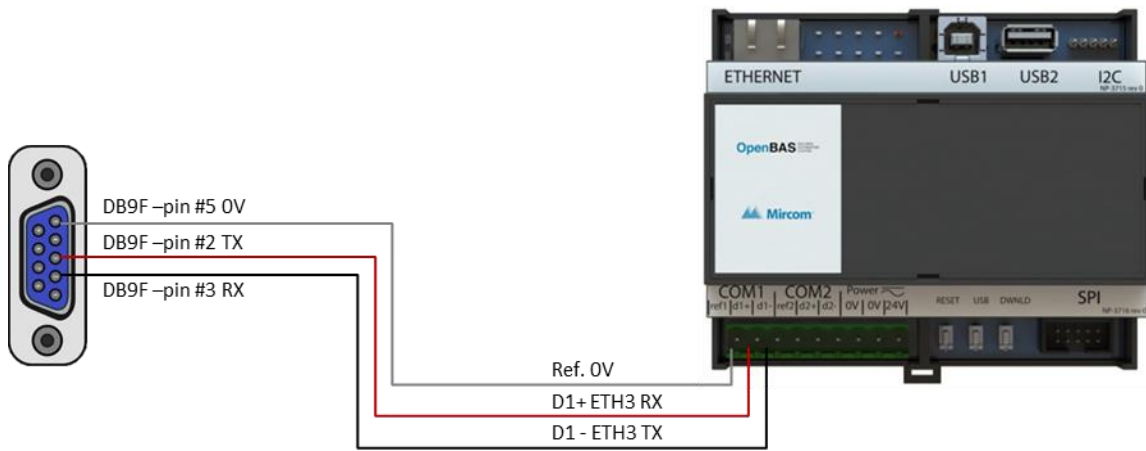


## Connecting a Mircom fire panel using the printer port to convert to BACnet and Modbus.

Follow the same connection procedure explained on the previous section to connect the fire panel's serial port to the ETH3.

### How to Connect the FX-3500 panel to the OpenBAS-NWK-ETH3?

Localize the printer port on the panel, use the pin 2, 3 and 5 on the DB9 connector and wire direct to the Port COM1 on the ETH3 controller as shown on the diagram bellow.



After connection is made use the MFC tool configurator and access the ETH3's remote point configuration to access the COM1 and COM2 configuration dialog.

When clicking on either the VIEW STATUS or the CONFIGURE main screen buttons you will find the following UI element:



When clicking it the dialog for setting the ETH3's COM1 and OCM2 remote points will pop up and click in the button labeled: CONIGURATION OF COM 1 AND COM2 PORTS OF THE ETH3.

This opens a dialog for configuring the COM1 and COM2 field buses shown on the next page.

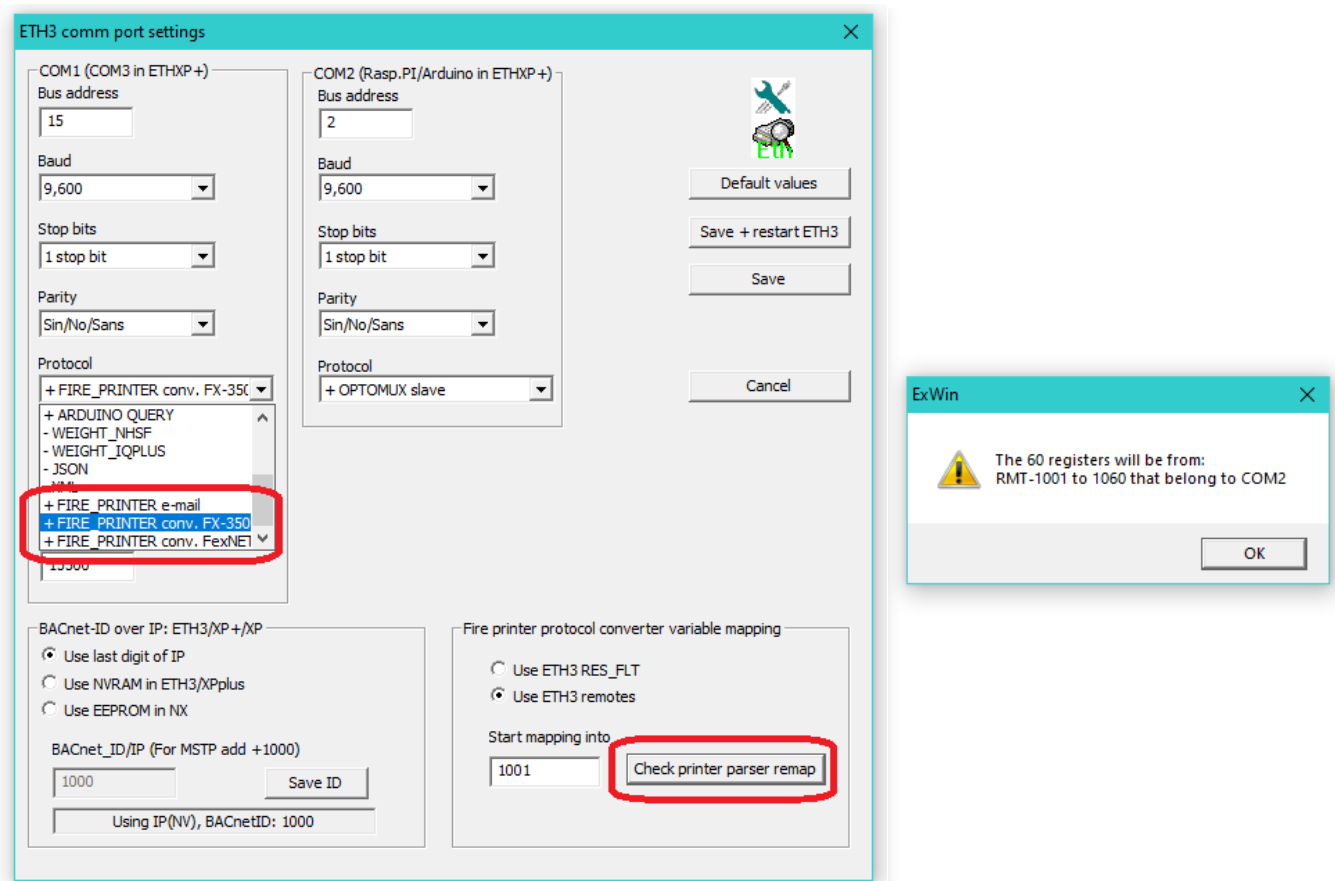
Here you select one either od COM1 or COM2 to be serial port that will attach to the fire panel. Note that there are currently three protocols used for fire panels:

- **Fire printer e-mail** Send's e-mails to different recipients on fire panel events.
- **Fire printer converter FX3500** Convert FX3500 fire panel's printer output to Modbus and BACnet.
- **Fire printer converter FlexNet** Convert FlexNet fire panel's printer output to Modbus and BACnet.

