Mircom

OpenBAS | BUILDING AUTOMATION SYSTEM

System Design Studio User Guide

LT-6675 Rev. 10
Aug 2025

Revision history

Date	Revision	Description
February 2020	Rev. 0	Initial Release with System Design Studio 1.0.1
April 2020	Rev. 1	Fixed bookmarks and minor updates for SDS 1.0.2
August 2020	Rev. 2	Project tree support and OpenBAS-NWK-ETH3 configuration support for SDS 1.1.0
March 2021	Rev. 3	Emulator, built in user guide, thermistor, BACnet tunneling and multi-language support for SDS 1.2.0
June 2021	Rev. 4	Analog Output configuration bug fixes, Configure from script support for emulator; SDS 1.2.1
December 2022	Rev. 5	Live values in ladder editor, offline ladder editor, standalone OpenBAS-NWK-ETH3 support, fieldbus statistics window; SDS 1.3.0
June 2023	Rev. 6	Input Overrides, call across PLC's, Lagger calculator option in Leader Calc. instruction, check for updates; SDS 1.3.1
Nov 2023	Rev. 7	Fix inability to connect to slaves through BACnet/MSTP fieldbus; last release supporting 32 bit systems; SDS 1.3.2
Jan 2024	Rev. 8	Support for "Persistent point name groups", fix incorrect output from Proportional Control instruction when using RF as output; SDS 1.3.3
Oct 2024	Rev. 9	Batch programming including support for script programming on OpenBAS-NWK-ETH3, BACnet/MSTP remote point support on OpenBAS-NWK-ETH3, dedicated integration settings for Binary inputs of OpenBAS-HV-NXSF, support for various Modbus point value encodings; SDS 2.0.0
Aug 2025	Rev. 10	OpenBAS-HV-NXVAV support, MiPages for AI, BACnet/Modbus IP remote points, Cloudflare tunnel support, ETH3 fieldbus and BACnet/MSTP statistics, ETH3 SPI cache settings, ETH3 Modbus packet recorder and other features, Remove & archive, Force save scripts; SDS 2.1.0

About this guide

This guide covers the System Design Studio version 2.1.0 with **recommended firmware versions of 3.18.0 and greater. Note**: a firmware version of 3.15.5 or greater is required on "NX" controllers and a firmware version of 3.17.0 or greater is required on OpenBAS-NWK-ETH3 controllers. See LT-6630 – Driver Installation and Firmware Upgrade Procedures.pdf for help with upgrading a controller's firmware.

Overview

The Mircom System Design Studio is a powerful configuration tool for OpenBAS systems. The System Design Studio can configure every aspect of an OpenBAS system including communication settings, PLC logic, schedules, trends and more. In addition to system configuration, users can easily test their logic with personalized views and filters. Although the System Design Studio can be used for basic system monitoring, Mircom also offers the System Management Studio (Powered by Niagara Framework) which is intended for system monitoring, analytics, advanced trending, alarming, master scheduling and dashboard design.

SYSTEM DESIGN STUDIO	SUPPORTED CONTROLLERS	FIRMWARE VERSION MINIMUM	FIRMWARE VERSION SUPPORTED
2.1.0	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN	3.15.5+	3.18.0+
	OpenBAS-NWK-ETH3	3.17.0+	3.18.0+
2.0.0	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN	3.15.5+	3.17.0+ 3.17.0 is BTL Certified
	OpenBAS-NWK-ETH3	3.17.0+	3.17.0+
1.3.3	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN, OpenBAS-NWK-ETH3	3.15.5+	3.16.3+
1.3.2	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN, OpenBAS-NWK-ETH3	3.15.5+	3.16.1+
1.3.1	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN, OpenBAS-NWK-ETH3	3.15.5+	3.15.5+
1.3.0	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN, OpenBAS-NWK-ETH3	3.15.1+	3.15.1+
1.2.1	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN	3.04.0+	3.09.4+
	OpenBAS-NWK-ETH3	3.06.0+	3.09.4+
1.2.0	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN	3.04.0+	3.09.0+
	OpenBAS-NWK-ETH3	3.06.0+	3.09.0+
1.1.0	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN	3.04.0+	3.04.0+
	OpenBAS-NWK-ETH3	3.06.0+	3.06.0+
1.0.2	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN	3.04.0+	3.04.0+
1.0.1	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN	3.04.0+	3.04.0+
1.0.0	All "NX" controllers (OpenBAS-xx-NXxxxx), OpenBAS-HV-LEARN	3.0.0+	3.0.0+

Specifications

- Supported OS
  - Windows 10, Windows 11
- Supported architecture
  - 64-bit
- Supported controllers
  - All NX-based controllers (OpenBAS-XX-**NX**XXX, OpenBAS-HV-LEARN), OpenBAS-NWK-ETH3

Mircom

Table of Contents

User interface overview 8

Selecting a language 10

Connecting to and running the NX emulator 10

Connecting to controllers 10

Setting up your first OpenBAS network 15

Configuring network settings and IP address 17

Upgrading firmware 17

Discovering OpenBAS gateways 18

Creating projects 19

Removing projects and controllers 21

Naming projects and controllers 21

Importing projects 22

Get scripts and Force save scripts 22

Remotely accessing OpenBAS controllers using Cloudflare tunnels 23

Recovering EZ-Script files from USB Flash memory 23

Importing and exporting IP connection lists 24

Software resetting controllers 24

Power up initializations 24

Performing a factory reset 25

Memory overview 25

Dynamic memory 28

Viewing and synchronizing a controller's real time clock 28

Date and time settings 29

Creating schedules 30

Configuring and calibrating inputs 30

Airflow balancing on the OpenBAS-HV-NXVAV Series 33

Configuring and calibrating outputs 34

Output priorities 37

Naming points 37

Limiting point values 38

Configuring fieldbus and communication settings 39
Configuring OpenBAS-NWK-ETH3 communication settings through USB flash memory 49
Configuring email alerts 50
Debugging and testing logic 51
Adding expandable memory accessories 51
LCD display and alarm configuration 52
Dual core display settings 54
Viewing, Setting and understanding the internal point database 55
Viewing trends 60
Configuring trends 60
Configuring USB trends 61
Configuring remote points 61
Configuring IP remote points 65
Connecting wireless devices 66
Backup to disk 67
Restore from disk 68
MiPages for AI (Beta) 68
SPI cache 70
SMS alerts 70
Configure from script (EZ-Script) 71
Script status 72
Batch programming 73
File transfer 74
Logic and the PLC 75
Logic instruction overview 77
Ladder (boolean) type instructions 77
And instruction 77
Nand instruction (inverted And) 78
Or instruction 78
Remote online (Or instruction) 78
Nor instruction (inverted Or) 78

Xor instruction (Exclusive Or) 79
Nxor instruction (Also known as Xnor, inverted Exclusive Or) 79
Invert instruction (Also known as Not) 79
And/Or instruction (A·B+C·D) 79
Or/And/Inv instruction (And/Or inverted) 80
Math type instructions 80
Min instruction (minimum) 80
Max instruction (maximum) 81
Avg instruction (average) 81
“+” instruction (addition) 81
“-“ instruction (subtraction) 81
“*” instruction (multiplication) 82
“/” instruction (division) 82
Cmp (comparison) type instructions 82
“>” instruction (greater than) 82
“<” instruction (less than) 82
“=” instruction (equal) 83
“>=” instruction (greater than or equal to) 83
“<=” instruction (less than or equal to) 83
“!=” instruction (not equal to) 83
Other instructions 84
Label instruction 84
Long label instruction 84
Timer instruction 84
Jump instruction 87
Call instruction (subroutine call) 87
End instruction 88
LightG instruction (lighting group) 88
Assign instruction (controlling outputs) 89
Multiple Assign instruction 89
Totalizer instruction (pulse accumulator and energy metering) 90

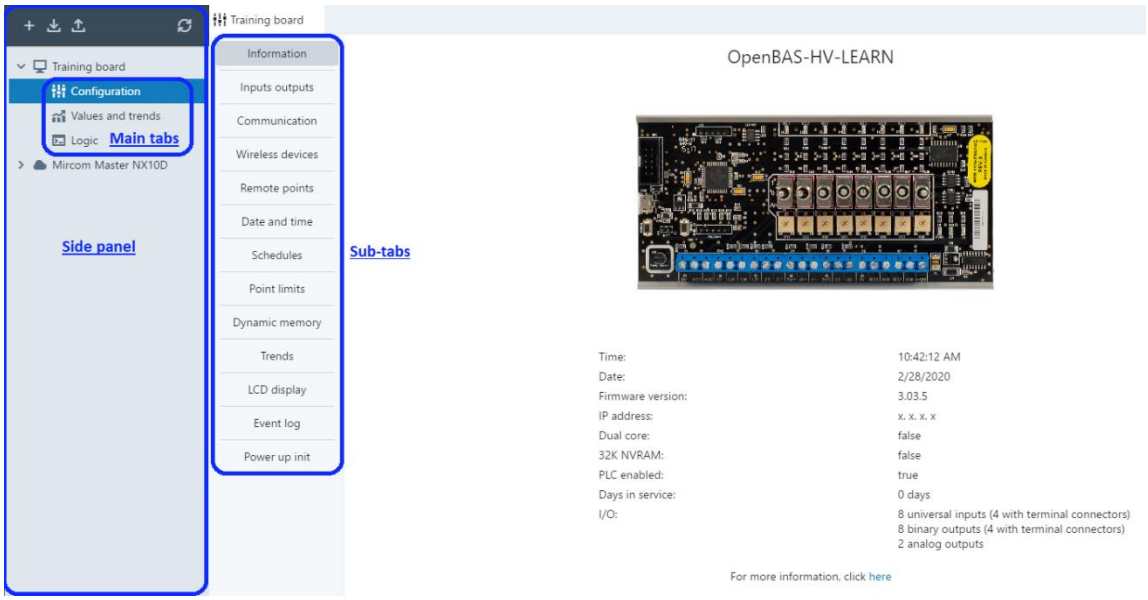
PropCtrl instruction (proportional control) 92
AHU/RTU instruction (air handling unit or roof top unit) 93
HourCnt instruction (hour counter) 95
Alt/Pll instruction (alternate and parallel) 95
In Range instruction (and hysteresis/deadband) 98
Start/Stop instruction 99
Limit instruction 100
PID instruction (proportional, integral, derivative controller) 100
PropStager instruction (proportional stager) 103
LeaderCal instruction (leader and lagger calculator) 106
Staggered Start/Stop instruction 107
Special instructions 108
Random number special instruction 109
Modulus special instruction (division remainder) 109
Square root special instruction 110
Superheat special instruction (for R22 refrigerant) 110

User interface overview

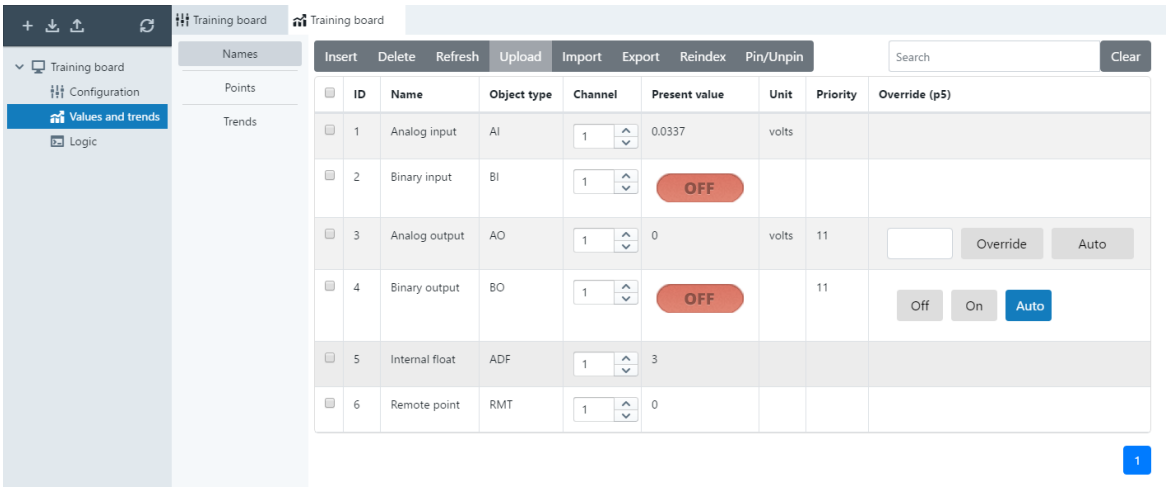
The user interface can be divided into four main sections including the side panel, main tabs, sub-tabs and user guide. The side panel is where connections to controllers are made and maintained. The main tabs include *Configuration* for all controller configuration, *Values and trends* for viewing or modifying all points and trends, and *Logic* for setting up all control logic. The user guide is hidden off to the right.

- Both the side panel and user guide have 3 visibility modes which are collapsed-locked, expanded-locked and hover mode. Hover mode will overlay the side panel or user guide on top of the screen when “hovering” the mouse cursor over the left, or right side of the screen respectively.
- The user guide’s mode can be cycled through at any time using the shortcuts Ctrl + H (“H” for “help”), or F1 function key.

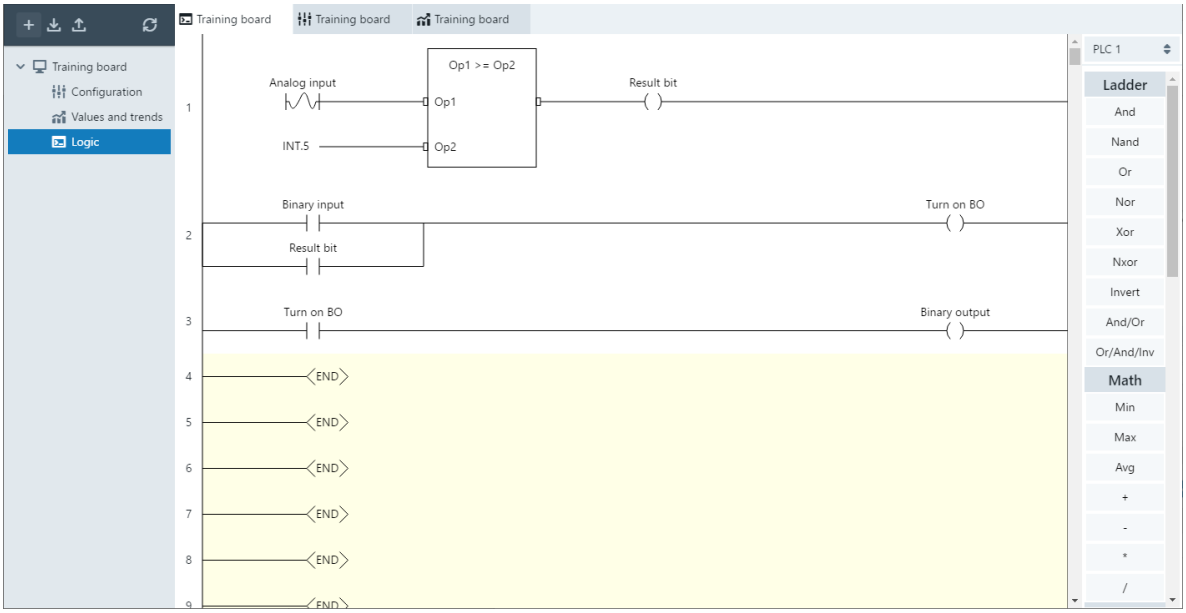
User interface and Configuration main tab

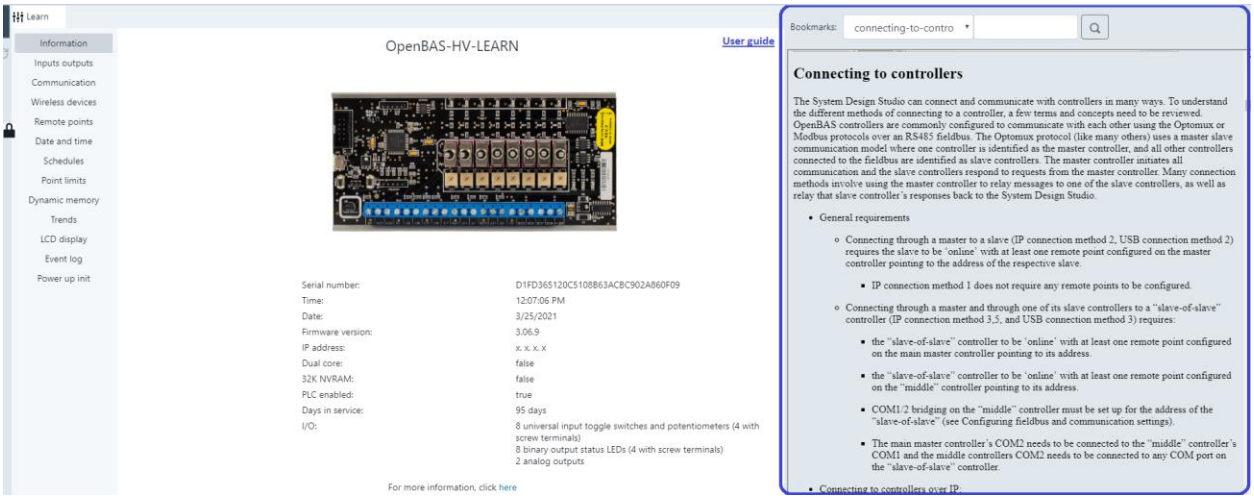


Values and trends main tab



Logic main tab





Selecting a language

The System Design Studio supports English, Spanish, German, French, Italian, Dutch Nederlands, Hindi and Punjabi. The language can be changed by clicking the Help menu at the top left and clicking Select language. Alternatively, the shortcut Ctrl + L can be used to quickly access the language menu.