Select any of the last two to use the ETH3 as a protocol converter. Next on the section named FIRE PRINTER PROTOCOL CONVERTER VARIABLE MAPPING select where the fire panel reports will be remapped. There are two options:

- **Use RES\_FLOATS** Uses 60 contiguous RES\_FLT registers to map the fire panel events.
- **Use ETH3 remotes** Uses 60 contiguous Remote Point registers to map the fire panel events.

Finally, you must enter the START or the first of the 60-register used that will be used to do the conversion. The CHECK PRINTER PARSER REMAP verifies the ranges and provides a brief description of this mapping.



So how exactly this converter works out? As the fire panel sends textual information to the printer, these messages are intercepted and parsed by the ETH3 and stored into registers of the ETH3 database, where they can be read from any 3<sup>rd</sup> party software. These 60 registers can be read simultaneously over any of the following protocols:

- **BACnet/IP**
- **BACnet/MSTP**
- **Modbus-TCP**
- **Modbus-RTU**
- **Optomux-IP**
- **Optomux-RS485**
- **SQL over Telnet-IP**
- **SQL over RS485**
- **N2-Open**



The following table shows a relationship between the fire panel events and the block of 60 registers mapped into the database.

First comes ten register containing counters of how many events have been sent by the fire panel since the past power on of the ETH3.

Register nr.	Description	Notes
1	Total event count	The total number of events sent by the fire panel since the last ETH3 power on.
2	Loop event count	Events from loop devices since the last ETH3 power on.
3	System event count	System events since the last ETH3 power on.
4	Exception event count	Exception events, that either contained an error or could not be parsed.
5	Non data event counts	Lines containing non parseable data, usually sent after an event by the fire panel.
6	Active events	Number of events stored in the buffer, 0 and up to 10 events are stored as an array below.
7	Free	Reserved for future use
8	Free	Reserved for future use
9	Free	Reserved for future use
10	Free	Reserved for future use

Of relevance is the 6<sup>th</sup> register that shows how many events are stored in the event queue. The last ten events are stored in the next 50 registers. Each 5 registers contain information of the event and are encoded as follow:

Register nr.	Description	Notes
11	Time stamp or Node	Time Stamp HH:MM or Node 1..63 in FlexNet
12	Loop or CPU	Loop 1..3 or CPU 0..7 in FlexNet
13	Address or Zone	Address 1..159/201..359 or Zn: 1..400 in FlexNet
14	Type	Type Sensor = 1, Module = 2, External = 3
15	Event	Loop event // Alarm activated = 1 // Alarm restored = 2 // Trouble = 3 // Trouble restored = 4 // Missing device = 5 // Open circuit trb = 6 // Other unknown = 255  System event // For FX3500 codes 101..199 // For FlexNet codes 201..254

The most recent event is always the first one, and as events arrive, they are pushed up in 5 register increments, thus registers 11. thru 15 are always the newest, then follows 16 thru 21 all the way to the last event stored at the registers 55 thru 60. Only the last 10 events are stored in the registers and pushed out as new ones arrive as depicted below.

1 thru 10	11 - 15	16 - 20	21 - 25	26 - 30	31 - 35	36 - 40	41 - 45	45 - 50	51 - 55	56 - 60	
Counters	Most recent event	Event + 2	Event + 3	Event + 4	Event + 5	Event + 6	Event + 7	Event + 8	Event + 9	Oldest Event + 10	Oldest event(s) lost

Even while old events are pushed out of the register stack, if a USB memory is installed, they are always kept there. On the next page it is shown how to obtain the whole list using a web browser.

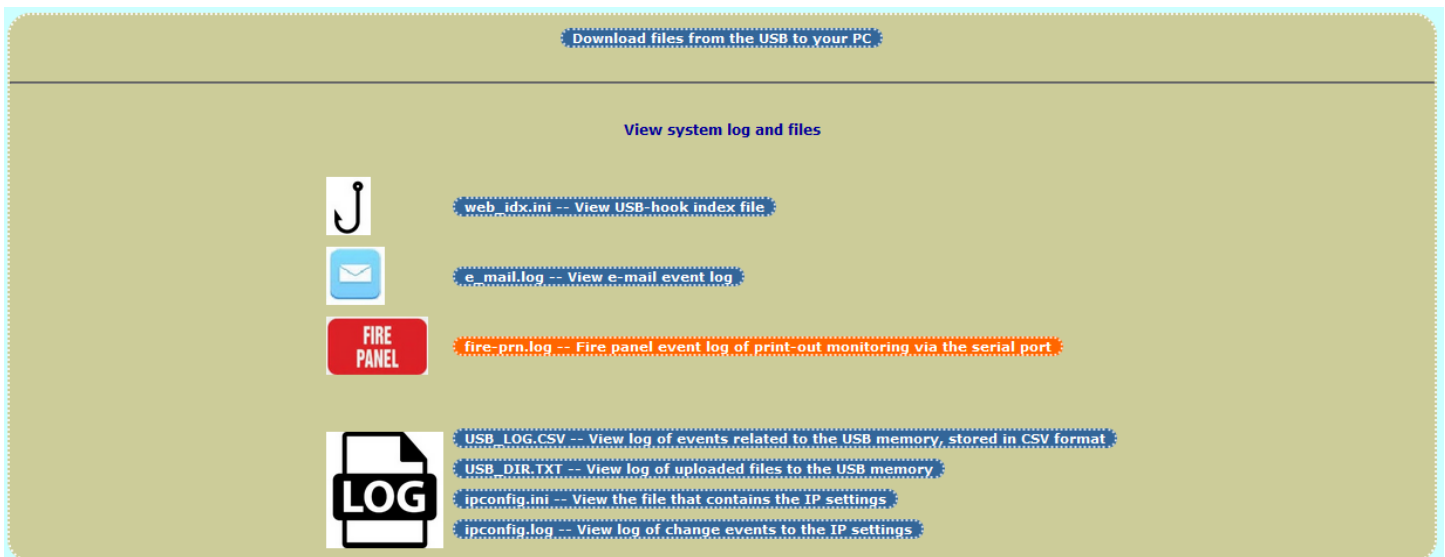
First log in to the web server using the IP address on any browser and enter credentials if required: The defaults are:

- User: **admin**
- Password: **Mircom**

Select the CONFIGURE link and then VIEW SYSTEM LOGS NAD FILES as depicted below.