Connecting to and running the NX emulator

The NX emulator is a small Java-based program that acts like an OpenBAS-HV-NX10D controller. The emulator is great for learning and training purposes since it allows users to configure logic and schedules without any hardware. To run and connect to the emulator, after clicking the "+" icon in the top left-hand corner, select Connect from the *Action:* dropdown, then select NX-Emulator from the *Device type:* dropdown and click Add. The OpenBAS emulator will open as a separate program and a controller with the name "NX-emulator" will be added to the System Design Studio's side panel.

- Not all settings and features that are available when connected to an OpenBAS-HV-NX10D are available when using the NX emulator.
- Most interaction will be through the System Design Studio however every point can be Set from the separate NX emulator program.
- All configurations will be lost when the NX emulator program is closed however the current state of the NX-emulator can be saved to and restored from disk using the *Disk* button found at the top of the NX-emulator program.

Connecting to controllers

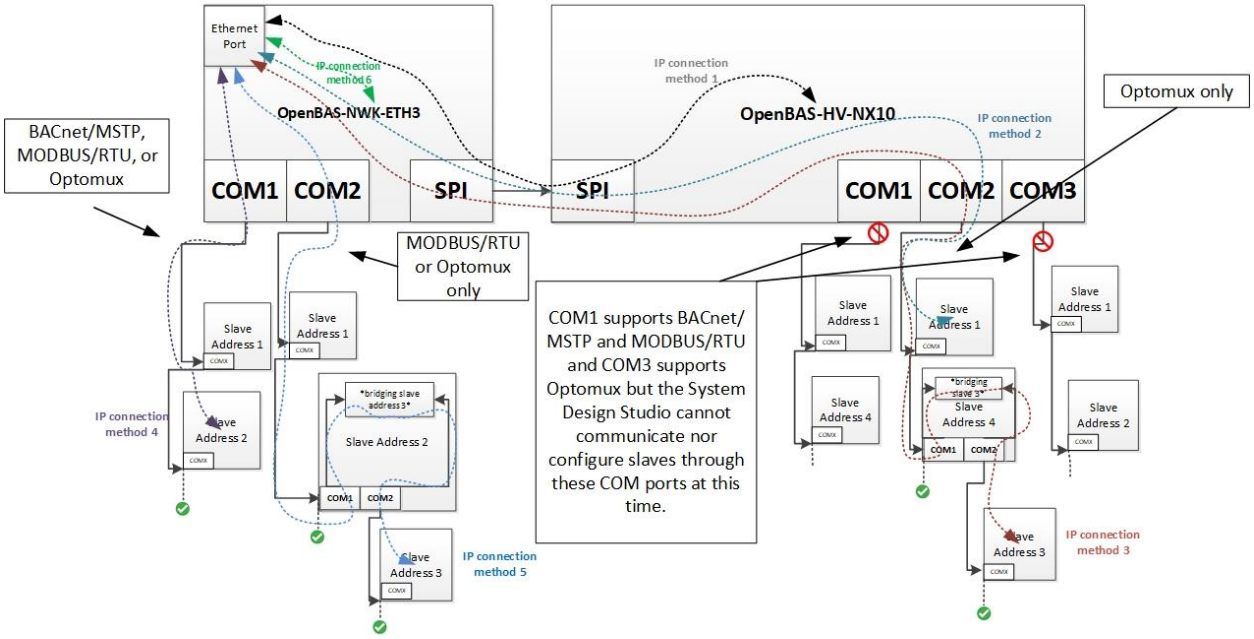
The System Design Studio can connect and communicate with controllers in many ways. To understand the different methods of connecting to a controller, a few terms and concepts need to be reviewed. OpenBAS controllers are commonly configured to communicate with each other using the Optomux, Modbus or BACnet protocols over an RS485 fieldbus. The Optomux protocol (like many others) uses a

master slave communication model where one controller is identified as the master controller, and all other controllers connected to the fieldbus are identified as slave controllers. The master controller initiates all communication, and the slave controllers respond to requests from the master controller. Many connection methods involve using the master controller to relay messages to one of its slave controllers, as well as relay the slave controller's responses back to the System Design Studio. This section details the steps and requirements involved in connecting to controllers in various ways. See *Setting up your first OpenBAS network* for a step by step walk through.

- **General requirements**
  - The OpenBAS-NWK-ETH3 enables the System Design Studio to communicate with OpenBAS controllers through IP to controllers connected to its COM1, COM2 and SPI ports. While the System Design Studio only supports Optomux, the OpenBAS-NWK-ETH3 facilitates connections to its COM1 and COM2 slaves over Modbus and/or BACnet (COM1 only) by tunneling Optomux over Modbus/RTU and Optomux over BACnet/MSTP.
  - Connecting through a master to a slave (IP connection method 2, USB connection method 2) requires the slave to be 'online' with at least one remote point configured on the master controller pointing to the address of the respective slave.
    - IP connection method 1 does not require any remote points to be configured.
  - Connecting through a master and through one of its slave controllers to a "slave-of-slave" controller (IP connection method 3,5, and USB connection method 3) requires:
    - the "slave-of-slave" controller to be 'online' with at least one remote point configured on the main master controller pointing to its address.
    - the "slave-of-slave" controller to be 'online' with at least one remote point configured on the "middle" controller pointing to its address.
    - COM1/2 bridging on the "middle" controller must be set up for the address of the "slave-of-slave" (*see Configuring fieldbus and communication settings*).
    - The main master controller's COM2 port needs to be connected to the "middle" controller's COM1 port and the middle controllers COM2 port needs to be connected to any COM port on the "slave-of-slave" controller.
    - Both involved fieldbuses must be configured with Optomux protocol at a minimum baud rate of 38400 bps.
- **Connecting to controllers over IP, outside of Projects** (not recommended since scripts will not be saved to main ETH3, see *Creating projects*)**:**
  - **Requirements**:
    1. PC must be on same network as controller.
       1. Port forwarding and dynamic DNS domains are supported.
    2. The IP address of the respective OpenBAS-NWK-ETH3 is needed. A Discovery feature is available (dependent on network configurations).
       1. See *Configuring network settings and IP address* for help determining the OpenBAS-NWK-ETH3's IP address.

3. Connecting through an OpenBAS-NWK-ETH3's SPI port to an NX10 series controller, OpenBAS-HV-LEARN, or slaves to any of these controllers requires that the SPI port is enabled on the OpenBAS-NWK-ETH3 (disabled by default). To enable, follow the steps below to connect directly to an OpenBAS-NWK-ETH3, right click the connection in the side panel and then click "Enable SPI". If "Disable SPI" is displayed, then the SPI port is already enabled.

System Design Studio IP Connection Options



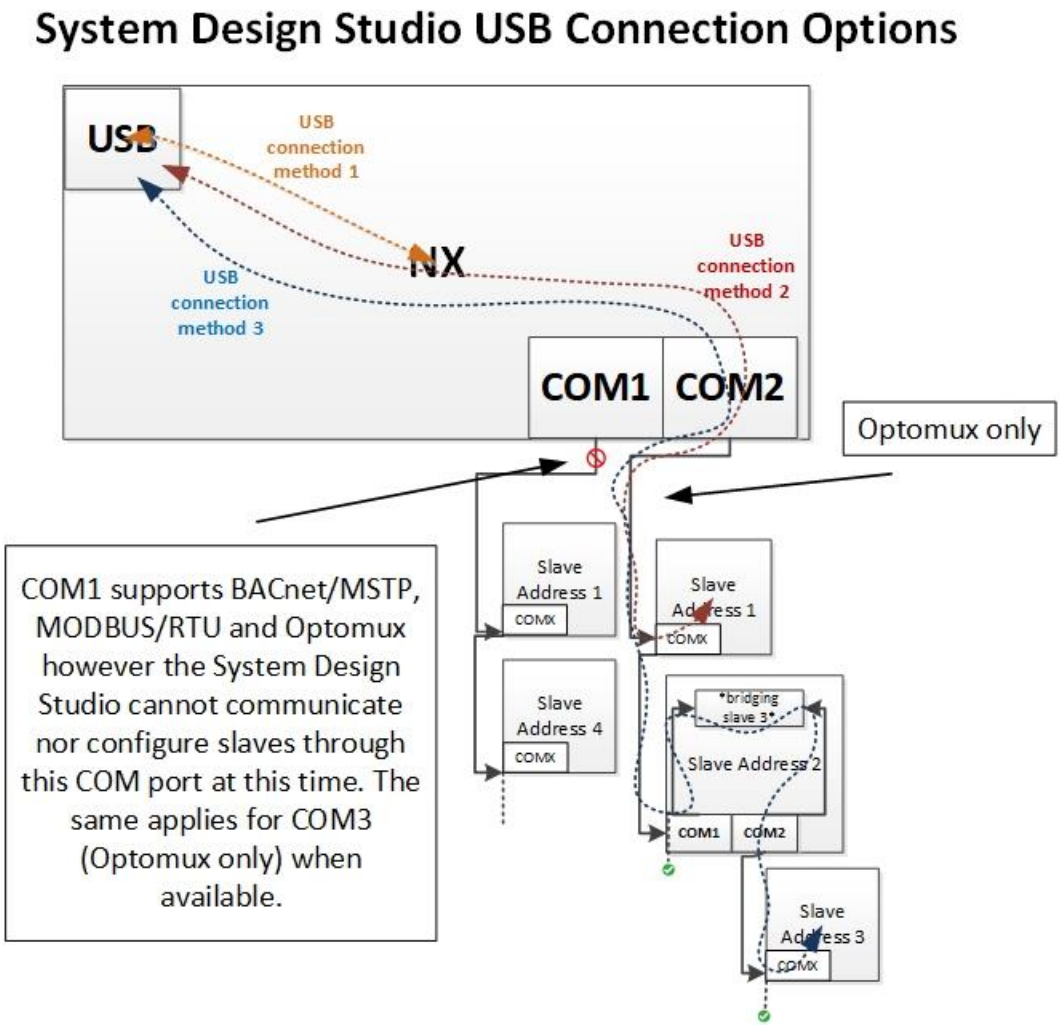
- **Detailed steps to connect to controllers outside of a project using the 6 IP connection methods:**
    1. **IP connection method 1:** Connect to an OpenBAS-HV-NX10 series controller (or an OpenBAS-HV-LEARN training module) that is connected to the OpenBAS-NWK-ETH3's SPI port.
        1. Click the "+" icon in the top left-hand corner, select Connect from the *Action*: dropdown, fill in the IP address and click "Add". The IP address should be in the form of "123.123.123.123:1234" or "www.sampleIP.com:8080"
    2. **IP connection method 2:** Connect to a slave controller, on **COM2** of an OpenBAS-HV-NX10 series controller (or OpenBAS-HV-LEARN training module) that is connected to the OpenBAS-NWK-ETH3's SPI port.
        1. First connect to the master controller following the steps for connection method 1.
        2. Right click the master controller item in the side panel and click "Add slave".

3. Select SPI_NX/COM2 from the channel drop down box.
4. Type in the address of the slave controllers' respective COM port and click "Ok".

3. [1] **IP connection method 3:** Connect to a slave, on **COM2** of a slave, on **COM2** of an OpenBAS-HV-NX10 series controller (or OpenBAS-HV-LEARN training module) that is connected to the OpenBAS-NWK-ETH3's SPI port (commonly referenced as a "slave-of-slave" connection).
    1. First connect to the master controller following the steps for IP connection method 1.
    2. Right click the master controller item in the side panel and click "Add slave".
    3. Select SPI_NX/COM2 from the channel drop down box.
    4. Type in the address of the "slave-of-slave" controllers respective COM port and click "Ok".
4. **IP connection method 4:** Connect to a slave controller that is connected to an OpenBAS-NWK-ETH3's COM1 or COM2 ports.
    1. First either connect to an NX10 series (or OpenBAS-HV-LEARN) controller following the steps for IP connection method 1 or connect to an OpenBAS-NWK-ETH3 following the steps for IP connection method 6.
    2. Right click the recently added controller item in the side panel and click "Add slave".
    3. Select ETH3/COM1 or ETH3/COM2 appropriately from the channel drop down box.
    4. Type in the address of the slave controllers' respective COM port and click "Ok".
5. [1] **IP connection method 5:** Connect to a slave controller, on **COM2** of a slave controller, that is connected to an OpenBAS-NWK-ETH3's COM1 or COM2 ports (commonly referenced as a "slave-of-slave" connection).
    1. First either connect to an NX10 series (or OpenBAS-HV-LEARN) controller following the steps for IP connection method 1 or connect to an OpenBAS-NWK-ETH3 following the steps for IP connection method 6.
    2. Right click the recently added controller item in the side panel and click "Add slave".
    3. Select ETH3/COM1 or ETH3/COM2 appropriately from the channel drop down box.
    4. Type in the address of the "slave-of-slave" controller's appropriate COM port and click "Ok".
6. **IP connection method 6:** Connect directly to an OpenBAS-NWK-ETH3.
    1. Click the "+" icon in the top left-hand corner, select Connect from the *Action*: dropdown, select ETH3 for the Device type, fill in the IP address,

and click “Add”. The IP address should be in the form of “123.123.123.123:1234” or “www.sampleIP.com:8080”

- **Connecting to controllers over USB, outside of Projects** (Only recommended for initial COM port configuration, since scripts will not be saved to main ETH3, see *Creating projects*)**:**
    - **Requirements**
        1. USB connections require a Type A Male to Type B Male USB cable except for the OpenBAS-HV-LEARN controller which requires a Type A Male to Type Micro B USB cable.



    - **Detailed steps to connect to controllers outside of a project using the 3 USB connection methods**
        1. **USB connection method 1:** Connect directly using a controllers USB port.
            1. Connect a USB cable between the PC and target controller.
            2. Click the refresh icon[2.]

2. **USB connection method 2:** Connect through a master controller to one of its **COM2** slave controllers.
    1. Connect a USB cable between the PC and the master controller.
    2. Click the refresh icon[2].
    3. Right click the master controller item in the side panel and click “Add slave”.
    4. Verify SPI_NX/COM2 is selected from the channel drop down box.
    5. Type in the address of the slave controller’s appropriate COM port and click “Ok”.
3. [1] **USB connection method 3:** Connect to a slave controller, on **COM2** of a slave controller, on **COM2** of a master controller with the USB cable connected (commonly referenced as a “slave-of-slave” connection).
    1. Connect a USB cable between the PC and the master controller.
    2. Click the refresh icon[2].
    3. Right click the master controller item in the side panel and click “Add slave”.
    4. Verify SPI_NX/COM2 is selected from the channel drop down box.
    5. Type in the address of the “slave-of-slave” controller’s appropriate COM port and click “Ok”.

[1] *“Slave-of-slave” connections require Optomux protocol and a minimum baud rate of 38400 on both involved fieldbuses.*

[2]*Occasionally, switching a USB cable between controllers too quickly can cause problems in Windows and the USB cable will need to be momentarily disconnected from the PC and reconnected to establish a connection. Alternatively, the COM port can be disabled and re-enabled within Windows Device Manager.*

Setting up your first OpenBAS network

Here are some step by step instructions on configuring a network of 3 OpenBAS controllers to be set up as a Master, Slave, and Slave of Slave. This setup will allow for IP connection methods 4,5, and 6 to be used. An OpenBAS-NWK-ETH3 will be used as the “Main master”, an OpenBAS-HV-NX10P as a Slave and an OpenBAS-HV-NXHALF as a Slave of Slave.

1. Review and follow the installation manuals for wiring power to the controllers.
2. Using appropriate 3 conductor communications cable (1 twisted pair + 1), connect COM1 on the OpenBAS-NWK-ETH3 to COM1 on the OpenBAS-HV-NX10P (connect ref to ref, d+ to d+, and d- to d-).
3. Using appropriate 3 conductor communications cable, connect COM2 on the OpenBAS-HV-NX10P to COM1 on the OpenBAS-HV-NXHALF.
4. Follow the steps to connect to controllers using USB connection method 1:

a. Connect to the OpenBAS-HV-NXHALF
    i. From Configuration->Communication, set:
        1. COM1
            a. Address = 3
            b. Protocol = OPTO22 slave
            c. Baud = 38400
        2. Leave everything else default, and click **Save and restart**.
b. Connect to the OpenBAS-HV-NX10P (*if using 1 USB cable, first remove the previous USB controller from the side panel by right clicking and selecting Remove*).
    i. From Configuration->Communication, set:
        1. COM1
            a. Address = 2
            b. Protocol = OPTO22 slave
            c. Baud = 38400
        2. COM2
            a. Protocol = OPTO22/N2 master
            b. Baud = 38400
            c. Last point to poll = 10 (RMT_1 – RMT_10 will be enabled)
        3. Others
            a. COM1/COM2 bridge
                i. Min address = 3
                ii. Max address = 3
        4. Leave everything else default, and click **Save and restart**.
    ii. From Configuration->Remote points, set:
        1. RMT_1: Address = 3, Object type = Analog Input, Channel = 1
        2. Within a few seconds the RMT_1 present value should start updating.