Then select the link: "Fire **panel print event log**" and all the events will be downloaded to your PC for viewing.



Going back to the 10 events stored in the queue, and reviewing the table on last page, more information is provided on the next page on how to decode the information.

As previously mentioned, each fire printer event is stored into five registers. The way they are stored internally is depicted below:

```

////////////////////////////////////
// Registers
unsigned short R_node;      // Time Stamp HH:MM or Node 1..63 in FlexNet
unsigned char  R_slc;       // Loop 1..3 or CPU 0..7 in FlexNet
unsigned short R_addr;     // Address 1..159/201..359 or Zn: 1..400 in FlexNet
unsigned char  R_type;     // Type Sensor = 1, Module = 2, External = 3
unsigned char  R_event;    // Loop event
                        // Alarm activated = 1
                        // Alarm restored = 2
                        // Trouble = 3
                        // Trouble restored = 4
                        // Missing device = 5
                        // Open circuit trb = 6
                        // Other unknown = 255
// System event
// For FX3500 codes 101..199
// For FlexNet codes 201..254

```

And as all the protocols mentioned that it is converted to can only handle numbers, the information parsed and decoded from the printer string message is decoded as follows:

- **R\_NODE** The first register if it is an FX3500 panel contains a time stamp in military format from 0000 up to 2359 with the Hours + Minutes, if it is FlexNet panel and the event contains a node information, the node number will be stores instead, otherwise will contain a ZERO value.
- **R\_SLC** The second register contains the SLC loop number that can be 0 thru 3 for FX3500 panels and for FlexNet panels 0 thru 7.
- **R\_ADDR** The third register contains the device address of the event: 1 thru 159 for detectors, 201 thru 359 for modules and optionally for FlexNet 1 thru 400 for Zones.
- **R\_TYPE** The fourth parameter decoded the type of the event, it can be: 1 for Sensors /Detectors, 2 for modules and 3 for external or system events.
- **R\_EVENT** The last register contains a code depending on the event. Loop events are 1 thru 6, for events parsed from a FX3500 panel codes 101 thru 199 are used, for FlexNet panels codes 201 thru 254, and a 255 code is used for unknown or unrecognized events.

Loop codes for FX3500 and FlexNet	#
Alarm activated	1
Alarm restored	2
Trouble restored	4
Trouble	3
Missing device	5
Open circuit trb	6

FX3500 system codes	#
System reset in progress	101
System reset completed	102
Signal silence active	103
Signal silence restored	104
Auxiliary relay disconnect active	105
Auxiliary relay disconnect restored	106
Buzzer silenced	107
Printer dataloss trouble restored	108

FlexNet system codes	#
Access level changed	201
Addressable loop short restored	202
Addressable loop shorted trouble	203
Battery trouble active	204
Command menu activated	205
Command menu exited	206
Configuration mode activated	206
Fire drill activated	207
Fire drill cancelled	208
Ground fault active	210
Ground fault restored	211
Network reset activated	212
Network Restart command activated	213
Operator alert alarm buzzer silenced	214

FlexNet system codes (continued)	#
Operator alert trb buzzer silenced	215
Reports command selected	216
Signal silence active	217
Signal silence restored	218
Slave CPU Comms trouble	219
Slave CPU Comms trouble restored	220
Slave CPU config. trouble	221
System configurator connected	222
System reset activated	223
System reset complete	224
System reset in progress	225
System returned to normal mode	226
System startup	227
Total evacuation activated	228
Unconfigured CPU trouble	229

Once the registers information is known you can refer to the mapping table document:

- ❖ Mapping Table NX controllers for: **BACnet, Modbus, Optomux and N2-Open.**
  - <http://www.rikmed.com/OpenBAS/Programming/Mapping%20Table%20NX%20controllers.pdf>

The pertaining information for mapping into the most relevant protocols depending on if you selected to use the RES\_FLT or REMOTE registers extracted from page 21 of the mapping tables is shown below:

21

### ETH3.database RES\_FLT 1-255 (float) result registers in RAM 32 bit

**Un-scoped PLC operand: RES FLT 1001-1255**

<u>Protocol</u>	<u>Default mapping</u>		<u>Alternate mapping 1</u>
Optomux	RES_FLT	1-255	ADF 101-200 (1..100)
N2-Open	ADF	101-200	
Modbus RTU	Holding registers	5501-5755	
Modbus TCP	Holding registers	5501-5755	
BACnet MSTP	Analog value	5501-5755	
BACnet IP	Analog value	5501-5755	
SQL IP/Serial	Not available (planned)		

### ETH3.RMT.COM1 Remote points (float) in RAM via communication port 32 bit

**Un-scoped PLC operand: AO 1001-2000**

<u>Protocol</u>	<u>Default mapping</u>		<u>Alternate mapping 1</u>
Optomux	AO	1-100	(only the first 100 are available on this protocol)
N2-Open	AO	1-100	(only the first 100 are available on this protocol)
Modbus RTU	Holding registers	12001-13000	6501-7000 (only the first 500 available on alternate mapping)
Modbus TCP	Holding registers	12001-13000	6501-7000 (only the first 500 available on alternate mapping)
BACnet MSTP	Analog value	12001-13000	
BACnet IP	Analog value	12001-13000	Alternate Binary value dual mapping added on v3.10.1
SQL IP/Serial	Not available (planned)		

### ETH3.RMT.COM2 Remote points (float) in RAM via communication port 32 bit

**Un-scoped PLC operand: RMT 1001-2000**

<u>Protocol</u>	<u>Default mapping</u>		<u>Alternate mapping 1</u>
Optomux	AO	101-200	(only the first 100 are available on this protocol)
N2-Open	AO	101-200	(only the first 100 are available on this protocol)
Modbus RTU	Holding registers	13001-14000	4501-5000 (only the first 500 available on alternate mapping)
Modbus TCP	Holding registers	13001-14000	4501-5000 (only the first 500 available on alternate mapping)
BACnet MSTP	Analog value	13001-14000	
BACnet IP	Analog value	13001-14000	Alternate Binary value dual mapping added on v3.10.1
SQL IP/Serial	Not available (planned)		

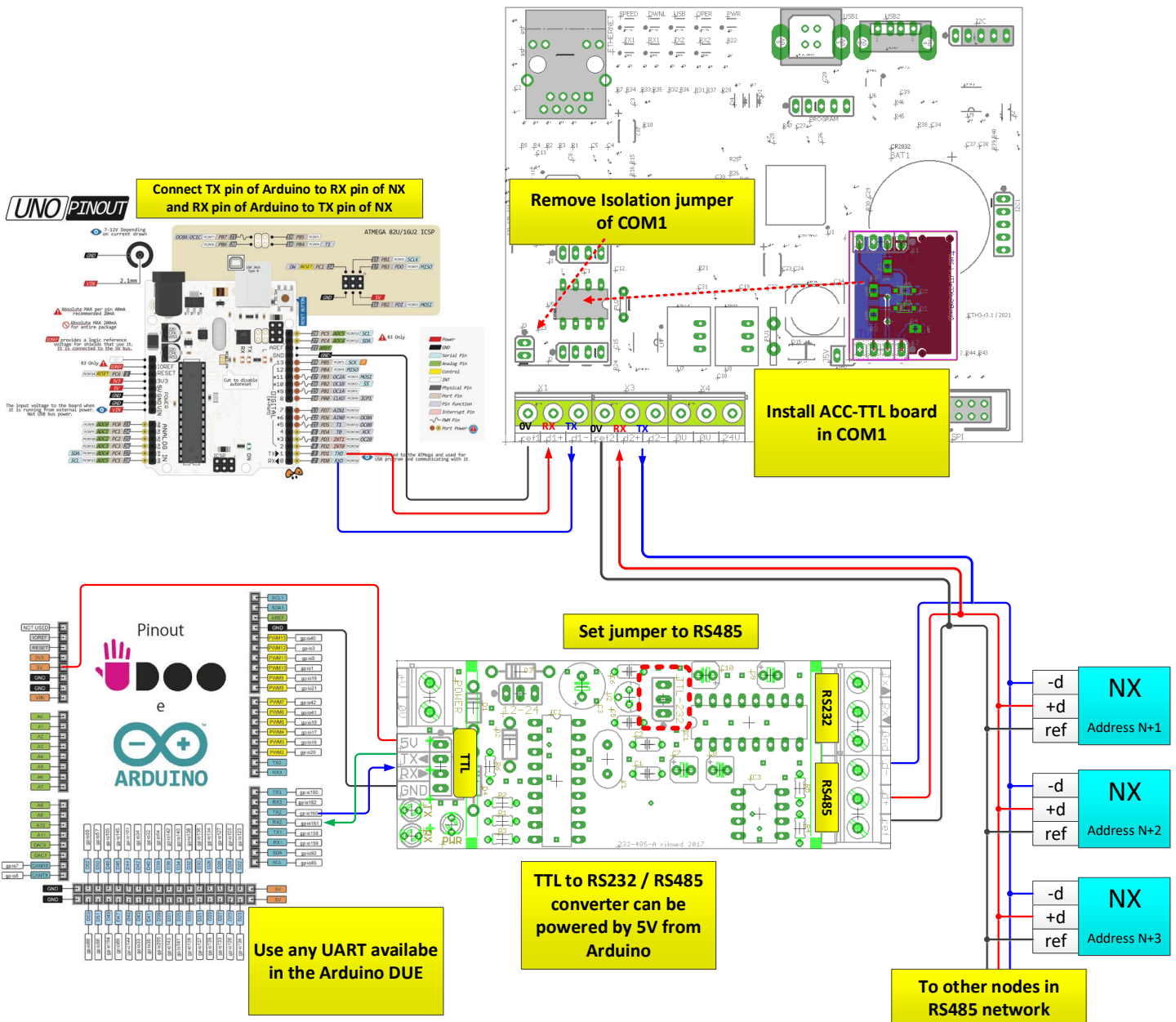
# Attach to Arduinos and Raspberry Pi

The ETH3 has different ways to connect an Arduino and a Raspberry. First the serial communication is done using either of COM1 or COM2.

While using COM1 an ACC-TTL adapter can be installed in place of the factory default RS485 driver to make the UART-to-UART connection between the ETH3 and the Arduino's or Raspberry PI's serial port.

When connecting to COM2 as it has a factory soldered SMD RS485 driver, an external converter must be used. This has the advantage that an RS485 network can be used instead of a 1-to-1 connection.

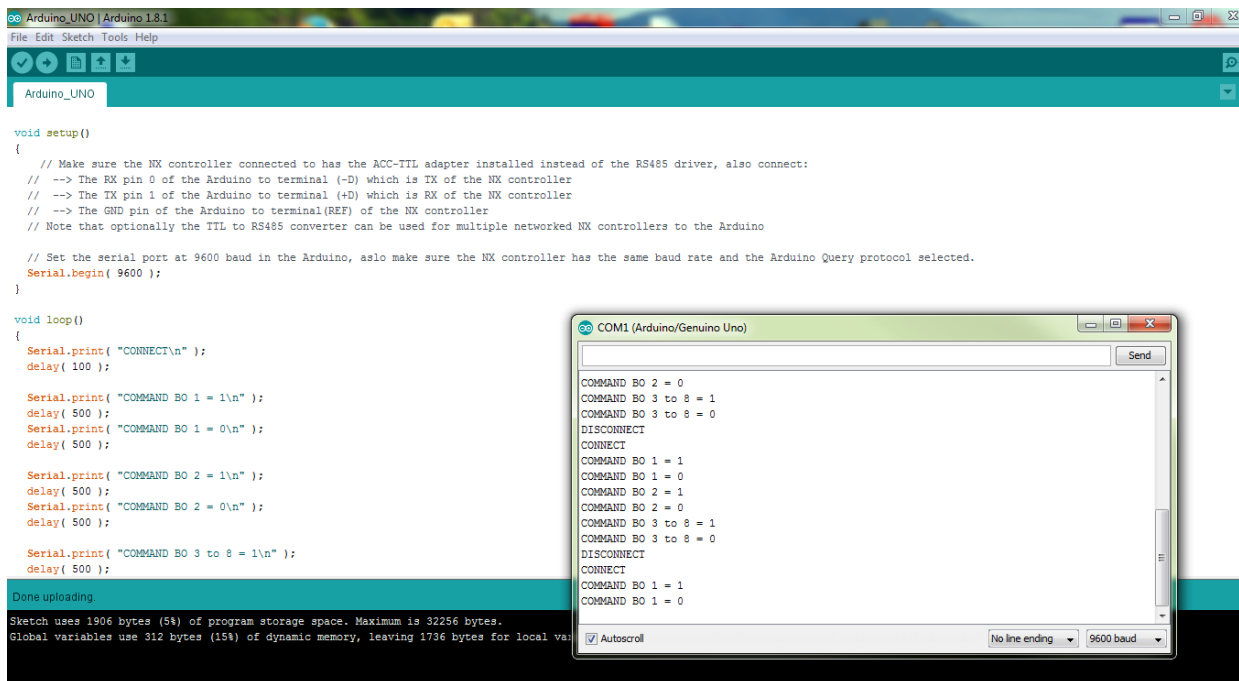
The following diagrams depict both connection methods:



## Creating Arduino sketches

A typical program in the Arduino shows how to turn ON/OFF relays.