5. Follow the steps to connect to the OpenBAS-NWK-ETH3, IP connection method 6.
    a. From Configuration->Communication, set:
        i. COM1
            1. Protocol = OPTO22/N2 master
            2. Baud = 38400
            3. Last point to poll = 10 (COM1_RMT_1 – COM1_RMT_10 will be enabled)
        ii. Leave everything else default, and click **Save and restart**.
    b. From Configuration->Remote points, set:
        i. COM1_RMT_1: Address = 2, Object type = Analog Input, Channel = 1
        ii. COM1_RMT_2: Address = 3, Object type = Analog Input, Channel = 1
        iii. Within a few seconds the COM1_RMT_1 and COM1_RMT_2 present values should start updating.
6. Now all 3 controllers can be added to a project, or connected to outside of projects, using IP connection methods 4, 5, and 6.

Configuring network settings and IP address

There are two ways to configure the network settings of an OpenBAS-NWK-ETH3.

1. If the current IP address is known (Hint: try the Discovery feature within the Add controller menu of System Design Studio), browse to http://[IP address]/protect/config.htm and change the network settings as needed.
    a. **Warning**: Changing the network settings incorrectly can cause the OpenBAS-NWK-ETH3 to become unreachable requiring physical access to the controller. It is recommended to only change network settings while physical access to the controller is available.
    b. In order to access the web pages, you will need to log in. The default username and password are: Username: admin Password: mircom.
    c. The default IP address is 192.168.1.51. Please see LT-6630 for details on the various methods of determining the IP address or continue below for the USB flash memory method.
2. Alternatively, the network settings can be viewed and modified using a USB flash memory stick:
    a. First insert a cleared, FAT32 formatted USB flash memory stick into the OpenBAS-NWK-ETH3 for 10 seconds.
    b. After 10 seconds, remove the USB flash memory stick from the OpenBAS-NWK-ETH3 and reinsert into your PC.
    c. Use any text editor to open the ROOT/_OPENBAS/IPCONFIG.INI file.
    d. After making changes to IPCONFIG.INI, be sure to change "[N] = APPLY_CHANGES" to "[**Y**] = APPLY_CHANGES" and click save.
    e. Reinsert the USB flash memory stick into the OpenBAS-NWK-ETH3 and wait for it to automatically reset.

Upgrading firmware

The firmware can easily be field-upgraded on all "NX" controllers, the OpenBAS-NWK-ETH3 and the OpenBAS-HV-LEARN. The general process for all controllers involves using 1 of 3 Windows based "bootloader" programs, putting the controller into "bootloader mode" (using the *Download* and *Reset* buttons), and connecting a USB cable between the controller and Windows PC. For more information, see LT-6630 – Driver Installation and Firmware Upgrade Procedures.

1. Connect A USB cable between the controller and the PC.
    a. USB type B for OpenBAS-NWK-ETH3 and "NX" controllers (except OpenBAS-HV-LEARN).
    b. Mini USB for OpenBAS-HV-NX10D's *display* board.
    c. Micro USB for OpenBAS-HV-LEARN.
2. Run the appropriate* bootloader program from *C:\OpenBAS_NX5 version X.XX.X\Bootloader*.
    a. Bootloader 8-bits/**HIDBootloader.exe** for "NX" controllers and OpenBAS-HV-LEARN.
    b. Bootloader 32-bits/**PIC32UBL.exe** for OpenBAS-NWK-ETH3 and OpenBAS-HV-NX10D's *display* board.

c. *There is a link to the appropriate bootloader program at the bottom of the Configuration->Information sub tab however this tab will need to be opened before putting the controller into bootloader mode.

3. Put the controller into "Bootloader mode".
    a. Press and hold the Download/DWNLD button.
    b. While holding Download, press and release the RESET button.
4. Depending on the Bootloader program used the next steps are slightly different.
    a. In HIDBootloader.exe:
        i. If all steps were followed properly, text will appear saying that a device was detected.
        ii. Click Open Hex file and browse to and select the latest firmware being sure to select the appropriate firmware for the controller.
        iii. Check the box to "**Allow Configuration Word Programming**".
        iv. Click Program/Verify and wait for a confirmation that the process is complete.
        v. Click Reset Device.
    b. In PIC32UBL.exe:
        i. Check the box to Enable USB connections (leave VID and PID to defaults).
        ii. Click Connect.
        iii. If all steps were followed properly, text will appear saying that a Device was connected.
        iv. Click Load Hex file and browse to and select the latest firmware being sure to select the appropriate firmware for the controller.
            1. The latest firmwares can be found after installing the System Design Studio in the C:\OpenBAS_NX5 version X.XX.X\Bootloader_HEXfiles\OpenBAS\ directory.
        v. Click Erase-Program-Verify and wait for confirmation that the process completed successfully.
        vi. Click Run Application.

Discovering OpenBAS gateways

OpenBAS gateways such as the OpenBAS-NWK-ETH3 can be discovered if they are on the same local network as the PC. After clicking the "+" icon in the top left-hand corner of the side panel, select the PC's IP address from the *Interface* dropdown (as seen from the OpenBAS gateway) and click Discover. Any OpenBAS gateways that are discovered will show their name and IP address within the *IP address* dropdown.

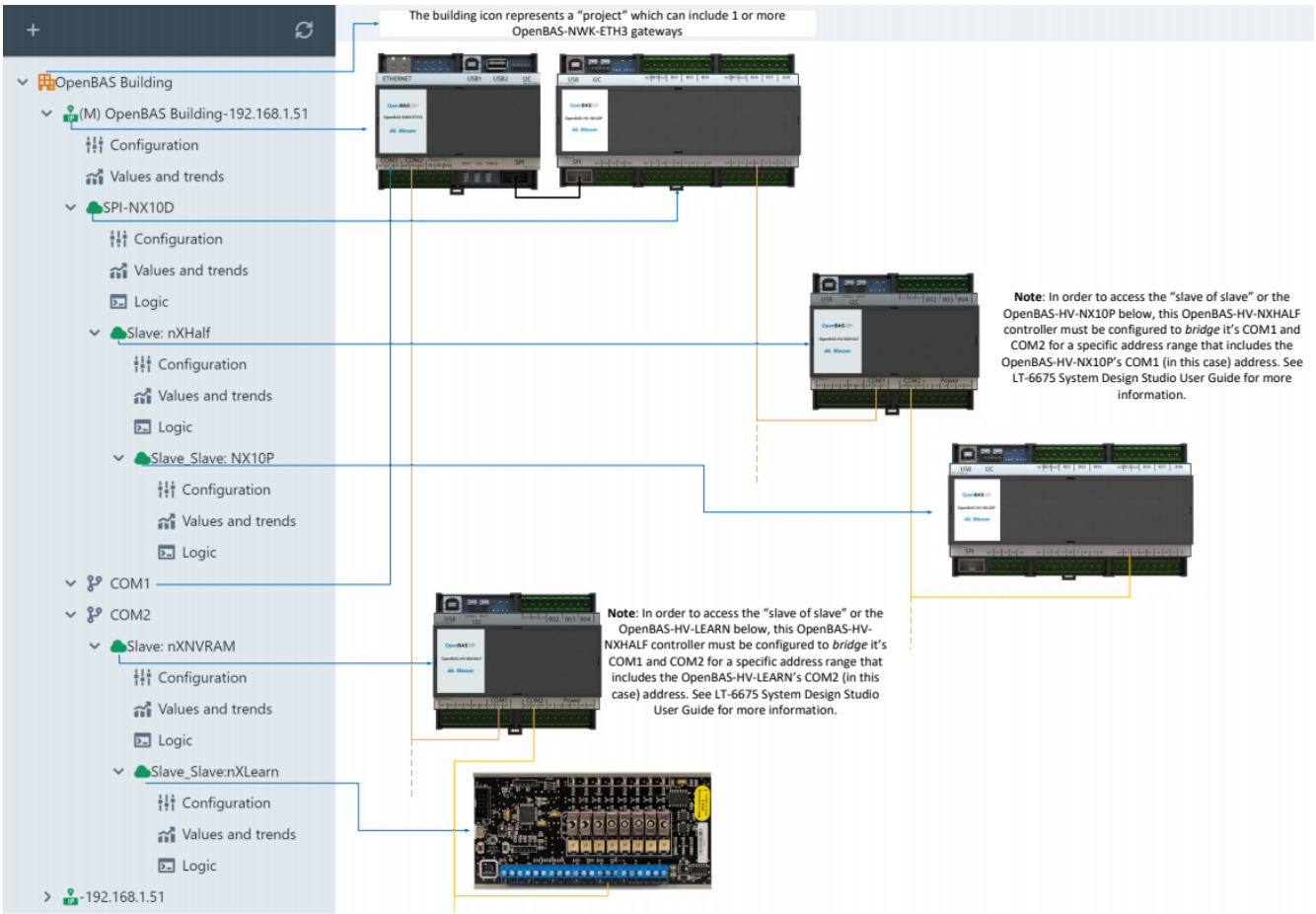
- Discovery uses BACnet/IP which requires a BBMD to work across subnets.
- Once discovered, the IP address can be used to make a one-off connection to the OpenBAS-NWK-ETH3 or SPI-NX controller (if available), import projects, or create projects.

Creating projects

Projects are folders within the side-panel that organize many controllers according to how they are connected. A Project can be thought of as a list of controllers along with information on how they are connected. Projects require at least one OpenBAS-NWK-ETH3 and it's highly recommended to install a USB flash memory to it. After clicking the "+" icon in the top left-hand corner:

1. Select *Create project* from the *Action* dropdown.
2. Enter a name for your project.
3. Browse to and select a Project directory (must be empty).
    a. All your script files will be created here.
4. Fill in an OpenBAS-NWK-ETH3's IP address and click Add.
    a. To work offline, enter any temporary IP address.

The System Design Studio will create a project folder (identified by a building icon) and automatically add OpenBAS-NWK-ETH3 (IP connection method 6) and SPI-NX (IP connection method 1) controller items to the project.



- The first OpenBAS-NWK-ETH3 added to a project is marked with an "(M)" for Main, indicating users should install a USB flash memory stick to it, to facilitate on-site backup of the Project (controller list) and script files.
    - The main ETH3 can be changed when adding a new OpenBAS-NWK-ETH3 to a project or by right clicking an existing OpenBAS-NWK-ETH3 item in the project and selecting *Edit ETH3*.
- An orange building icon indicates the project (controller list) has not been saved, or there were changes since it was last saved, to the Main OpenBAS-NWK-ETH3 (Requires USB flash memory stick).
    - The USB flash memory stick should be formatted as FAT32 before first use.
    - The project will automatically be saved to the main OpenBAS-NWK-ETH3 when configuring multiple controllers from script, see *Batch programming*.
    - By default, the project will automatically be saved to the main OpenBAS-NWK-ETH3 when configuring a single controller from script however there is a checkbox "Save script to Main ETH3", that can be unchecked, to disable this.
    - The project can be saved to the main OpenBAS-NWK-ETH3 by right clicking the project and selecting *Save project to Main ETH3*.
    - Only one Project can be saved to an OpenBAS-NWK-ETH3 at a time, meaning each time the Project is saved, the previous Project is overwritten.
    - Once saved to the Main OpenBAS-NWK-ETH3, Projects can later be imported from it, see *Importing projects*.
- An asterisk (*) is appended to the end of the Project name when the Project has not been saved to disk, or there are changes since it was last saved to disk, in the selected Project directory.
    - The Project is **never** automatically saved to disk, aside from when initially importing a Project from an OpenBAS-NWK-ETH3.
        - This allows users to remove the project, and re-import from disk in the event they accidentally remove one or more required controllers.
    - To save or backup a Project to disk, right click the Project and select Save project to disk.
- A Lock icon to the left of the Project name and every controller indicates that the main OpenBAS-NWK-ETH3 has not been logged into. To login, right click the Project and select Login to Main ETH3. After successfully logging in, a Logout from Main ETH3 option will be available.
- Controller icons (both IP and cloud) will turn green once the controller is connected and will turn orange when the controller is not responding.
    - At any time, you can right click a controller item within a project and select Update status to force check if the controller is responding.
- Additional OpenBAS-NWK-ETH3's can be added by right clicking a project folder and selecting Add ETH3.
- Slave* and "Slave of slave" controllers can be added by right clicking on their respective master controllers within the project and selecting Add slave, or Add slave of slave, accordingly.

- *OpenBAS-NWK-ETH3 COM1 and COM2 slaves can be added by right clicking the COM1 and COM2 folders within the appropriate OpenBAS-NWK-ETH3 controller and selecting Add slave.
- Multiple controllers with consecutive addresses can be added by entering the Address of the first controller, and the total Quantity to add.
  - If a controller address already exists it will be skipped.
- Default name is the same as when right clicking an offline (black icon) controller item, and selecting Rename (temp.). See *Naming projects and controllers*.

Removing projects and controllers

Right click a project or one or more controller items in the side panel and click “Remove” or “Remove and archive”.

- Selecting Remove on a Project will remove the Project item from the side panel. Be sure to save the Project to the Main OpenBAS-NWK-ETH3 or to disk, or both, if you intend to import it later.
- Selecting Remove and archive on one or more controllers within a Project will remove the controller items (and any associated slaves) from the Project in the side panel and archive any script files that may exist for the selected controllers and any of their associated slaves. This does not delete the script files on the main OpenBAS-NWK-ETH3’s USB flash memory, but they may be overwritten at any time.
  - When archived, the folders for each controller are moved from the project directory into the /Archive/removed/YYYY-MM-DD/ folder.
  - To restore a script, copy the entire controller folder back into the root of the project directory. **Do not change the folder name.**
- Selecting Remove on an IP connection controller outside of a Project will remove the controller item from the side panel.
- Selecting Remove on a USB connection will release or close the COM port in Windows in addition to removing the controller item from the side panel.

Naming projects and controllers

Right click a project or controller item from the side panel, click “Rename”, or “Rename (temp.)”, enter the desired name and press Enter.

- Controller names are limited to 18 characters, and must only include alpha-numeric characters or underscores “_”.
- Rename (temp.) will show the new name in the side panel, at least until the first time the controller becomes online (green icon).
  - When the controller first becomes online, if it has no name configured, it will be set to the current name as shown.
  - If the controller already has a name configured, the name shown in the side panel will be updated to reflect the controllers previously configured name.

- The controller's name is technically stored in the 200th index/ID of the Point names list within the controller's memory. However, the controller's name is hidden from the **Values and trends->Names** table. Knowing this allows the controller's name to be displayed on the LCD display (See LCD display, alarms and configuration).

Importing projects

Projects can be imported from an OpenBAS-NWK-ETH3 with a USB flash memory connected, or projects can be imported from disk, provided a project has previously been saved to an OpenBAS-NWK-ETH3 or to disk respectively.

- To Import a Project from an OpenBAS-NWK-ETH3, after clicking the "+" icon in the top left-hand corner:
  - Select Import project from the *Action:* dropdown.
  - Select the ETH3 radio button.
  - Check the "Save all script files to project directory" checkbox to automatically Get scripts for all controllers previously Configured from script.
    - Alternatively, a Get scripts function is available from each controller's right click context menu.
  - Browse to and select a Project directory.
    - The selected Project directory Must be completely empty.
  - Fill in the IP address for the main OpenBAS-NWK-ETH3 used in the Project.
  - Fill in the Username and Password fields.
    - The defaults are: Username: admin Password: mircom.
    - The username and password can be changed from the OpenBAS-NWK-ETH3's web pages using any web browser.
- To Import a Project from disk, after clicking the "+" icon in the top left-hand corner:
  - Select Import project from the *Action:* dropdown.
  - Select the Disk radio button.
  - Browse to and select the Project directory that was used to save the Project to disk previously.
    - To import a project file backed up from previous versions of the System Design Studio, rename it to "project.json", move it into a new folder, and select the new folder as the Project directory.
  - Leave the "IP address:" textbox blank to import from the same IP address used previously in the Project, or fill in a new IP address if it has changed. This can be especially useful when connecting through a DDNS address or through a port forwarded IP address.

Get scripts and Force save scripts

Each controller's right click menu has a Get scripts option. This allows the user to get the current script file from the Main ETH3's USB memory for the respective controller.

- Speed up Import project from ETH3 and only Get scripts from the controllers you are actively working with.
- Revert your local changes to the currently active (last configured) script on the controller.