The program first will CONNECT, then in sequence COMMAND (turn ON/OFF) relay 1, then do the same for relay 2 and finally simultaneously turn ON/OFF relays 3 to 8, at the end will DISCONNECT.



Turning on the serial monitor in the Arduino IDE shows the commands as they are sent. The whole sketch listing is shown below.

```
void setup()
{
  // Make sure the ETH3 is connected to has the ACC-TTL adapter installed instead of the // RS485 driver,
  // also connect:
  // --> The RX pin 0 of the Arduino to terminal (-D) which is TX of the NX controller
  // --> The TX pin 1 of the Arduino to terminal (+D) which is RX of the NX controller
  // --> The GND pin of the Arduino to terminal(REF) of the NX controller
  // Note that optionally the TTL to RS485 converter can be used for multiple networked NX
  // controllers to the Arduino

  // Set the serial port at 9600 baud in the Arduino, aslo make sure the NX controller has the
  // same baud rate and the Arduino Query protocol selected.
  Serial.begin( 9600 );
}

void loop()
{
  Serial.print( "CONNECT\n" );          // Disconnect to database
  delay( 100 );

  // Toggle first relay
  Serial.print( "COMMAND BO 1 = 1\n" );
  delay( 500 );
  Serial.print( "COMMAND BO 1 = 0\n" );
  delay( 500 );

  // Toggle second relay
  Serial.print( "COMMAND BO 2 = 1\n" );
  delay( 500 );
  Serial.print( "COMMAND BO 2 = 0\n" );
```

```

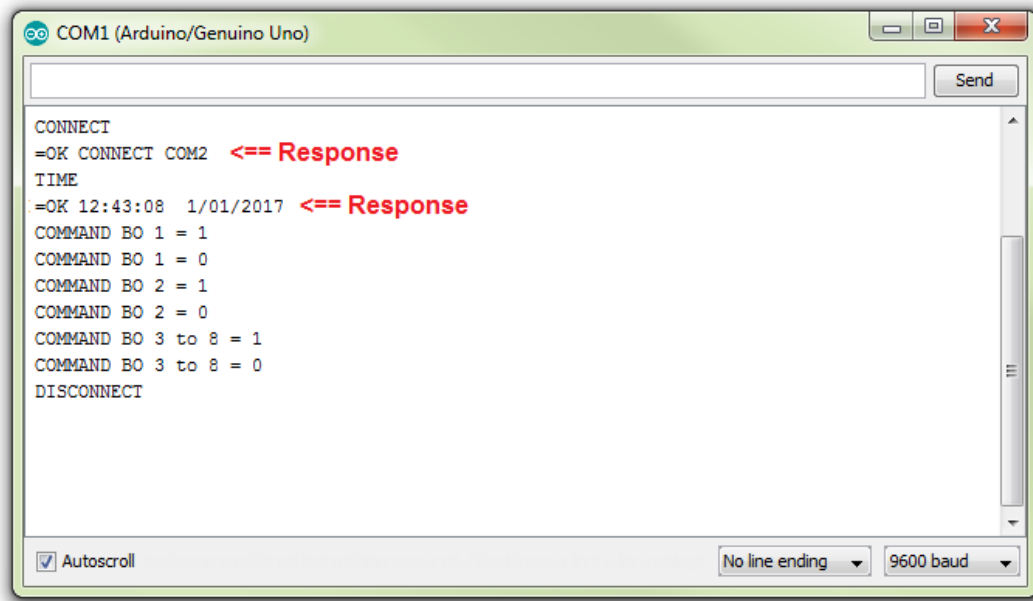
delay( 500 );

// Toggle simultaneously relays 3 to 8
Serial.print( "COMMAND BO 3 to 8 = 1\n" );
delay( 500 );
Serial.print( "COMMAND BO 3 to 8 = 0\n" );
delay( 500 );

Serial.print( "DISCONNECT\n" );    // Disconnect from database
delay( 100 );
}

```

This is the same program as the previous but now the responses are received and printed out in the serial monitor.



```

void setup()
{
  Serial.begin( 9600 );    // Same as last time
}

void loop()
{
  String response = { "" };

  // send a query and print the response
  Serial.flush();

  Serial.print( "CONNECT\n" );
  delay( 100 );
  response = Serial.readString();
  Serial.print( response );    // Print the CONNECT response in the serial monitor
  delay( 100 );
  response = Serial.readString();    // Get whatever is responded by the NX from the last
  // Serial.print() to flush the buffer

  Serial.print( "TIME\n" );
  delay( 100 );
  response = Serial.readString();
  Serial.print( response );    // Print the TIME response in the serial monitor
  delay( 100 );

  Serial.print( "COMMAND BO 1 = 1\n" ); // Toggle relay #1
  delay( 500 );
  Serial.print( "COMMAND BO 1 = 0\n" );
  delay( 500 );
  Serial.print( "COMMAND BO 2 = 1\n" );    // Toggle relay #2
}

```

```
delay( 500 );
Serial.print( "COMMAND BO 2 = 0\n" );
delay( 500 );

Serial.print( "COMMAND BO 3 to 8 = 1\n" ); // Toggle relays 3 to 8 at once
delay( 500 );
Serial.print( "COMMAND BO 3 to 8 = 0\n" );
delay( 500 );

Serial.print( "DISCONNECT\n" );
delay( 100 );
}
```

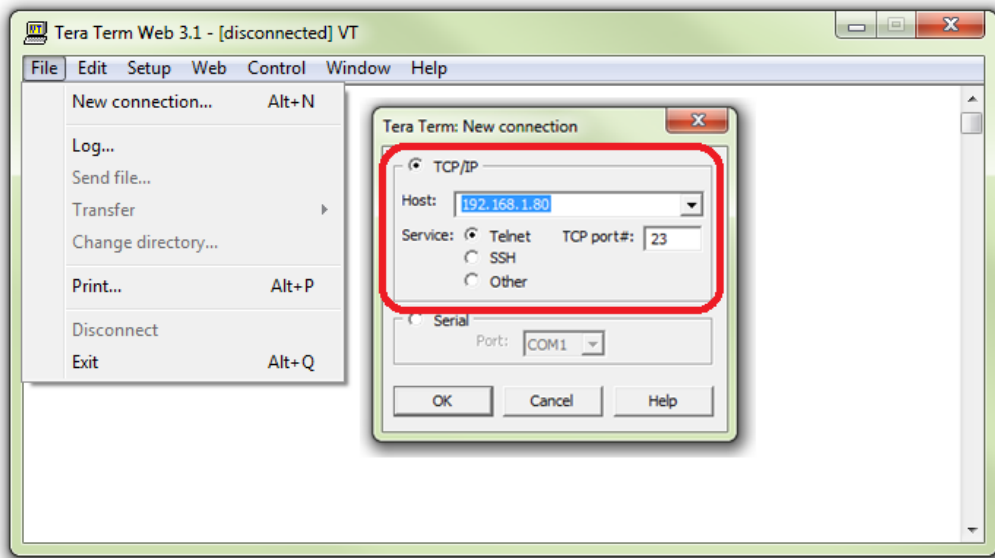


## Connect via Telnet and use SQL commands.

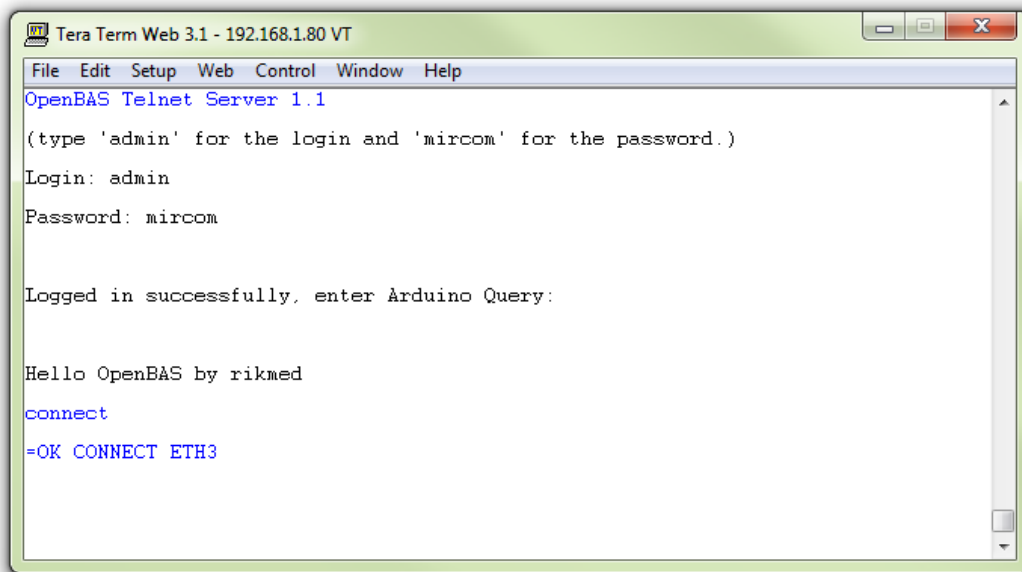
Connecting via **ETHERNET** is also similar to COM1 and COM2 connectivity, the only minor difference is the connection is done via the ETH3's IP port using TCP Telnet protocol on port 23. In a Raspberry PI or an Arduino with Ethernet networking a Telnet socket must be opened and used to send the SQL commands.

An example is given here using a terminal program called **TeraTerminal** running on Windows:

Select FILE →New connection... and select the TCP/IP connection method, input the Host IP address, select Telnet as the service and select port 23.

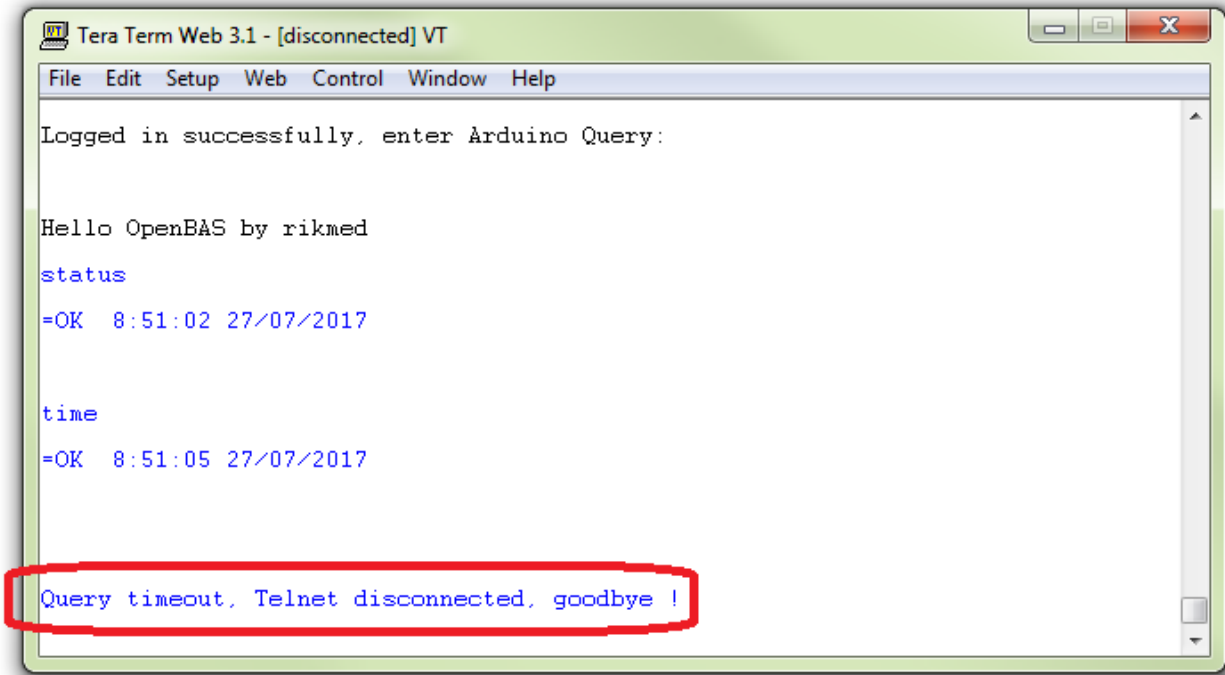


Connecting via **ETHERNET** is also similar to USB, COM1 and COM2 connectivity, the only minor difference is the connection is done via the ETH3 gateway using TCP Telnet protocol on port 23.



Once the connection is established the credentials are requested, type **“admin”** as the login and **“mircom”** as the password. Once connected the standard queries can be issued.

When logged in, Queries can be issued any time; however, after more than 60 seconds of inactivity the Telnet will force a disconnection and a login must be performed again:



```
Tera Term Web 3.1 - [disconnected] VT
File Edit Setup Web Control Window Help
Logged in successfully, enter Arduino Query:

Hello OpenBAS by rikmed
status
=OK 8:51:02 27/07/2017

time
=OK 8:51:05 27/07/2017

Query timeout, Telnet disconnected, goodbye !
```

---

## SQL Query Reference of Arduino Query protocol

The following section contains additional references and examples for creating the queries:

```
////////////////////////////////////  
// Arduino Query by rikmed
```

The "OpenBAS Arduino Query Language" or OAQL for short, referred as such hereafter, is simple using plain text for communications, and a response is returned for a Query.