The Force save scripts option facilitates backing scripts up to the main ETH3 for older sites when it wasn't automatic, and for any controllers that may have been configured from script outside of a project (e.g. USB). This option can be found by right clicking any controller item in the side panel.

Remotely accessing OpenBAS controllers using Cloudflare tunnels

System Design Studio (SDS) supports secure, remote access to OpenBAS controllers via Cloudflare Tunnels. This integration allows authorized users to **connect to** controllers from anywhere, with no VPNs or special apps required. When SDS detects an OpenBAS **controller** behind a Cloudflare Tunnel, it will automatically present a login window for users to request and enter their one-time passcode. Access is **typically** controlled through user email addresses, providing simple and secure authentication.

- No VPN Required: Technicians and end-users connect using a secure site URL (e.g., site1.mydomain.com) — no need for VPN apps or complex setups.
- Seamless Authentication: On first access, SDS automatically detects Cloudflare Tunnel protection and prompts the user to enter a one-time passcode sent to their whitelisted email address.
- Access Control: Only the email addresses whitelisted in Cloudflare Access can log in. Each login uses a single-use code, emailed upon request.
- Account & Domain Setup:
    - Create a free Cloudflare account (supports up to 50 users).
    - Register a (paid) domain in Cloudflare, and define unique subdomains for each site (such as site1.mydomain.com).
    - Map these subdomains to the local network IP address of the Main OpenBAS-NWK-ETH3 at each site.
    - Device Hosting: The Cloudflare tunnel daemon (cloudflared) should run on an always-on device (PC, single-board computer, compatible router, or LTE router for sites without broadband). Unofficial OpenWRT ports may be available.
- For step-by-step instructions, refer to LT-6797 — OpenBAS Cloudflare Tunnel Setup Guide .

Recovering EZ-Script files from USB Flash memory

While script files are automatically sent to the main OpenBAS-NWK-ETH3 of a project (when a USB flash memory has been installed), they are compressed into a *.zip archive, and have a somewhat cryptic name. The easiest way to recover script files is by removing the USB flash memory and connecting it to a PC. Script archives are stored in the /OpenBAS/ folder with a name consisting of 8 numbers and letters.

- The first 3 characters are a hexadecimal, 1-based number that increments for each OpenBAS-NWK-ETH3 added to a project.

- The fourth character is a number between 0 and 3, where 0 refers to the OpenBAS-NWK-ETH3 itself, 1 refers to the SPI-NX port, 2 refers to the COM1 port, and 3 refers to the COM2 port.
- The fifth and sixth characters are a hexadecimal number referring to the address of a slave controller.
    - If this number is 255 (0xFF), and the fourth character is 1 (SPI-NX), then the script is for the SPI-NX controller itself.
- The seventh and eighth characters are a hexadecimal number referring to the address of a slave of slave controller.
    - If this number is 0 (zero) then the script is for the previously identified slave controller, otherwise the script is for the slave of slave controller with this number as it's address.

Importing and exporting IP connection lists

Deprecated as of 1.1.0 due to project support

After connecting to multiple controllers using any of the IP connection methods, the up-arrow button located at the top of the side panel can be used to save a *.json file to disk that allows for quick and easy reconnection to the same set of controllers at a future time, or from another computer with the System Design Studio installed.

- To export a list of the current IP based connections, click the up-arrow button, browse to an appropriate directory, enter a descriptive file name and click Save followed by Ok. The file will be created.
- To import a previously exported IP connection list, click the down-arrow button, browse to the appropriate *.json IP connection list file, and click Open followed by Ok. The System Design Studio will attempt to connect to all controllers, populating the side panel with each controller that is responding.
- USB connections are not included in the exported file and any existing USB based connections will remain in the side panel after importing an IP connection list file.
- Existing IP based connections will remain in the side panel after importing an IP connection list file.

Software resetting controllers

Right click a controller item from the side panel and click "Restart".

- All "NX" controllers will essentially power cycle and re-initialize according to the settings in the *Configuration->Power up init* tab. See Power up initializations for more information.
- Alternatively, pressing the RESET button will also power cycle and re-initialize according to the settings in the *Configuration->Power up init* tab.

Power up initializations

The power up initializations let the user decide what the controller should do with its volatile memory (memory that is lost without power) during each power cycle and can be found under **Configuration->Power up init**. Typically, these settings relate to whether the controller has a backup battery installed

to retain the volatile data during a power cycle or short-term power outage. If there is no battery installed, then the present value of points such as RES_BIT or RES_FLT will automatically be Set to zero after a power cycle (regardless of which options are checked).

- "Erase bit registers" will Set the present value of all RES_BIT type points to 0 (zero) during the power on process.
- "Erase float registers RES_FLT 1-40, binary input 1-40, analog input 1-40, analog output 1-10" will Set the respective points' present value to 0 (zero) during the power on process.
- "Erase local overrides" will clear any local overrides (priority 1) to Binary outputs (Set using a display) during the power on process.
- "Erase communication overrides to remote points of binary or analog outputs" will clear any communication overrides (priority 5) to Binary or Analog outputs during the power on process.

Performing a factory reset

A factory reset will return the controller to the state it left the factory. The Factory reset window can be found by right clicking any controller item from the side panel and selecting *Factory reset.* There are two main sections in the Factory reset window, general and remote points. The general section allows for individual sections of the controller's memory to be factory reset, and the remote points section allows for the various sets of remote points to be individually factory reset as well. For the general section's "General configuration" option to be enabled, the controller must be connected using IP or USB connection method 1 to prevent a user from accidentally modifying the communication settings of a remote controller and therefore causing it to stop responding.

- To perform a complete factory reset through software:
    1. Select all remote points in the remote points section and click "Clear".
    2. Once the remote points have been cleared, in the general section, click the "All" button followed by the "Clear and reset" button. The controller will now be returned to the factory default state after initiating a software reset.
- To perform a partial factory reset through software:
    1. First, if needed, factory reset any remote points in the remote point section by checking the box next to the appropriate sections and clicking "Clear".
    2. Next select the appropriate sections in the general section followed by the "Clear and reset" button. The controller will reset the respective sections to factory defaults after initiating a software reset.
- To perform a hard reset for the OpenBAS-NWK-ETH3:
    1. Press and release the Reset button, and after 1 second press and hold the Download button. You will see the LEDs begin blinking rapidly, release the button after the LEDS stop blinking rapidly.

Memory overview

There are a few different memory options with respect to OpenBAS hardware. All OpenBAS-XX-**NX**XXX controllers (also referred to as "NX" controllers) as well as the OpenBAS-HV-LEARN module support the

standard database however the OpenBAS-HV-NX10D, and any controller with a memory expansion accessory installed, will have an expanded database. **The OpenBAS-ACC-32KNV cannot be installed on an OpenBAS-HV-NX10D**. The OpenBAS-NWK-ETH3 database has some differences from the standard database and should be reviewed.

- The standard "NX" database:

Feature	Amount
Analog inputs (AI)	1..40
Binary inputs (BI)	1..40
Analog outputs (AO)	1..10
Binary outputs (BO)	1..40 + 41..60 (lighting group control points)
Internal bytes (ADB)	1..100
Internal integers (ADI)	1..100
Internal floats (ADF)	1..100
PLC1	400 instructions
PLC2 *see Dynamic memory	400 instructions *see Dynamic memory
PLC3 *see Dynamic memory	400 instructions *see Dynamic memory
Point names	199
Alarms	8
Event log	200
Trends *see Dynamic memory	0..16 *see Dynamic memory
Script/file storage *see Dynamic memory	0-16KB *see Dynamic memory
Result bits (RES_BIT)	1..255
Result floats (RES_FLT)	1..40
Remote points (RMT)	1..50
Timers (TMR)	1..16

- The standard "NX" database with the OpenBAS-ACC-32KNV memory expansion (Changes from NX(standard) in **green**):

Feature	Amount
Analog inputs (AI)	1..40
Binary inputs (BI)	1..40
Analog outputs (AO)	1..10
Binary outputs (BO)	1..40 + 41..60 (lighting group control points)
Internal bytes (ADB)	1..100
Internal integers (ADI)	1..100
Internal floats (ADF)	1..100
PLC1	400 instructions
PLC2	**400 instructions**
PLC3	**400 instructions**
Point names	199
Alarms	8
Event log	200

Trends	**8**..16 *see Dynamic memory
Script/file storage	**16-24KB** *see Dynamic memory
Result bits (RES_BIT)	1..255
Result floats (RES_FLT)	1..40 **+51-255** *Note 41-50 are **not** available
Remote points (RMT)	1..50 **+51-255**
Timers (TMR)	1..16

- The OpenBAS-HV-NX10D database (changes from NX(standard) in **green**):

Feature	**Amount**
Analog inputs (AI)	1..40
Binary inputs (BI)	1..40
Analog outputs (AO)	1..10
Binary outputs (BO)	1..40 + 41..60 (lighting group control points)
Internal bytes (ADB)	1..100
Internal integers (ADI)	1..100
Internal floats (ADF)	1..100
PLC1	400 instructions
PLC2	**400 instructions**
PLC3	**400 instructions**
Point names	199
Alarms	8
Event log	200
Trends *see Dynamic memory	0..16 *see Dynamic memory
Script/file storage	**16-112KB** *see Dynamic memory
Result bits (RES_BIT)	1..255
Result floats (RES_FLT)	1..40 **+41-255** *Note 41-50 **may** be available
Remote points (RMT)	1..50 **+51-255**
Timers (TMR)	1..16

- The OpenBAS-NWK-ETH3 database

Feature	**Amount**
PLC1	400 instructions
PLC2	400 instructions
Point names	199
Internal bytes (ADB)	1..100
Internal integers (ADI)	1..100
Internal floats (ADF)	1..100
Result bits (RES_BIT)	1..255
Result floats (RES_FLT)	1..255
Remote points (RMT)	1..2000
Timers (TMR)	1..16

Dynamic memory

Dynamic memory allows the default Trend storage to be *traded* for extra script/file storage or PLC instruction storage on "NX" controllers. Dynamic memory can be configured in **Configuration->Dynamic memory.** It is recommended to only change these options directly after a full factory reset since changing them will corrupt whatever data is currently in the respective memory storage areas.

- Base memory controllers (all OpenBAS-XX-**NX**XXX controllers + OpenBAS-HV-LEARN without memory expansion accessories).
    - Section one: Trends 1-8 can be traded for +8KB of script/file storage.
    - Section two: Trends 9-12 can be traded for +4KB of script/file storage or PLC3.
        - OpenBAS-HV-NX10P controllers must use firmware SOB-1101EN (not SOB-1101 which is intended for the OpenBAS-HV-NX10D).
    - Section three: Trends 13-16 can be traded for +4KB of script/file storage or PLC2.
        - OpenBAS-HV-NX10P controllers must use firmware SOB-1101EN (not SOB-1101 which is intended for the OpenBAS-HV-NX10D).
- Controllers with the OpenBAS-ACC-32KNV memory expansion (includes 16 KB of script/file storage by default). Note that the OpenBAS-ACC-32KNV cannot be used on the OpenBAS-HV-NX10D.
    - Section one: Trends 1-8 cannot be traded.
    - Section two: Trends 9-12 can be traded for +4KB of script/file storage.
    - Section three: Trends 13-16 can be traded for +4KB of script/file storage.
- OpenBAS-HV-NX10D (Assuming the Dual core display is connected, includes 16 KB of script/file storage by default, PLC2, and PLC3).
    - Section one: Trends 1-8 can be traded for +32KB of script/file storage.
    - Section two: Trends 9-12 can be traded for +32KB of script/file storage.
    - Section three: Trends 13-16 can be traded for +32KB of script/file storage.
- Note that Section three on the OpenBAS-HV-NXSMS is used for SMS text message storage and cannot be traded.

Viewing and synchronizing a controller's real time clock

In **Configuration->Information** "Time" and "Date" shows the current time and date of the real time clock in the controller. A controller's real time clock can be synchronized with your PC by right clicking the controller in the side panel and clicking "Sync time with computer". Additionally, upon adding a controller, the System Design Studio will pop up asking the user to sync the controller's time if it is off by more than 5 minutes.

- Slave controllers will have their real time clock synchronized with their master controller periodically if they have at least one point configured as a remote point on the master controller.

- Controllers will backup the current time and date into non-volatile memory every 15 minutes meaning the controller time will only ever be off by a maximum of the time the controller was without power (or battery backup) + 15 minutes.
- When OpenBAS controllers are configured on a BACnet network, the BACnet time synchronization feature can be enabled on a master controller such as a JACE-8000 or from the System Management Studio.
- Only the OpenBAS-HV-NX10 series controllers and the OpenBAS-LC-NX12R lighting controller come with a rechargeable backup battery which will keep track of time even when the controller loses power, for up to seven days. Every controller has a connector to allow a battery to be installed.
    1. Allow for 24 hours for the battery to fully recharge.

Date and time settings

In **Configuration->Date and time**, holidays, daylight savings, daily clock adjustment and time mapping options are available.

- Holidays
    - Up to 25 holidays can be configured for specific days of the year. Holidays are used within Configuration->Schedules (see Creating schedules).
    - To remove a holiday, hover over the schedule you want to delete and click the “x” icon that appears to the right of the date.
- Daylight savings time
    - Click the enable button to turn on daylight savings
    - Use the “+1 start” date and time settings to adjust the date and time that the clock should be adjusted forward 1 hour.
    - Use the “-1 start” date and time settings to adjust the date and time that the clock should be adjusted backwards 1 hour.
- Time mapping (NX controllers only)
    - Time mapping allows various time related points to be mapped to result float (RES_FLT) type points (RES_FLT_1-RES_FLT_**245**) that can then be used within the PLC. These points are read only.
    - Check the box next to any required time related data and then choose any available RES_FLT channel.
    - “All time data” will use a consecutive series of 6 RES_FLT points to store the current minute, hour, day, month, year and weekday (0=Sunday to 6=Saturday).
    - A RES_FLT channel of 255 will disable the respective time data point’s mapping.
- Daily clock adjustment (NX controllers only)
    - Over time, the real time clock can drift and become inaccurate if it is not being synced periodically. The daily clock adjustment slider can be set between -25 seconds and +25 seconds per day.

Creating schedules

In **Configuration->Schedules,** up to 400 schedules can be created (200 on OpenBAS-NWK-ETH3). Before testing any schedules, it is important to make sure that the controller's time is correct, and to think about how the time will be synchronized (see Viewing and synchronizing a controller's real time clock). After clicking on the Schedules tab, the System design studio will begin reading all schedules in the background.

- Only enabled or previously configured schedules will be shown.
- Click the "+" icon to create a new schedule.
- To delete a schedule, hover over and click the garbage can icon.
- Each schedule can be set to run on a single day of the year, certain days of the week, or on every configured Holiday (see Date and time settings to configure Holidays).
- Each lighting group control point (whether using RES_BIT_1-20 or BO_41-60 for priority control) can have up to 10 schedules. The System design studio will not let you create an 11th schedule for any lighting group. RES_BIT_1 and BO_41, for example, can both be used separately (requires two lighting group instructions, see LightG overview) although only 10 schedules can be created in total to control either RES_BIT_1 or BO_41.
- Up to 200 schedules can be created to Set ADB, ADI, ADF, and RES_FLT type points as well as RES_BIT_21-RES_BIT_255 and BO_1-BO-40.
- Schedules have the least priority, equivalent to BACnet priority 11. This means that points previously Set by schedules can still be Set by PLC instructions or overrides.
- Any schedules for BO or RES_BIT type points can be configured to turn both on and off at specified times using only 1 schedule.
    - After selecting the BO or RES_BIT point type when creating a schedule, enter 1 into the "Set value" text box and a drop-down menu will appear with options for Normal, Period, or Period force.
        - Normal will Set the points state to 1 at the specified time.
        - Period will Set the points state to 1 at the specified Start time and 0 (zero) at the specified End time.
        - Period Force will Set the points state to 1 every minute until the End time where it will be Set to 0 (zero).

Configuring and calibrating inputs

Most OpenBAS controllers have universal inputs which allow users to connect either an analog or binary signal into them. To specify that a universal input is being used as either a binary or analog input, select the appropriate Type within **Configuration->Inputs outputs->Universal Input calibration**. In addition, be sure to correctly reference either the analog or binary point within the PLC, schedules etc. For example, when there is a voltage of 3V DC on universal input 1, analog input 1's (AI_1) present value (if Type set as 0-10VDC input) will read 3 Volts but binary input 1's (BI_1) present value will read 0 (zero). For more information on wiring and connecting different types of inputs, see the installation manual (for any OpenBAS controller with universal inputs).

- Universal inputs can be calibrated by going to **Configuration->Inputs outputs** and selecting the **Universal input calibration** option from the main dropdown box.
    - From this table, users can select input types, define units, and further calibrate an analog input's present value property with either an offset value or multiplier (depending on the input type).
    - When changing the offset or multiplier, the respective analog input's present value property will be updated at a faster rate to allow the user to better match the real time value of a multimeter or other measuring device.
    - For help wiring Universal inputs correctly, click the ? icon located beside the input's selected Type to see a common wiring diagram with proper dip switch settings.
- For noisy or unstable inputs, within the **Configuration->Inputs outputs->General** option and under the **Custom input averaging** header, the universal inputs 1 through 8 can be averaged over a multiple of 200ms for analog inputs, and a multiple of 50ms for binary inputs. In addition, there is a "global AI averaging" calibration parameter located under the **General IO configuration** header which affects all 40 analog inputs.
    - By default, Custom input averaging is disabled, and the values are configured automatically based on input Type.
    - When analog inputs 1 through 8 have their individual input averaging calibration parameters set to 0 (zero), they will use the global AI calibration parameter instead. This requires checking the *Enable custom input averaging* checkbox and manually setting each value.
    - The OpenBAS-HV-NXSF, starting with firmware version 3.15.5, has separate input averaging parameters for its Analog and Binary inputs.
- Any NTC thermistor can be used for temperature sensing applications since OpenBAS controllers are not limited to a specific type or manufacturer and up to 3 different thermistors can be used on each controller with high accuracy. To support the wide range of thermistors available, some configuration is needed.
    - For most 10K Type 3 thermistors, select the 10K Type 3 NTC Thermistor °C (or °F) option from the Type dropdown within the **Configuration->Inputs outputs->Universal input calibration** table. No other configurations are needed.
    - For any NTC thermistors aside from 10K Type 3 NTC thermistors, a custom thermistor Type needs to be selected and the A, B and C parameters need to be configured.
        1. Select a Custom Thermistor Type within the **Configuration->Inputs outputs->Universal input calibration** table.
            - When selecting a Custom Thermistor Type, "Ext. 12V" (externally powered) or "Int." (internally powered) refers to how the thermistor will be wired.
            - It is highly recommended to use the "Ext. 12V" wiring (click ? icon for diagram) even if the controller does not provide a 12VDC source and it means purchasing an external 12VDC power supply. 12VDC will provide

much higher resolution (smaller noticeable temperature increments) compared to the internal 5V supply.

2. Configure the A, B and C parameters for Custom Thermistor A and/or Custom Thermistor B within the **Configuration->Inputs outputs->General** option under **Custom thermistor configuration (Steinhart-Hart)** header:
    - Select from available presets or select Manual.
    - To manually configure a thermistor, you will need to reference its datasheet. If the Steinhart-Hart A, B and C parameters are given on the datasheet, they can be entered directly otherwise they can be calculated using the temperature/resistance chart or table. If using a temperature/resistance chart or table, three sets of temperature and resistance pairs are needed and should be chosen around 5%, 50% and 95% of your expected temperature range.
        1. For a temperature range of 0-100 degrees, enter the resistance at 5 degrees, 50 degrees and 95 degrees, respectively.
        2. Click calculate to populate the values for the A, B and C parameters.
        3. Click Save. Note that the temperature and resistance values used to calculate the A, B and C parameters will not be saved.
3. For unstable, external 12VDC supplies, the temperature accuracy can be improved by wiring the output of the 12VDC supply into one of UI_1-8, and selecting that Analog input Channel from the *Custom thermistor external 12V reference* dropdown within the **Configuration->Inputs outputs->General** option under the **General IO configuration** header. This will allow the temperature conversion calculation to use the real time supply voltage instead of assuming a constant 12.0V (default behaviour).

- Various other settings can be found within the **Configuration->Inputs outputs->General** option under the **General IO configuration** header:
    - The “Invert local binary input logic” option will change binary input 1 through 8 to use “active-low” logic which is useful when all the universal inputs (which are being used as binary inputs) have their internal pull up resistor enabled and are wired through equipment status relays to ground. The internal pull up resistors can be enabled with dipswitches or jumper pins (depending on the controller) and are referenced in the OpenBAS controller installation manuals (under the section for connecting resistive temperature sensors).
        1. This feature is not supported on the OpenBAS-HV-NXSF controller since there are no pullups available for the Binary inputs.
    - “OpenBAS-HV-NXSF CT input noise level” will subtract a raw 12-bit number from the AI_1 to AI_8 raw input values (that are configured as “Current Transformer (1000:1)” type inputs) on the OpenBAS-HV-NXSF controller. This is useful for current metering applications where the connected CTs have some noise due to large nearby electrical

currents. This setting can take a value between 0 (zero) and 50 otherwise it will be disabled.

1. After installation, temporarily configure all AI's that are connected to CTs as "12-bit ADC" type inputs.
2. Find the maximum of their present values, while the measured circuits are off.
3. Enter this maximum value into the "OpenBAS-HV-NXSF CT input noise level" textbox and click Save.
4. Configure respective Analog inputs back to "Current Transformer (1000:1)" type inputs.

- The "Local output state when offline as slave" section will automatically Set local Binary outputs and Analog outputs to user configured values after a user configured amount of time has passed without the master communicating.
    1. This requires the slave device to be configured as a Slave type remote point (using addresses 100-103) in the master controller.
    2. The slave must be connected through it's COM1 or COM2 ports.
    3. Not currently supported when configured as a BACnet slave on COM1.
- The "ADF to store NXVAV pressure sensor offset and trigger Auto zero" will define the ADF channel to be used for zeroing the pressure sensor reading on the OpenBAS-HV-NXVAVAP or OpenBAS-HV-NXVAVP. The present value of the selected ADF_X will show the current offset and any Set commands to this point will be ignored with two exceptions. Setting this point to -10.10 will trigger the auto zero function only if the current pressure reading is less than 0.1 inH2O. As a backup, setting this point to -10.25 will trigger the auto zero function if the current pressure reading is less than 0.25 inH2O.

- See "Naming points" for information on creating, editing and deleting point names for inputs.

Airflow balancing on the OpenBAS-HV-NXVAV Series

The OpenBAS-HV-NXVAVAP and OpenBAS-HV-NXVAVP are often used in pressure independent control strategies which require accurate airflow measurement. The internal differential airflow sensor is used along with a K-factor to calculate airflow. This K-factor (or sometimes an amplification factor which can be used to calculate K) is often provided by the VAV box manufacturer, but onsite calibration is commonplace due to increased comfort and energy savings. While this calibration can be done directly in the System Design Studio, we have a mobile-friendly **VAV Tool web app hosted on every OpenBAS-NWK-ETH3** at [IP address]/protect/vavtool.htm (with firmware/webpages 3.18.0 or later). **It's important to note that this app will only work for the supplied default VAV scripts, as each point and form button is mapped to specific points defined in the scripts.** The intention for this app is that the integrator can set up a Cloudflare tunnel to allow access to the tool through the internet and simply share the website URL with the balancing technicians. The balancing technicians can access the tool through their phones browser without needing to install any apps or configure any complicated VPN services. If there's no cell service, or a Cloudflare tunnel is not available, the standard practice of having two technicians with walkie-talkies still works great. One tech can stand somewhere with network access to the main ETH3, and the other can walk around to each VAV box.

1. Browse to the VAV Tool web app on the main OpenBAS-NWK-ETH3 on site.
    - If for any reason the VAV Tool is not available, use the System Design Studio to apply the necessary conditions, and ultimately Set measuredV to the current measured airflow in CFM to trigger the Kfactor update.
2. Under Load controller list, check the box to “Load from this ETH3”.
    - If not on the main ETH3, enter the IP address, username and password of the main ETH3.
3. Click “Load from ETH3”.
4. Minimize the “Load controller list” card by clicking anywhere on the header.
5. Within the NXVAV tool card, click on the “Search and select controller dropdown” and begin typing the name of the VAV controller to be balanced. Once you find the controller, select it.
6. After 5 seconds, the tool will read the compatible tool version from the script. If it matches, the rest of the interface will be enabled. If it doesn’t match, first confirm you’ve selected the correct controller. If the correct controller is selected, either the vav script or the OpenBAS-NWK-ETH3’s webpages need to be upgraded/downgraded to a matching version.
7. Override the airflow or damper position as needed.
    - When finished, Set VSptOverride back to Auto. If you forget, the override will be removed in ~30 minutes as seen beside “Auto’s in:”.
8. Measure and enter the current airflow reading and click Update Kfactor.
9. Heating, Fan, and pressure auto zero features are also available on the page for convenience.
10. The AHU control section can be used to ideally select and control the necessary AHU inputs and outputs, however this requires the AHU controller to use the following point mappings:
    - BO_1 for SupplyFanStartStop
    - AO_1 for fanSpeed
    - AI_1 for supplyTemp
    - AI_2 for staticPressure

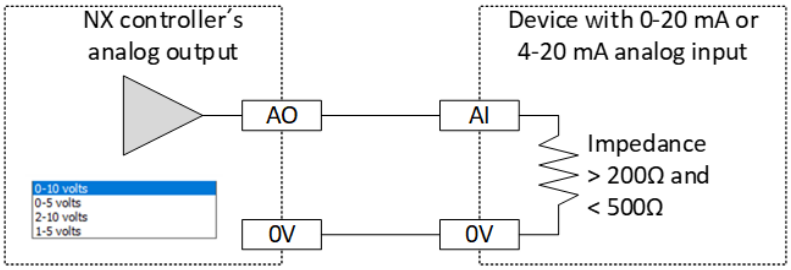
Configuring and calibrating outputs

OpenBAS controllers can have two different types of output points which are Binary outputs (BO_X) and Analog outputs (AO_X). Both types of output points can be configured in **Configuration->Inputs outputs** by selecting Analog output or Binary output from the main dropdown box**.**

- Analog output configuration
    - Analog outputs are controlled with a 0-100 value representing an output voltage between 0 and 10V respectively by default.
    - The local Analog outputs AO_1 and AO_2 (AO_3 and AO_4 as well for the OpenBAS-HV-NX4AO) can have their output voltage range changed to 0-5VDC, 2-10VDC or 1-5VDC using the drop-down menu within the table.
        - Remote or slave expansion Analog outputs must be configured by connecting to the controller in which they physically exist.

- An Invert checkbox can be selected to invert the logic (Ex. From 0-5VDC to 5-0VDC for a respective value of 0-100.

- Set each point's applicable unit by double clicking the appropriate cell under the Unit column. Select a unit category in the first dropdown box, followed by the appropriate unit in the second dropdown box.
  - Note that the selected unit is currently intended for reference only as the Analog outputs are still controlled using a value between 0 and 100, regardless of the unit selected.
  - The Analog output units will be included when the point is discovered over BACnet.
- When being used as a Slave Expansion device, the local Analog outputs can be configured with a default present value in case they go offline, or the master controller stops communicating. This default value, and the amount of time (without communication from the master) before these present values should be used can be configured under the "Local output state when offline as slave" header within **Configuration->Inputs outputs->General** option.
- The Analog outputs can sometimes be used to drive 0-20mA or 4-20mA inputs but only if:
  - isolation is not needed
  - the impedance is fixed (does not change during operation)
  - the inputs impedance is close to or between 250 Ohm to 500 Ohms.



  - Determine the input's impedance by referencing the devices datasheet or measure it directly with an Ohm meter.
  - Set the Analog output's voltage range (Hint: I = V/R):
    - 0-10V mode for 500 Ohm impedance resulting in 0-20mA.
    - 2-10V mode for 500 Ohm impedance resulting in 4-20mA.
    - 0-5V mode for 250 Ohm impedance resulting in 0-20mA.
    - 1-5V mode for 250 Ohm impedance resulting in 4-20mA.
    - For lower impedances, option 1 is to add a resistor in series to match either 250 Ohm or 500 Ohm options.
      - Ex. For 200 Ohm impedance, wire a 50 Ohm resistor in series with the input to get 250 Ohms, and select either the 0-5V or 1-5V voltage ranges for the Analog output.

- For lower impedances, option 2 is to select the closest voltage range and then the Analog output should be scaled accordingly in the PLC.
    - Ex. For 200 Ohm impedance, the 0-5V range would give 0-25mA which is too much current. The scaling factor can be calculated like: x = 20mA/25mA Therefore the analog output should be scaled by x = 0.8.

- Binary output configuration
    - Binary outputs 1 through 8 can be configured with a minimum on/off time or they can be averaged over time to prevent the relays from wearing out.
        - Selecting “Averaging” from the drop-down box within *Configuration->Inputs outputs->Binary output*, allows each local binary output (i.e., BO_1-8 for OpenBAS-HV-NX10 series) to be configured so that their present value is averaged over a multiple of 200ms between 0 (zero) and 255 (between 0 (zero) and 51 seconds). The binary output will not change until the averaged value is exactly equal to 1 or 0 (zero).
        - Selecting “Min ON/OFF” from the drop-down box within *Configuration->Inputs outputs->Binary output*, allows each local binary output (i.e., BO_1-8 for OpenBAS-HV-NX10 series) to be configured so that the state will be locked for a set amount of time after each state change (from 1 to 0 (zero) or vice versa). Like “Averaging”, this setting is configured using a multiple of 200ms between 0 (zero) and 255 (between 0 (zero) and 51 seconds). If some process tries to change or set the state of a binary output within the min on/off time period, the controller will update the Binary output’s state once the timer expires.
            - If for example a Binary output was turned off, and then before the min on/off timer expires, is set on and then back off again, the Binary output will remain off once the timer expires.
    - The “Ignore Overrides (priority 5) to BO_1-8” checkbox when selected will ignore any communication Overrides received for BO_1-8 (over USB, SPI, COM1-3). It will also release any existing Overrides (p5) on BO_1-8 once every minute.
    - When being used as a Slave Expansion device, the Binary outputs can be configured with a default present value or state in case the master controller stops communicating. This default value, and the amount of time (without communication from the master) before these values should be used can be configured in the “Local output state when offline as slave” section in **Configuration->Inputs outputs->General** (select General from the main drop-down box). If the respective Binary output’s check box is checked, the Binary Output will turn on when communication is lost.
    - “Minimum 200ms period for Binary outputs” within **Configuration->Inputs outputs->General**, is checked by default and ensures that no individual local relay (i.e., BO_1-8 on the OpenBAS-HV-NX10 series) will be toggled more than 5 times per second.
    - “Invert OpenBAS-LC-NX12R latching relay outputs” within **Configuration->Inputs outputs->General**, is checked by default to support the TC30T and WR61613K-84 (UL)

latching relays. Unchecking this box with either of these relays would cause the above latching relays to be closed (conducting) when the Binary outputs have a present value or state of 0 (zero) and vice versa.

- "Inter relay time" within **Configuration->Inputs outputs->General**, creates a delay between turning on each binary output when restoring their states after a power failure. This is intended for any controller aside from the OpenBAS-LC-NX12R since it uses latching relays which maintain their state during a power failure. "Inter relay time" is configured with a multiple of 50ms between 0 (zero) and 10 (between 0 (zero) and 0.5 seconds).

- See "Naming points" for information on creating, editing and deleting point names for outputs.

Output priorities

Binary outputs have four main priorities referred to from highest priority to lowest priority as Local Override, Communication Override, Logic Set and Schedule Set. These priorities map to BACnet priorities 1, 5, 9 and 11 respectively. Analog outputs only have the Communication override and Logic Set level overrides.

- Local overrides can be activated or released from a local display such as the OpenBAS-HV-LCD which is used in the OpenBAS-HV-NX10L or through any BACnet client*.
- Communication overrides can be activated or released (Auto) from various locations within the System Design Studio, or through any BACnet client*.
  - All communication overrides can be released for BO_1-BO_59 by Setting BO_60 (BACnet Binary value 160) to 0 (zero), 1, or releasing it **through BACnet/IP**. This will only release BACnet priority 5 when BACnet BO priorities are enabled.
- PLC Set level priorities are used internally by each controller's PLC and can also be activated for example by clicking the On or Off present value icon while viewing any BO in the System Design Studio. They can also be activated through any BACnet client*.
- Schedule Set level priorities are used when any schedule Sets a point's present value. They can also be activated through any BACnet client*.
- *The "Enable BACnet BO priorities" setting needs to be enabled from the Configuration->Communication tab under the Others heading. This will enable BACnet priorities for Binary outputs either when commanded through COM1 BACnet/MSTP or when commanded through an OpenBAS-NWK-ETH3 using BACnet/IP.

Naming points

Up to 199 point names can be configured, modified, uploaded and deleted from within the **Values and trends->Names** tab. The System Design Studio will only read the point names from the controller when changes are detected. Point names are shown throughout the System Design Studio when available and are included during BACnet discoveries. To add a point name, click the Insert button, enter a Name, select an Object type and Channel, and then click Save. **Point names are limited to 18 characters and**

can only contain alpha-numeric characters and underscores. Continue adding any required point names and then click the Upload button when finished.