To set up communication between an Arduino/Raspberry PI/Terminal (hereafter only referred as "Arduino") using this Query language will be depending on the communication port.

Using COM1 or COM2 interface to an Arduino is possible by installing an ACC-TTL interface board instead of the standard RS485 interface.

This way a one to one connection can be created between one Arduino and one NX board, distance between the two of them is however limited to only three feet (1 meter).

Due to the one-to-one connection method no addressing information is needed in the Query at all.

Optionally a TTL-RS485 converter can be used to overcome this limitation, and if using addressing information in the Query several NX can be networked to a single Arduino communication port.

The USB port supports the OAQL as a point to point only as it is by nature a one-to-one connection, using the CONNECT / DISCONNECT commands an attachment is made to the NX database

Prior to sending queries, the USB should be placed into "Arduino Query" mode by sending the '~' character.

Similarly sending a space ' ' places the USB port in ASCII mode and sending a '>' character places the USB port in Optomux/N2-open mode

The Ethernet gateway will have its own implementation of the OAQL as its own database might differ from that of a standard NX/Dual core controller with its own expanded ETH3 database.

The Query structure is as follows, it is case insensitive and always should start by attaching the Arduino to the NX controller using the CONNECT command:

### **DISCONNECT**

Disconnect (or de-attaches) the Arduino any attached NX database (as this is broadcast all NX controllers on the network de-attach at once and revert to 1-1 mode)

### **CONNECT**

Connects (or attaches) the Arduino to the NX database in a 1-1 fashion

### **DISCONNECT\_NWK**

Disconnect (or de-attaches) the Arduino from the network, (as this is broadcast all NX controllers on the network de-attach at once but stay in networked mode)

### **CONNECT\_NWK [Address: is a number between 1..254]**

Connects (or attaches) the Arduino to the NX database with the corresponding address, NX devices should be set with unique addresses in the RS485 network

The first query should be always a DISCONNECT as if a previously CONNECT command is in effect, it will force the connection to start with a known state in 1-1.

Note that when issued on a NX networks, as all controllers will reply with =OK\_DISCONNECT, garbage might be received.

Once connected, the NX controllers will monitor communications, if no activity is detected for more than 10 minutes they will automatically revert to the disconnected state.

If a connection is successful, the connected NX controller will reply with an acknowledge to the connect message with the following format:

**=OK CONNECT PORT [ADDRESS 1..254]**

Optionally the address will be reported back if a network connection was requested  
Once connected, the database can be queried using the following commands:

```
-- QUERY--                                -- NOTES --  
NAME  
-- RESPONSE--                              -- NOTES --  
=NAME [Controller name]                 Controller (Database source) name
```

```
-- QUERY--                                -- NOTES --  
VERSION  
-- RESPONSE--                              -- NOTES --  
=v.vv.v NX10                             Returns version number and controller type
```

If any query is not being able to be interpreted it will give an error message in the form of:

**=ERROR [Optional error code]**

```
-- QUERY--                                -- NOTES --  
TIME  
-- RESPONSE--                              -- NOTES --  
=OK hh:mm:ss wdy dd/mm/yyyy             Will report back current time
```

```
-- QUERY--                                -- NOTES --  
STATUS  
-- RESPONSE--                              -- NOTES --  
=OK hh:mm:ss wdy dd/mm/yyyy             Will return OK of no alarms present and local time  
-- RESPONSE--                              -- NOTES --  
=ALM [0x00]                             Will return alarms present and show in flag bits  
                                           1 to 8 in hexadecimal format
```

```
-- QUERY--                                -- NOTES --  
STAT_0 .....                          Will send current value (value) of selected DB object with 0 decimal  
                                           positions (integer)  
STAT_1 .....                          Will send current value (value) of selected DB  
                                           object with 1 decimal positions .  
STAT_2 .....                          Will send current value (value) of selected DB  
                                           object with 2 decimal positions  
STATUS .....                          Will send current value (value) of selected DB  
                                           object with 3 decimal positions (DEFAULT)
```

-- QUERY--

**STATUS AI [1..40] [-D0..3]**

**STATUS AI [1..40] TO [1..40]**

**STATUS AO [1..10] [-D0..3]**

**STATUS AO [1..10] TO [1..10]**

**STATUS BI**

**STATUS BO**

**STATUS BI 1**

**STATUS BI 1 TO 40**

**STATUS BO 1 to 60**

**STATUS ADB 1 TO 100**

**STATUS ADI 1 TO 100**

**STATUS ADF 1 TO 100**

**STATUS RBIT 1 to 255**

**STATUS RFLT 1 to 255**

**STATUS RMT 1 to 255**

**STATUS TMR 1 to 16**

-- RESPONSE--

**=VAL: x,y,z...**

**=VAL: x,y,z...,+**

-- QUERY--

**STATUS AI > 5**

-- RESPONSE--

**=VAL: [1]6.000,[2]1500.120,**

-- NOTES --

Will send current value (value) of selected AI

Will send current value (value) of selected AI's up to 40 can be queried at once

Will send current value (value) of selected AO's

Will send current value (value) of selected AO's up to 10 can be queried at once

Will send current status of BI-1..8 in a comma separated value CSV format

Will send current status of BO-1..8 in a comma separated value CSV format

Will send current status of one binary input

Will send current status of one binary inputs, up to 40 binary inputs can be queried at once

Will send current status of one binary output current status of one binary output, up to 60 binary outputs can be queried at once, values in this query will be 1 or 0

Will send current status (value) of the BYTE setpoint selected

Will send current status (value) of the INT setpoint selected

Will send current status (value) of the FLOAT setpoint selected

Will send current status (value) of the RES\_BIT selected

Will send current status (value) of the RES\_FLOAT selected

Will send current status (value) of the REMOTE POINT selected

Will send current status (value) of the REMOTE POINT selected

-- NOTES --

Returns current value or values of queried data

Returns current value or values of queried data, buffer is full

-- NOTES -

Will select any AI whose value is greter than 5

-- NOTES --

Any AI that meets the criteria will be returned The index of the AI will be returned between square Brackets followed by the value, each data base Object will be separated by comma.