- Any existing point names will be loaded after connecting to a controller.
- The “Refresh” button will ignore any existing changes in the Names table and load all existing point names from the controller.
- Point names can be modified by double-clicking on their current name under the “Name” column. Clicking outside of the text box or pressing enter will apply the change.
- When there are unsaved changes, the Names table will be highlighted.
- To save all changes to the controller, click “Upload”. This will replace all existing point names on the controller with the point names currently in the table. When the point names have been saved to the controller, the table’s highlighted background will return to default/white.
- The “Export” and “Import” buttons can be used to save multiple sets of 199 point names to your PC which can be helpful for debugging bigger projects.
    - After inserting your labels, click Export, enter a descriptive file name and browse to an appropriate location to save your labels.
    - Click “Import” and browse to a previously exported label file to replace all current labels currently in the Names table.
    - You can also export pinned points only – however note that importing these will still replace **all** point names with only the pinned ones. This is mainly intended to support MiPages for AI.
    - You do not need to export labels separately when using the “Backup to disk” feature from within the right-click context menu of a controller item in the side panel. The point names will be backed up when selecting the “Configuration” option of the “Backup to disk” feature.
- The “Delete” button will only delete point names from the table and the “Upload” button must be used to save any changes that have been made to the controller.
- The “Reindex” button will reorder the point names in order to make BACnet discoveries more efficient. This “reindexing” is done automatically when configuring from script or clicking upload.
- See “Debugging and testing logic” for more information on the “Pin/Unpin”, “Group” and “Update group” buttons.

Limiting point values

On NX controllers, up to 16 ranges of points (ADB, ADI, ADF, RES_FLT and RMT) may have their present value property limited to a minimum, maximum or both a minimum and maximum value. Point limits can be configured in the **Configuration->Point limits** tab. Note that point limits only apply to Set commands received by communication (i.e., through IP, USB, SPI, COM1, COM2, or COM3). Any Set commands originating internally, either from the PLC (logic) or Schedules will ignore all point limits.

- Select a limit type of either MIN, MAX or MIN+MAX along with a point type, End channel, Start Channel and, Minimum and Maximum values as applicable.
    - Start and End channels are inclusive.
    - Minimum and Maximum values are inclusive.
- Only base points are supported (RES_FLT_1-RES_FLT_40 and RMT_1-RMT_50).
- Limits only prevent a point from being Set with a present value outside of the configured Minimum and/or Maximum values.
    - Existing present values, regardless of whether they are within the applicable limits, will remain unchanged until a point is Set with a new value that is within the configured Minimum and/or Maximum values.
- Any point with conflicting limits will use the applicable limit with the lowest ID (closest to the top of the limit table).
- Point limits will also be ignored when using the LCD or Dual core display to adjust them.
- The Limit PLC instruction can also be used to limit specific points.

Configuring fieldbus and communication settings

Every OpenBAS controller has at least one communication port which supports the RS-485 standard by default. Communication port settings such as address, protocol, and baud rate can be configured in the **Configuration->Communication** tab. Since some OpenBAS controllers have two or three COM ports, it is important to note that individual addresses need to be configured for each communication port and not just one address per controller. In general, all standard practices for RS-485 buses (wire gauge/length, biasing, EOL) and the protocols in use should be followed but there are some OpenBAS specific recommendations:

- General
    - Assume a poll rate of about 25 points per second. Although the poll rate can vary widely depending on the protocol, baud rate, bus load, and many other factors, 25 points per second is a good “rule-of-thumb”.
    - It is good practice to limit the number of controllers on a fieldbus. Splitting large field buses into smaller field buses decreases risk since less controllers will go offline should any single controller or bus have an issue.
        - All OpenBAS controllers use RS485 drivers with no greater than ¼ Unit Load.
    - Use 22 AWG, three conductor communication rated cable (120 Ohm characteristic impedance, with one twisted pair).
        - The third conductor should be connected to the ref terminal of every OpenBAS controller. The ref terminals of OpenBAS controllers are internally connected through a 10 Ohm resistor to ground.
            - Shields should not be connected to any terminals except for a single connection to ground.
    - **Isolation and surge protection is absolutely necessary when running RS485 outside or between buildings.**

- Protocol support by COM port (master protocols in **bold**):
  - NX Series controllers
    - COM1
      - OPTO22 slave
      - N2 slave
      - MODBUS slave
      - BACnet MSTP server
      - **MODBUS master**
      - ARDUINO query (OpenBAS-HV-NX10D only as of firmware 3.17.0)
    - COM2 (when available)
      - Terminal ASCII (ECM on OpenBAS-HV-NX10D)
      - OPTO22 slave
      - N2 slave
      - **OPTO22/N2 master**
      - ARDUINO query (OpenBAS-HV-NX10D only as of firmware 3.17.0)
      - NSHF/NSGP (OpenBAS-HV-NX10D only)
      - IQ plus 355 (OpenBAS-HV-NX10D only)
    - COM3 (when available)
      - Terminal ASCII
      - OPTO22 slave
      - N2 slave
      - MODBUS slave
      - **OPTO22/N2 master*** (*System Design Studio cannot connect to OpenBAS slaves through COM3)
  - OpenBAS-NWK-ETH3
    - COM1
      - Terminal ASCII
      - OPTO22 slave
      - N2 slave
      - Modbus slave
      - **BACnet MSTP**
      - **OPTO22/N2 master**
      - **Modbus master**
      - ARDUINO query
      - Fire printer to email
      - FX-3500 printer gateway
      - FleX-Net printer gateway
    - COM2
      - Terminal ASCII
      - OPTO22 slave

- N2 slave
- Modbus slave
- **OPTO22/N2 master**
- **Modbus master**
- ARDUINO query
- Fire printer to email
- FX-3500 printer gateway
- FleX-Net printer gateway

  - Use one of the following baud rates depending on the protocol:

Protocol	Baud Rate
Arduino Query	2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200*
Optomux	2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200*
N2-Open	9600
Modbus	2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200*
BACnet/MSTP	9600, 19200, 38400, 57600, 76800

*115200 is only supported on the OpenBAS-NWK-ETH3

- Communication settings:
  - Depending on the controller, any available com ports (COM1, COM2 or COM3) and associated settings will appear with common fieldbus settings such as Address, Protocol, Baud rate, Stop bits, and Parity. All these settings must match (aside from Address, which must be unique, and the master controller must be configured with the appropriate master protocol) for every controller on a fieldbus.
    - Note that the address of a COM port that is configured with a Master type protocol is ignored (Optomux/N2 master or MODBUS master).
  - "Last point to poll" will cause the controller to reset its polling cycle after polling a specific remote point. This setting applies to both RMT and RES_FLT type remote points.
    - Ex. A last point to poll of 55 will disable RMT_56**+** and RES_FLT_56**+** (assuming expanded remote points are available).
    - Note: An expanded RMT or RES_FLT point enabled by the OpenBAS-ACC-32KNV memory expansion with an address of zero and a type of NULL will disable all higher remote point channels for RMT or RES_FLT remote points, respectively.
  - "Comm status mapping" allows X number of consecutive RES_BITS to be used to represent the online status of the first X number of unique slaves, that have at least one point configured as a remote point within the Master's **COM2** fieldbus.
    - "Start RES_BIT" will accept a number between 1 and 240.
    - "Amount" will accept a number between 1 and 24.
    - If the "Amount" + "Start RES_BIT" sum goes over 250, RES_BIT_250 will be the last RES_BIT used, and the rest of the slave controllers will not have their online status mapped.

- If the first 5 managed remote points in COM2 are configured with address 1 and the 6th managed remote point in COM2 is configured with address 2, then the online status of the slave device with address 2 will be mapped to the Start RES_BIT + 1. For example, if the Start RES_BIT is set to 40, the status of the slave device with address 2 is then mapped to RES_BIT_41.

- "High priority points" allows a range starting from RMT_1 up to RMT_10 to be given higher priority in the polling cycle. This causes the selected range to be polled at least once every 1-5 seconds depending on the "Period" selected.
  - "Period" is set as a multiple of 200ms and will accept a number between 5 and 25 for a total time of between 1 and 5 seconds.
- "Fieldbus statistics" has a "View" button that opens a modal dialogue with various statistics to help diagnose any issues on a fieldbus:
  - The first section titled "Packet statistics" shows two horizontal bar graphs that convey the percentage of packets that are Ok, Lost or have errors for COM1 and COM2 respectively.
  - The "Master scan time" section shows the previous, current, and maximum times that it has taken the controller to poll all configured and enabled remote points in either COM1 or COM2. Note that for NX controllers, this feature cannot be used if both COM1 and COM2 are configured with master type protocols.
  - "Eeprom write counter" shows the total amount of writes to the controllers internal eeprom. The NX controllers use Eeprom for the internal floats, internal integers and internal bytes (ADF, ADI, ADB), so it is important to minimize writes, ideally to less than once per 10 minutes. Note that while eeprom writes could come from a fieldbus, they can also originate from the PLC (do not attempt to create an Assign instruction copying the present value of an Analog input into an ADF for example), so this counter should always be reviewed even with single standalone controllers.
  - COM1 polling and COM2 polling sections both show the Last polled point and COM2 polling also has a Loop count where a loop is defined as every enabled remote point getting polled at least once.
  - SDS bridging shows the Average request time in ms, and busy response count of messages the OpenBAS-NWK-ETH3 forwards from SDS to downstream controllers.
  - The BACnet/MSTP section shown when connected to an OpenBAS-NWK-ETH3 effectively acts as a powerful MSTP debugging tool by providing statistics such as total active devices, highest active address, average token loop time, requests sent per token (bridged and polled remote points), and bandwidth percentage.
    - Under Device monitoring, there is a Generate button that will create a text file with even more detailed BACnet/MSTP statistics. Users should

first click the Clear statistics button to reset the statistics, and then after some time, click Generate to create and open the file.

- Bridging ("COM1/COM2 bridge") allows a range of addresses that exist on a controllers COM2 bus to be accessible by a Master controller connected to this controller's COM1 port. This is also required for IP connection 3/5 and USB connection 3. When configured correctly, this controller will forward any messages within the address range requested on COM1 onto COM2, wait for the response, and then forward the response back on COM1.
  - COM1 bus must be configured with Opto22 slave at minimum 19200 Baud.
  - COM2 bus must be configured with Opto22 Master at minimum 19200 Baud.
  - Due to the second "hop" this type of communication will necessarily be slower than "single hop" communications.
- Bridging for COM2/3 ("COM2/COM3 bridge", "COM2/COM3 alternate bridge") is not currently supported.
- Master of RMT51+ determines whether COM2 or COM3 will manage (poll for) RMT_51-RMT_255 with some exceptions. Please see the OpenBAS Remote Point Truth Table for more details.
  - Within *Configuration->Remote points*, the current managing com port will be listed beside each remote point.
  - In general, there are 3 sections of remote points that can be "moved around" between managing COM ports. They are RMT_1-50, RMT_51-255* and RES_FLT_41-255*.
    - *Expansion memory is needed for all remote points.
- "BBT address" or broadcast block transfer is an advanced feature that when set to 253, will allow the present value of RES_FLT_31-40 to be sent to all slaves in a single message, called a broadcast message. This feature is disabled at any value other than 253.
- "Remote point override rung" will send a communication override (priority 5) to any remote points used as the output of any "assign" instructions located in the first [X] instructions of PLC1, 2 or 3 (where [X] is the value of the "Remote point override rung" setting). Any "assign" instructions located **after** the X^{th} instruction where a remote point is used as the output will cause the controller to send a regular PLC Set command (priority 9). A value of 255 will disable this feature and all remote points used as the output of an "Assign" instruction will be sent communication overrides at priority 5.
  - OpenBAS controllers configured with the Opto22 slave protocol, only accept overrides to Analog and Binary outputs from a master controller so this feature is only intended to be used when interfacing with N2-Open controllers.
- "Discovery" allows you to select the range of each point type that will be discoverable over BACnet IP (requires OpenBAS-NWK-ETH3) and BACnet/MSTP.
  - Ranges must start from 1.
  - Ranges are automatically configured when configuring from script.

        - For point names to be included during a discovery, the point names must be reindexed. Point names will be reindexed automatically when configuring from script or go to *Values and trends->Names* and click the "Reindex" button to manually reindex them.
    - The "BACnet MSTP slave" or "BACnet MSTP" protocol is only available on COM1 of any "NX" controller or OpenBAS-NWK-ETH3 gateway and has some additional settings that appear when selected.
        - The BACnet MAC address can be configured in the always visible *Address* setting.
            - Always configure all slave-type controllers with consecutive addresses starting from 5.
            - The OpenBAS-NWK-ETH3 is typically set with a MAC address between 0 (zero) and 4.
        - "BACnet/MSTP_ID" is an identifier that must be unique among the entire BACnet network.
            - If BACnet/IP and BACnet/MSTP are both used in the same BACnet network (e.g. using a BACnet router), it is important to make sure every device has a unique ID across both IP and MSTP networks.
            - For "NX" devices, the default *BACnet/MSTP_ID* is technically 65535 but this specific value will force the ID to be the same as the *Address* setting (i.e., MAC). Similarly, in firmware 3.17.0 or newer, setting the ID to 4194303 will also force the ID to be the same as the Address setting.
                - Firmware versions < 3.17.0 have a useable range of 0-65534.
                - Firmware versions 3.17.0 or newer have a useable range of 0-65534, and 65536-4194302.
            - For the OpenBAS-NWK-ETH3, the default BACnet/MSTP_ID is technically 0 but this specific value will force the ID to be the BACnet/IP_ID + 1000. The useable range is therefore 1-65535.
        - "Max info frames" is available on the OpenBAS-NWK-ETH3. This controls the number of messages that the OpenBAS-NWK-ETH3 is allowed to send/request/forward each time it receives the token.
            - The default setting is 1.
            - Mircom JACE controllers' default to a Max info frames of 20, meaning that on a bus with 1 JACE and 1 OpenBAS-NWK-ETH3, both with default Max info frame settings, the JACE will send/request 20 messages for every 1 message that the OpenBAS-NWK-ETH3 can send/request.
                - For *Batch programming*, in this example, it would be recommended to temporarily Set the JACE Max info frames setting to 1 (or physically disconnect the JACE from the bus if acceptable), and Set the OpenBAS-NWK-ETH3 Max info frames setting to at least 50.

- If the OpenBAS-NWK-ETH3 is not being used to poll BACnet/MSTP points on COM1, Max info frames can be left at 1, once the technician is finished configuring all controllers.

- "Force recover time" is a time from 5-60 seconds (0-4 is disabled) that the OpenBAS-NWK-ETH3 will wait to receive a token before forcefully resetting and rejoining the bus. This is intended to improve Batch configure from script reliability on large networks where if the main OpenBAS-NWK-ETH3 somehow gets kicked out of the loop, it would otherwise take a long time to rejoin, which could cause the Configure from script process to fail for many controllers.
    - The OpenBAS-NWK-ETH3 should have the lowest address on the bus in order for this feature to work effectively.
    - Without this feature, the time for a device to rejoin can be *roughly* calculated as: Rejoin time = N*0.02s*50, where N is the number of devices on the network. So for 60 devices, the rejoin time is 60 seconds.
- "Max master" is a setting that should be configured on all devices within a BACnet/MSTP bus. It tells the controller which addresses are expected on the bus allowing significantly better communication efficiency and speed.
    - On each controller, set Max master to the number of controllers on the bus plus one. Setting the Max master to a number slightly above the total number of controllers on the bus will slow down communications slightly however it will save time and labour if more devices ever need to be added in the future since this setting should be configured on *every* device.
    - At minimum, set the max master properly on the **last two devices** so that if the last controller is offline, the second last controller will not waste time polling every possible address looking for the next controller to pass the token to, and the BACnet/MSTP bus will continue to function efficiently.
        - This is bad practice so if any issues arise, properly configuring Max master on all controllers should be the first step in debugging.
- "NPDU" can be set to local when connecting to an NAE/NCE for improved communications (legacy feature, "NX" controllers only).
- "Periodically broadcast I-Am" and "Start as MSTP master" are both experimental features that do not conform to the BACnet standard but have been included to improve integrations with some third party BACnet implementations (NX controllers only).
- BACnet priorities are enabled for any "NX" controller's Binary outputs when COM1 is configured for the *BACnet MSTP slave* protocol or the *Enable BACnet BO priorities* checkbox is selected. This will enable all 16 BACnet priorities for each Binary Output (through IP or MSTP) however the 16 priorities are all

mapped to the four OpenBAS priorities (BACnet priority level 1,5,9 and 11). Enabling BACnet BO priorities requires the controller to be restarted.

- "Time to offline (min)" is the time in minutes to wait without a BACnet/MSTP write to the Result float selected with "RF to watch", before Setting RES_BIT_247 (the "online as slave in COM1" reserved Result bit) to 0 (zero).
  - If "RF to watch" is 256, then a write to <u>any</u> Result float will be considered.

- "BACnet_IP/ID" and "BACnet ID mode" are available when connected to an OpenBAS-NWK-ETH3. By default, the *BACnet ID mode* is set to "Use last part of IP address" which means, for example, the *BACnet_IP/ID* will be 51 if the OpenBAS-NWK-ETH3's local IP address is 192.168.1.**51**. Setting the BACnet ID mode to *Manual* will make the BACnet/IP_ID whatever is entered into the textbox in the range 0-4194303 (default is 1000). This ID must be unique among the entire BACnet network.
  - If BACnet/IP and BACnet/MSTP are both used on the same network (e.g. through a BACnet router), it is important to make sure every device has a unique ID across both IP and MSTP networks.
- The "MODBUS master" and "MODBUS slave" protocols have additional settings that appear when selected. These settings can allow for some basic priority-based control if needed.
  - The Modbus master write coil function and write holding register options will either use the "write single" or "write multiple" Modbus function/message when writing to a coil or holding register respectively.
  - Response timeout (x10ms) is available on the OpenBAS-NWK-ETH3 when set as Modbus master and allows users to increase the default amount of time that the ETH3 will wait for a response. Some devices have been seen to take up to 1 second to respond.
  - Poll delay (ms) is available on the OpenBAS-NWK-ETH3 when set as Modbus master and allows users to increase the default "silence between frames" for particularly problematic or slow devices.
  - "Modbus slave write single coil action" will treat received "write single coil" functions/messages as either regular Set or as Override messages.
  - "Modbus slave write multiple coils action" will treat "write multiple coils" functions/messages as either regular Set or as Override messages.
  - Packets to capture (x10) is available on the OpenBAS-NWK-ETH3 and will record up to 2500 Modbus/RTU packets from either com1 or com2 and save them to the connected USB memory stick. This can be very helpful when requesting support from various Modbus device manufacturers.
- Fire gateway point mapping settings will appear when either FX-3500 printer gateway or FleX-Net printer gateway protocols are selected on COM1 or COM2 of the OpenBAS-NWK-ETH3. This requires Mircom's DT-9000 RS485 to RS232 converter and DT-9006A2 Null Modem cable.

- Object type determines whether to map 60 fire points into RES_FLT's, or COM1 or COM2 Remote points.
- When FX-3500 printer gateway is selected, RES_BIT_100-127 will also be used.
- Start channel refers to the first of 60 consecutive internal points (RES_FLT/RMT) that will be used for storing the data received from the Fire Panel.
- Note that only one COM port can be used as a fire printer port gateway at a time. Use multiple OpenBAS-NWK-ETH3's if more than one FACP (Fire Alarm Control Panel) needs to be monitored.
- The point mappings starting from "Start channel":
    1. Total event count – Total number of events sent from FACP since the last OpenBAS-NWK-ETH3 power on.
    2. Loop event count – Total number of events from loop devices since the last OpenBAS-NWK-ETH3 power on.
    3. System event count – Total number of system events since the last OpenBAS-NWK-ETH3 power on.
    4. Exception event count – Total number of events containing an error or that could not be fully parsed.
    5. Non data event count – Total number of events containing no recognized data.
    6. Active event count – Total number of events stored in points 11-60 (5 points per event, referenced from "Object type" and "Start channel")
    7. Free for future use
    8. Free for future use
    9. Free for future use
    10. Free for future use
    11. Most recent event: Time stamp (HH:MM) in FX-3500 or Node number (1-63) in FleX-Net
    12. Most recent event: Loop (1-3) in FX-3500 or CPU (0-7) in FleX-Net
    13. Most recent event: Address (1-159,201-359) in FX-3500 or Zone (1-400) in FleX-Net
    14. Most recent event: Type (Sensor = 1, Module = 2, External = 3)
    15. Most recent event: Event
        - Loop events
            - Alarm activated = 1
            - Alarm restored = 2
            - Trouble = 3
            - Trouble restored = 4
            - Missing device = 5
            - Open circuit trouble = 6
            - Other/Unknown = 255
        - System events

- (101-199 reserved for FX-3500)
- (201-254 reserved for FleX-Net)

16. (16 through 20 would be the second most recent event, and so on…)

- The RES-BIT_100-127 point mappings for FX-3500 printer protocol are as follows:
    1. Alarm
    2. Trouble
    3. Supervisory
    4. System reset in progress
        - Clears RES-BIT_100-127 when true.
    5. Zone 1 Alert
    6. Zone 1 Pre-release
    7. Zone 1 Imminent release
    8. Zone 1 Releasing agent
    9. (Continues for Zone 2… Zone 6)

- The NSHF/NSGP and IQ plus 355 protocols are only available on COM2 of an OpenBAS-HV-NX10D controller and supports integration with these specific weight scales.
- The Environmental Condition Monitor protocol is only available on COM2 of an OpenBAS-HV-NX10D controller and supports integration with this specific monitoring system.
- The MPBUS port is available when connected to an OpenBAS-HV-NXVAV Series controller. To simplify integration, hard-coded mappings have been implemented for up to two actuators, and two sensors. MP remote points can also be configured for individual “Data pool” values; see Configuring remote points.
    - Checking Enable will cause the hardcoded mappings to be used for that device.
    - Expected address should be configured to the actual address of the MP device.
        - PP stands for Point-to-Point, is sometimes referred to as “unaddressed”, and implies the use of broadcasts.
        - Address PP can only be used with a single MP device (excluding NXVAV). If using 2 or more MP devices, they must use addresses MP1-MP15.
    - Invert will swap the default open and closed positions of the actuator.
    - Discover will try to read the serial number of all MP addresses and populate the Set address dropdown with the results.
    - Get serial will try to read the serial number of a specific MP address.
        - Get serial on PP will return the lowest numbered serial number on the bus.
        - PPX stands for Point-to-Point eXclusive and Get serial will return the lowest numbered serial of any PP/unaddressed devices on the bus. Devices with addresses MP1-MP16 are ignored in this case.
    - Set address will try to set the address of the MP device with the selected serial number to the selected address.

- Setting a devices address to PP is often referred to as “unaddressing” the device.

- The hardcoded mappings are as follows:
  - General
    - BI_8 – Belimo PC Tool enabled status
  - Actuator 1
    - BI_7 – Online status
    - BO_6 – Disengaged status
    - BO_7 – Motor stopped status
    - BO_8 – Calibration active status
    - AI_7 – Current position percentage
    - AI_8 – Current setpoint percentage (read only)
    - AO_9 – Commanded position percentage (write only)
  - Actuator 2
    - BI_33 – Online status
    - BI_34 – Disengaged status
    - BI_35 – Motor stopped status
    - BI_36 – Calibration active status
    - AI_33 – Current position percentage
    - AI_34 – Current setpoint percentage (read only)
    - AO_10 – Commanded position percentage (write only)
  - Sensor 1
    - BI_25 – Online status
    - AI_25 – Temperature (0-50C)
    - AI_26 – Relative humidity percentage
    - AI_27 – CO2 (0-2000ppm)
    - AI_28 – Temp. Setpoint (10-50C)
  - Sensor 2
    - BI_29 – Online status
    - AI_29 – Temperature (0-50C)
    - AI_30 – Relative humidity percentage
    - AI_31 – CO2 (0-2000ppm)
    - AI_32 – Temp. Setpoint (10-50C)

Configuring OpenBAS-NWK-ETH3 communication settings through USB flash memory

While all settings on the OpenBAS-NWK-ETH3 can be configured through the System Design Studio, some settings can also be configured with just a USB flash memory stick preloaded with a text file. Two of the files that are created when inserting a USB flash memory into a powered on OpenBAS-NWK-ETH3 are COMM1CFG.ini and COMM2CFG.ini. These files allow modification of protocol and baud rate for both communication ports as well as the fire panel gateway protocol settings *Object type* and *Start*

channel. The files can be accessed by physically connecting the USB flash memory to any PC. Once the required changes have been made in the text file, change the "[N]" in the second line to "[Y]", save and reinsert the USB flash memory into the OpenBAS-NWK-ETH3.

- Do not connect a USB flash memory at the same time as a USB Type B cable (used for firmware upgrades). Remove one before inserting the other.
- Leave the USB flash memory connected for at least 15 seconds for the changes to take effect.
- Note that the controller will automatically change the [Y] back to [N] in the second line of the respective text file.

Configuring email alerts

The OpenBAS-NWK-ETH3 can send email alerts however it doesn't support TLS based email servers so a proxy like STunnel may be needed. Up to ten separate emails can be preconfigured on the OpenBAS-NWK-ETH3 which can be sent on a trigger. The email server settings, and individual emails can be configured from the web pages by going to the **Configure->email server configuration** page.

- The server configuration page allows the SMTP server domain, port, username, password and a from-email to be configured.
    - When using STunnel, use port 25, and leave the username and password blank, the username and password will be configured within the STunnel settings.
- Stunnel example configuration for Gmail
    - Install STunnel on an appropriate device.
    - Review the Stunnel documentation for configuration details. In general, set:
        - Operating Mode to Client
        - Listen IP to 0.0.0.0
        - Listen port to 25
        - Connect IP to smtp.gmail.com:587
        - TLS Cipher to None
        - Application Protocol to smtp, auth=Plain
        - Username to [full email]
        - Password to [application password from gmail can be generated after enabling 2 factor authentication]
        - No certificates are needed.
- To configure an individual email alert on the OpenBAS-NWK-ETH3, click on the respective "e-mail #x" button from the **Configure->email server configuration** page.
    - Enter a point to be used as the trigger. Any single point from the following range can be used:
        - ADB-1..100, ADI-1..100, ADF-1..100, RB-1..255, RF-1..255, RMT-1..2000, FIRE-1..2
    - The To:, CC:, and BCC: fields can have multiple emails separated by semicolons with a max character limit of 49 characters total in each field.
    - The Subject field has a character limit of 49 characters.

- The Message Body field has a character limit of 189 characters.

Debugging and testing logic

The **Values and trends->Names** tab doubles as a powerful debugging tool by allowing any set of points to be grouped together in order to monitor, Set and Override their present values as needed.

- Use the instant search box to find points related to the logic you are testing.
  - When filtering by object type, be sure to use the short forms as you see them. Ex.: Type "BO" to find binary output points, not "binary" or "digital".
- Use the Pin/Unpin button to hold points of interest at the top of the table.
  - Note that the pinning feature is decoupled from the group feature described below.
  - If no changes have been made to the current selection of points, currently pinned points will be unpinned.
  - If changes have been made to the current selection of points, currently pinned points that have been deselected will be unpinned and any new selections will be added or pinned to the top, along with the existing pinned points.
- Group X and Update group
  - Note that the grouping feature is decoupled from the pinning feature described above.
  - Note that point names in Groups are based on (remembered by) the Object type and Channel number, not the Name text.
  - Groups will persist based on the controller's Serial number, even after closing the System Design Studio.
  - By default, the Names table will load with Group 1 points selected (checked, but not pinned).
  - Changing the current Group will update the currently selected point names. Be sure to select the group you want to work on **before** beginning to select or make changes to your selection.
  - Use the Update group button to save the currently selected point names into the currently selected group (as identified by the "Group [x]" drop down to the left of the Update group button).
- Use the buttons in the Override column to command input or output points at priority 5.

Adding expandable memory accessories

When adding or removing accessories such as the OpenBAS-ACC-32KNV or converting an NX10 series controller into an OpenBAS-HV-NX10D by adding the OpenBAS-HV-CORE2 "dual core display", it is highly recommended to perform a factory reset after following the proper connection procedure.

Note: Currently the OpenBAS-NWK-ETH3 does not support any of the I2C accessories

1. Backup the controller to disk using the *Backup to disk* option.
2. Disconnect all power to the controller being upgraded.
3. Physically connect the accessory to the controller.

4. Power up, connect to the controller in the System Design Studio, and perform a full factory reset.
5. Restore the previously created backup to the controller using the *Restore from disk* option.

LCD display and alarm configuration

The LCD display (OpenBAS-HV-LCD) comes with the OpenBAS-HV-NX10L by default and can be purchased as an add-on accessory for any NX-based controller (however it will only mount directly onto the OpenBAS-HV-NX10P controller). Up to 8 Alarms, or LCD display messages can be configured in **Configuration->LCD display**.

1. Select and enter a RES_BIT Channel number that will be used by the controller to determine whether the associated alarm message should be shown on the LCD display. For example, selecting RES_BIT_40 and entering "My first alarm" as the Alarm message will cause the LCD display to show the Alarm Message "My first alarm" while RES_BIT_40's state is True (1).
2. If there are multiple *active* alarms (i.e., RES_BIT_X == 1), the Alarm message of the Alarm with the lowest (closest to the top) ID will be shown on the LCD display.
3. Alarm messages are limited to 16 characters.
4. "Connection port" is currently limited to the default I^2C, or None. Selecting None will disable the communication between the controller and the LCD display and make it unresponsive.
5. "Time to standby" accepts a value in seconds between 15 and 254 (0 (zero) will never go to the standby screen) that determines how long the LCD display will show the current screen before returning to the standby screen. The standby screen simply reads "LCD display in standby…" along with the USB connection status ("USB" will blink when connected).
6. "Extra byte to read" is a legacy feature that is no longer supported.
7. "Default screen" allows the default screen to be selected which is used on power up, as well as when clicking the red back button from one of the main menu screens.
    - Selecting Custom will make two additional settings appear, "Header" and "Result floats".
        - "Header" allows a point name index to be selected (1..200) that will be used for the text of the top line of the LCD display.
            1. Index 200 can be used for the controller's name.
        - "Result floats" allows four RES_FLT channels to be selected that will have their present values shown on the bottom line of the LCD display.
8. LCD screen overview: (use up and down arrows to switch screens in the order below)
    - Start
        - Displays I^2C status, firmware version (sub version will blink), date, and time.
    - Alarms
        - Displays Alarm message of the active alarm with the lowest ID.
    - Operate
        - Allows operator to Set (priority 11) or local override (Priority 1) the binary outputs (BO_1-BO_40).

1. Hold the up and down arrow buttons simultaneously to access the Operate screen options.
2. Use the up or down arrow buttons to select between “Operate all” and “Relay Operate”, and then press the enter button.
3. If selecting “Relay Operate” use the up or down arrow buttons to select a Binary output and then press the enter button.
4. Use the up or down arrow buttons to select between “Off”, “On”, “OL_ON” (local override On), “OL_OFF” (local override off) or “auto” (release local override).

- View
  - Displays the present values for TMR, RES_BIT and RES_FLT type points.
    1. Press the enter button to access the View screen options.
    2. Use the up or down arrow buttons to select between “timer”, “result bits” and “res. values” (RES_FLT), and then press the enter button.
    3. Use the up or down arrow buttons to select the appropriate point channel.
- Hardware
  - Displays all IO point present values and priorities.
    1. Press the enter button to access the Hardware screen options.
    2. Use the up or down arrow buttons to select between “See binary I/O”, analog outputs, “Analog input” and “BO priority lev.” then press enter.
    3. Use the up or down arrow buttons to browse through sets of point channels with their present values.
- Communication
  - Displays various communication related information along with RMT_1-RMT_50 present values.
    1. Press the enter button to access Communication screen options.
    2. Use the up or down arrow buttons to switch between USB connection status, USB communication statistics, SPI communication statistics, COM1 communication statistics, COM2 communication statistics, or “Remote pt. COM2” (RMT_1-RMT_50 present value).
- Adjust
  - Allows an operator to Set ADB, ADI, and ADF, calibrate AI or modify schedules.
    5. Hold the up and down arrow buttons simultaneously to access the Adjust screen options.
    6. Use the up or down arrow buttons to switch between “Adjust ADB”, “Adjusts ADI”, “Adjust ADF”, or “calibrate AI”, then press enter. Note “Schedule adjust” is no longer supported.
    7. For Adjust ADB, ADI or ADF, use the up or down arrow buttons to select a point channel, press enter and then continue to use the up or down

arrow buttons to modify the present value (can only increment ADF's +/- 1). Press the enter button when finished.

8. For Calibrate AI, use the up or down arrow buttons to select a point channel, press enter, and then continue to use the up or down arrows to increment or decrement the calibration value +/- 1. Press enter when finished. Note that the calibration type cannot be changed from the LCD display.

- Custom
  - Displays user configurable text on the top line along with the present value of 4 RES_FLT points in the bottom line. See the LCD display "default screen" option explanation above.

Dual core display settings

When connected to an OpenBAS-HV-NX10D, the **Configuration->Dual core display** tab becomes available.

- "USB memory storage settings" allow for remote points, Binary inputs, Binary outputs and up to 16 other points configured in *Configuration->Trends* to be sampled and stored on a USB flash memory device once per minute.
  - When connecting a USB flash memory device to the USB-A port on the OpenBAS-HV-NX10D display:
    - The yellow USB indicator LED will illuminate momentarily.
    - A folder will be created in the root folder named "/_NX5_/YR_[XXXX]/MES_[X]/" (with the current year, and index of the current month respectively).
    - The /_NX5_/USB_LOG.CSV file will be appended (and created if needed) with the current time and date.
  - When Trends is checked, the GRAF-1.CSV through GRAF-16.CSV files will store the present value, time and date every minute for the points configured in *Configuration->Trends*.
  - When 'Remote points RMT 51…255' is checked, GRAF-17.CSV will store the present value, time and date every minute for them.
  - When 'Remote points RES_FLT 41…255' is checked, GRAF-18.CSV will store the present value, time and date every minute for them.
  - When 'Remote points RMT 1…50' is checked, GRAF-19.CSV will store the present value, time and date every minute for them.
  - When 'Binary input and output status' is checked, GRAF-20.CSV will store the state, time and date every minute for BI_1-40 (EB) and BO_1-40 (SB).
  - "Include headers in CSV files" will store the point type and channel number beside each sample point when checked in the files GRAF-17.CSV through GRAF-20.CSV.
- "Display language" will change the language used on the OpenBAS-HV-NX10D's display only (OpenBAS-HV-CORE2).