Note the =VAL: identifier at the beginning of the Response to identify the STATUS between the SELECT queries

```

-- QUERY--
SELECT AI > 5
-- RESPONSE--
=IDX: 1,2,

-- NOTES --
Will select any AI (1 to 8 default) whose value > 5

-- NOTES --
Any AI that meets the criteria will be returned
The index of the AI which meet the criteria will be
Returned, each object will be separated by comma.
Note the =IDX: identifier at the beginning of the
Response to identify the SEELECT between the STATUS queries.

-- QUERY--
SELECT AI 1 TO 40 > 5
SELECT AI 1 TO 40 > 5 AND <= 10
-- RESPONSE--
=IDX: 1,2,3,4
=IDX: 1,2,3,4,5,6,7,8,9,10,11,+
-- NOTES --
Will select any AI between 1 and 40 whose value > 5
Will select any AI between 1 and 40 whose value >
5 AND <= 10
-- NOTES --
Query returns AI's that meet the criteria, the
database object queried can be any and not only
limited to analog inputs
Query returns AI's that meet the criteria, the
database object queried can be any and not only
limited to analog inputs, buffer is full is denoted
with the plus (+) character at the end.

-- RESPONSE--
SELECT BI == 1
SELECT BO 1 TO 20 = 1
-- RESPONSE--
IDX: 1,2,3,4,
=IDX: 1,*2,#3,4,5,6,7,8,9,10,11,+
-- NOTES --
Will select all binary outputs in the range 1 to 8
that are = 1 (ON)
Will select all binary outputs in the range 1 to 20
that are = 1 (ON)
-- NOTES --
Query returns BO's that meet the criteria, the
database object queried can be any and not only limited to binary
outputs
Query returns BO's that meet the criteria,
the database object queried can be any and not only limited to binary
outputs, buffer is full is denoted
with the plus (+) character at the end.
For analog outputs and binary outputs an asterisk (*) means the
output is overridden by communications, and the pound character (#)
denotes it has a high priority local override.

```

The following comparison operators can be used, also they can be joined with the **AND** or **OR** logic operators to form compounded comparisons:

- **>** Greater than
- **>=** Greater or equal than
- **<** Less than
- **<=** Less or equal than
- **=** or **==** Treated as equal than or assignment indistinctively
- **!=** Not equal than

## Priorities for writing to data base objects in NX controllers

When commanding in a query a data base object, its priority level should be taken into account, higher priority commands will have precedence over low priority commands, the table below depicts the priority levels used for NX controllers.

### HIGHEST PRIORITY

PRIORITY_UNDEFINED_OFF	0	// 0000-0000	PRIORITY = 0	(if no LOCAL/COMM override)
PRIORITY_UNDEFINED_ON	1	// 0000-0001	PRIORITY = 1	(if no LOCAL/COMM override)
PRIORITY_UNDEFINED_OFF	2	// 0000-0010	PRIORITY = 2	(if no LOCAL/COMM override)
PRIORITY_UNDEFINED_ON	3	// 0000-0011	PRIORITY = 3	(if no LOCAL/COMM override)
PRIORITY_OVR_LOCAL_OFF	4	// 0000-0100, Set		LOCAL OVRERIDE BIT
PRIORITY_OVR_LOCAL_ON	5	// 0000-0101, Set		LOCAL OVRERIDE BIT
PRIORITY_OVR_COMM_OFF	6	// 0000-0110, Set		COMM OVRERIDE BIT
PRIORITY_OVR_COMM_ON	7	// 0000-0111, Set		COMM OVRERIDE BIT
PRIORITY_LOGIC_OFF	8	// 0000-1000	PRIORITY = 0	(if no LOCAL/COMM override)
PRIORITY_LOGIC_ON	9	// 0000-1001	PRIORITY = 1	(if no LOCAL/COMM override)
PRIORITY_SCHEDULE_OFF	10	// 0000-1010	PRIORITY = 2	(if no LOCAL/COMM override)
PRIORITY_SCHEDULE_ON	11	// 0000-1011	PRIORITY = 3	(if no LOCAL/COMM override)
PRIORITY_COMM_OFF	12	// 0000-1100	PRIORITY = 4	(if no LOCAL/COMM override)
PRIORITY_COMM_ON	13	// 0000-1101	PRIORITY = 5	(if no LOCAL/COMM override)
PRIORITY_OPERATE_OFF	14	// 0000-1110	PRIORITY = 6	(if no LOCAL/COMM override)
PRIORITY_OPERATE_ON	15	// 0000-1111	PRIORITY = 7	(if no LOCAL/COMM override)

### LOWEST PRIORITY

The **COMMAND** query sends a priority 12/13, so any logic instruction or schedule will have higher precedence on them, the **COMM\_OV** will send an override with priority 6/7 so will have higher priority.

Note that only **analog outputs** and **binary outputs** have priority bits for overriding them through communications, all other data objects will only have priority levels 8 and above.

**COMM\_AU** returns previously overridden analog and binary outputs to their released or auto (previous) state.

**COMMAND** can write **from 1 and up to 20 database objects in a single query**, the numbers below depict only the allowable ranges that can be addressed, but any **COMMAND** query with more than 20 data base objects addressed will fail.

-- QUERY--

**COMMAND AO 1 TO 10 = VALUE**

**COMMAND BO 1 to 60 = 1 or 0**

**COMM\_OV BO 1 to 20 = 1 or 0**

**COMM\_AU BO 1 to 20**

-- NOTES -

Commands analog outputs to the given value, range of Commanded value for analog outputs is **0-100 (%)**  
Commanded value for ADF is +/-32767

Commands binary outputs to the given value, value Of command is **0 or 1**

Overrides binary outputs to the given value, Value of override is **0 or 1**

releases communication overrides for binary and analog outputs.

-- QUERY--

**COMMAND ADB [1..100] = VALUE**

**COMMAND ADI [1..100] = VALUE**

**COMMAND ADF [1..100] = VALUE**

**COMMAND RBIT [1..255] = 1 or 0**

-- NOTES -

Commands BYTE setpoint to given value, range of commanded value is **0..255**

Commands INT setpoint to given value, range of commanded value is **0..65535**

Commands FLOAT setpoint to given value, range of commanded value is **+/-32767**

Commands RES\_BIT register to given value, range of

**COMMAND RFLT [1..255] = VALUE** Commanded value is 1 or 0  
Commands RES\_FLT register to given value,  
range of commanded value is **+/-32767**

**COMMAND RMT [1..255] = VALUE** Commands RES\_FLT register to given value,  
range of commanded value is **+/-32767**

-- RESPONSE--

**=OK =VALUE [1,2,3,4]**

**=OK =VALUE [1,2,3,4,+]**

-- NOTES --

Command accepted plus index of commanded objects

Command accepted plus data, more data is available  
but not able to fit in the 64 byte buffer size.

Note however that all objects were commanded

If any query is not being able to be interpreted it will give an error message in the form of:

**=ERROR [Optional error code]**