- Common display tasks:
    - Checking the IP address (of an SPI connected ETH3), controller time and date.
        - From the Start screen press the Enter button to view the IP address. Use the Right and Left arrow buttons to switch between IP address, time and date.
    - To view the present value of any Binary inputs, Binary outputs, Lighting groups, Analog inputs or Analog outputs, switch to the main Hardware screen and press the Enter button. Use the right and left arrow buttons to switch between point types.
        - For the Analog input and output sub-screens, pressing the Enter button will enable the up and down arrow buttons to traverse through all the point channels. Press the red button to go back.

Viewing, Setting and understanding the internal point database

The present value and name of each point in the controllers point database can be viewed and set from within the **Values and trends->Points** tab. Select an object type from the main dropdown list to view those points in a table.

- Universal inputs (AI, BI, UI)
    - View the state or present value of all 40 analog or binary inputs in the point database.
    - Each universal input can be used as binary or analog*
        - *The one exception is the OpenBAS-HV-NXSF has 8 dedicated binary inputs, and 8 dedicated analog inputs, and 0 (zero) universal inputs. Note that the PLC or Logic tab can be used to convert analog inputs to binary inputs using result bits.
    - Binary inputs 1-40 convert an input voltage of between 0 (zero) and 4V into a 0 (zero) value/state and convert an input voltage between 8 and 12V into a 1 value/state*.
        - *Unless configured with “Active-low” logic, see Configuring and calibrating inputs.
    - Analog inputs can be configured for many different signal types and so they can take various voltage (or current) ranges. See the controller’s installation manual for more information and wiring diagrams.
    - In order to take advantage of all 40 universal inputs, 4x OpenBAS-HV-NX10 series controllers must be configured as Slave type remote points at addresses 100-103 within a 5th NX10 series ‘master’ controller (See ‘Configuring remote points’ for more information).
    - As of SDS 1.3.1, Analog and Binary inputs can be Overridden by the user. Previously users had to modify the input calibration to achieve the required input value for testing.
        - Note that Overriding an Analog input will also cause the Binary input of the same channel number to stop updating (and vice-versa). Users can override specific values to AI_X and BI_X simultaneously in the Points->Universal inputs table.
- Analog outputs (AO)
    - View/set the present value and priority of all 10 analog outputs in the point database.

- Analog outputs 1-10 can take an integer percentage between 0 and 100. They can output a voltage between 0 and 10 volts.
- Analog outputs can bet Set from the PLC or Overridden by communication. The PLC will no longer be able to set the present value of AO's when they are overridden. To release an override to an analog output, click the Auto button from within the analog outputs table or the Names table.
  - By default, the present value and override will be cleared during a power cycle or software reset. This feature can be disabled in *Configuration->Power up init*.
- In order to take advantage of all 10 analog outputs, 4x OpenBAS-HV-NX10 series controllers must be configured as Slave type remote points at addresses 100-103 within a 5th NX10 series 'master' controller (See 'Configuring remote points' for more information).

- Binary outputs (BO)
  - View/set the state or present value and priority of all 60 binary outputs in the point database.
  - Binary outputs 1-40 are used to control physical relays and binary outputs 41-60 are reserved for priority-based control of lighting groups.
  - Binary outputs have 4 inherent priorities however all BACnet priorities* are supported:
    - Local write/Set (equivalent to BACnet priority 11)
    - PLC/Logic (equivalent to BACnet priority 9)
    - Communication override (equivalent to BACnet priority 5)
    - Local override (Requires display, equivalent to BACnet priority 1)
  - In order to take advantage of BO_1 through BO-40, 4x OpenBAS-HV-NX10 series controllers must be configured as Slave type remote points at addresses 100-103 within a 5th NX10 series 'master' controller (See *Configuring remote points* for more information).
- Internal floats (ADF)
  - View/set the present value and point names of all 100 ADF type points in the point database.
  - ADF points are typically used to store setpoints and can store any valid IEEE-754 32 bit float value.
  - The value of each ADF point will remain even if power is disconnected or lost.
  - When configuring from EZ-Script, up to 20 ADF points will be used for the stack starting from ADF_100 down to at most, ADF_81.
  - Warning: ADF points are stored in the eeprom of "NX" controllers which can only be written to about 1,000,000 times. Setting any eeprom based point 1,000,000 times can permanently damage the memory of the controller. The controller will ignore any attempts to Set or Override the present value when the current value is the same in order to limit the number of eeprom writes.
- Internal integers (ADI)

  - View/set the present value and point names of all 100 ADI type points in the point database.
  - ADI points are typically used to store setpoints and can take any positive integer value in the range 0 – 65535.
  - The value of each ADI point will remain even if power is disconnected or lost.
  - Warning: ADI points are stored in the eeprom of "NX" controllers which can only be written to about 1,000,000 times. Setting any eeprom based point 1,000,000 times can permanently damage the memory of the controller. The controller will ignore any attempts to Set or Override the present value when the current value is the same in order to limit the number of eeprom writes.
- Internal bytes (ADB)
  - View/set the present value and point names of all 100 ADB type points in the point database.
  - ADB points are typically used to store setpoints and can take any positive integer value in the range 0 – 255.
  - The value of each ADB point will remain even if power is disconnected or lost.
  - Warning: ADB points are stored in the eeprom of "NX" controllers which can only be written to about 1,000,000 times. Setting any eeprom based point 1,000,000 times can permanently damage the memory of the controller. The controller will ignore any attempts to Set or Override the present value when the current value is the same in order to limit the number of eeprom writes.
- Result bits (RES_BIT)
  - View/set the present value and point names of all 255 result bit type points in the point database.
  - Result bits are typically used to store the result of a Boolean type of PLC instruction and can take a value of 0 (zero) or 1.
  - Some result bits have predetermined uses and are generally not available for use:
    - RES_BIT_1-20 are control bits for Lighting groups 1 through 20 (NX controllers only, see the Lighting group PLC instruction overview (LightG) for more information).
    - RES_BIT_21-36 are control bits for Timers 1 through 16 (see Timer PLC instruction for more information).
    - RES_BIT_37-39 are fixed oscillating timers at 1, 2 and 4 seconds, respectively. They have a period (T) of 2, 4 and 8 seconds respectively (50% duty).
    - **RES_BIT_40-245 are available for use.**
      - When configuring from EZ-Script, a number of Result bits will be used for the stack (counting downwards from RES_BIT_240) depending on the COMPILER_SMALL_STACK compiler flag.
        - When using the COMPILER_SMALL_STACK compiler flag, at most RES_BIT_221-240 will be used.

        - When not using the COMPILER_SMALL_STACK compiler flag, at most RES_BIT_61-240 will be used.
    - RES_BIT_246 is the default Debug bit.
    - RES_BIT_247 is a status bit representing whether the controller is online as a slave in COM1.
    - RES_BIT_248 is a status bit representing whether the controller is online as a slave in COM2.
    - RES_BIT_249 is a fixed oscillating one-minute timer with a period (T) of 2 minutes (50% duty).
    - RES_BIT_250 forces a local override when a button labelled PB1 is depressed (legacy controllers only).
    - RES_BIT_251-253 are power on timers and their value will be True (1) for 1, 5 and 10 seconds respectively immediately after powering up a controller.
    - RES_BIT_254 is linked to the green status LED of any OpenBAS controller display.
        - Note that both the Green and Red LEDs cannot be illuminated at the same time. Red has priority.
    - RES_BIT_255 is linked to the red status LED of any OpenBAS controller display.
        - Note that both the Green and Red LEDs cannot be illuminated at the same time. Red has priority.
  - Unlike the eeprom-based internal bytes, integers and floats (ADB, ADI, ADF), Result bit points can be written to as frequently as needed (or as frequently as the PLC loops).
  - The present value of each Result bit point will be lost if power is lost to the controller unless a battery backup has been installed. Only the OpenBAS-HV-NX10 series and OpenBAS-LC-NX12R lighting controllers come with the battery backup by default however every NX-based controller has the required battery connector if needed.
    - The present value of each Result bit point will be cleared by default every time the controller is powered up. This setting can be changed in the *Configuration->Power up init* tab.
- Result floats (RES_FLT)
  - Result floats are typically used to store the result of a PLC instruction however the expanded Result floats can be used as extra Remote points on NX controllers when available.
  - Result floats can store any valid IEEE-754 32 bit float value.
  - View/set the present value and point names of all result float type points in the point database.
    - NX controllers
      - Expanded RES_FLT_**41**-255 are available on the OpenBAS-HV-NX10D.

- Expanded RES_FLT_**51**-255 are available on any NX-based controller when paired with the OpenBAS-ACC-32KNV or similar memory expansion.
- OpenBAS-NWK-ETH3
  - Expanded RES_FLT_41-255 are available on the OpenBAS-NWK-ETH3.

- When configuring from EZ-Script, a number of RES_FLT points will be used for the stack starting from RES_FLT_40 down to at most, RES_FLT_31.
- Unlike the eeprom-based internal bytes, integers and floats (ADB, ADI, ADF), Result float points can be written to as frequently as needed (or as frequently as the PLC loops).
  - Careful consideration should be given when writing to RES_FLT_41+ (such as from the PLC), and when used as remote points, to make sure that remote points that are configured to Set eeprom-based points such as ADB, ADI or ADFs are not Set too frequently as this can permanently damage the controller's memory. The controller will ignore any attempts to Set or Override the present value when the current value is the same in order to limit the number of eeprom writes.
- The present value of each result float type point will be lost if power is lost to the controller unless a battery backup has been installed. Only the OpenBAS-HV-NX10 series and OpenBAS-LC-NX12R lighting controllers come with the battery backup by default however every NX-based controller has the required connector.
  - The present value of each RES_FLT point will be cleared by default every time the controller is powered up. This setting can be changed in the *Configuration->Power up init* tab.

- Timer (TMR)
  - View the present value, control bit state, reload point*, reload present value*, and mode of every timer in the point database.
    - *reload point and reload present value are for decrementing timer modes only.
  - There are 16 Timer points in each controller.
  - An undefined mode indicates that the timer has not been configured in the PLC.
  - As of SDS-1.3.0 TMR points are no longer read only and can be Set by the user.
  - Control bit type and channel (RES_BIT_21-36) cannot be changed.
  - See the Timer instruction overview for more information.
- Remote points (RMT)
  - The present value of all remote points, including NX expanded remote points (RMT_51-255, RES_FLT_41-255), can be viewed in Configuration->Remote points.
    - RMT_51-255 are available on the OpenBAS-HV-NX10D or any NX-based controller paired with the OpenBAS-ACC-32KNV or similar memory expansion.
    - COM1-RMT_1-1000 and COM2-RMT_1-1000 are available on the OpenBAS-NWK-ETH3.

- Remote points need to be configured in order for their present value to be Set with (copied from) the current value of a point existing on another controller through a fieldbus (see Configuring remote points).
- The present value of RMT_1-50 (NX controllers) will be lost if the controller loses power unless the backup battery has been installed.
- The present value of RMT_51+ will be reset to 0 (zero) during a power cycle or software reset when using the OpenBAS-HV_NX10D, and the values will remain unchanged during a power cycle or software reset when using the OpenBAS-ACC-32KNV memory expansion.
- Careful consideration should be given when writing to remote points (such as from the PLC) to make sure that remote points that are configured to Set eeprom-based points such as ADB, ADI or ADFs are not Set too frequently as this can permanently damage the controller's memory. The controller will ignore any attempts to Set or Override the present value when the current value is the same in order to limit the number of eeprom writes.
- See Configuring remote points for more information on the various types of remote points.

Viewing trends

Trends can be viewed in the **Values and trends->Trends** tab.

- Currently available trends will show up in the main dropdown box.
- Trends must first be configured in Configuration->Trends (See Configuring trends)
- The Refresh button will reload all available trends currently available on the controller.
- The Redraw button will reload all sample points of the currently selected trend from the controller.
- The Export button will export the currently selected trend's sample points into a CSV (comma separated values) file.

Configuring trends

Up to 16 trends can be configured in the **Configuration->Trends** tab. Trends can be configured to store the present value, time and date of any point at a predefined interval. Since storage is limited, these trends are often used as a backup measure since there are more trending options, features and storage available when using the System Management Studio (Powered by Niagara).

- The number of sample points that can be saved depends on the storage mode and point type.

Trend type	Number of sample points
Binary + time + date	248
Value	248
Value + time	165
Value + time + date	124

- Binary type trends will be checked once per minute for a change in value. When the value has changed, the time and date will be saved along with the present value.
- Non-binary type trends can have their sampling interval set at **1*,** 5, 10, 15, 20, 30, or 60 minutes.
    - ***** One-minute intervals are temporary and will automatically switch to a 5-minute interval once the buffer is filled.
- When a trend's storage is full, the oldest sample point will be deleted to make room for the newest sample point. This is commonly referred to as a FIFO (First in, First out) or circular buffer.
- The Clear data button can be used to erase all sample points from a trend's storage. This is commonly used when changing the point being sampled within a specific Trend.

Configuring USB trends

On the Open-BAS-NWK-ETH3, up to 10 USB trends can be configured in the Configuration->USB trends tab. USB trends will always store the present value, time and date for the selected point at the selected interval.

- Each USB trend is saved in the format TRYYMMTT.CSV where YY is the year, MM is the month, and TT is the trend number (01 through 10).
- Click the Manage trends button to download or delete any USB trend file.

Configuring remote points

Remote points can be configured from within the **Configuration->Remote points** tab. Depending on the controller and memory expansions there are two Object types that can be configured as remote points, RMT_X and RES_FLT_X:

- NX Controllers
    - RMT_1-50
        - Available on all NX controllers.
    - RMT_51-255
        - Available on "expanded memory" controllers, see *Memory overview.*
    - RES_FLT_41-255
        - Available on "expanded memory" controllers, see *Memory overview.*
        - RES_FLT_1-40 are PLC result registers only and cannot be configured as remote points.
- OpenBAS-NWK-ETH3
    - COM1-RMT_1-1000
    - COM2-RMT_1-1000

Both RMT and RES_FLT type remote points work the same way. After clicking the Configuration->Remote points tab, the System Design Studio will begin reading remote point configurations from the controller. Remote points exist on a master controller, but their present value is periodically updated with the present value of a point existing on a slave controller that is connected through a fieldbus. Be

sure to configure the fieldbus and communication settings of all controllers connected to the fieldbus for the remote points to show an online status. In addition, it is important to consider the remote point's managing COM port as well as the Last point to poll setting(s). For Remote points to work, the controller in which they are configured on must have the appropriate fieldbus configured with the OPTO22/N2 master or Modbus master protocols. As of SDS 2.0.0 with firmware 3.17.0, the OpenBAS-NWK-ETH3 also supports Remote points over BACnet/MSTP, but **any points coming from an OpenBAS slave controller must be set as discoverable**. Points on OpenBAS slave controllers are set as discoverable automatically during the Configure from script process, or they can be set as discoverable manually from within the Configuration->Communication tab under the *Others* header.

- Use the main drop-down box to:
    - Switch between viewing RMT and RES_FLT type remote points when available on "NX" controllers.
    - Switch between viewing COM1-RMT_1-1000 and COM2-RMT_1-1000 on the OpenBAS-NWK-ETH3.
- Enter and select the slave device's Address, Object type and Channel to configure a remote point.
    - For MODBUS type remote points, The Object type is selected from a drop down, so do not include the Object type number when entering the Channel. Ex. For the MODBUS entity number 400001 (Holding register), select Holding Register from the Object type dropdown and enter 1 for the Channel.
        - MODBUS Remote point Channels can be configured to any integer from 1 to 65535 (inclusive).
        - MODBUS Remote points configured as Input registers or Holding registers can read or Set a Present value to any integer from 0-65535 (inclusive).
    - The OpenBAS-NWK-ETH3 supports various multi-byte encodings or float formats when polling Holding Registers or Input Registers over MODBUS/RTU.
        - When configuring a MODBUS/RTU based Remote point, and selecting Holding Register or Input Register as the Object type, two dropdowns will appear directly beneath the Object type dropdown.
            - The first dropdown allows for various data types to be selected:
                - UINT16 (Default): Unsigned 16-bit integer
                - INT16: Signed 16-bit integer
                - INT32: Signed 32-bit integer
                - UINT32: Unsigned 32-bit integer
                - INT64: Signed 64-bit integer
                - UINT64: Unsigned 64-bit integer
                - Float: 32-bit floating-point number
                - Double: 64-bit floating-point number
            - The second dropdown allows for various byte-orders to be selected (square bracketed bytes only applicable based on data type selected):

- AB CD - [EF GH] (default)
- [GH EF] - CD AB
- BA DC - [FE HG]
- [HG FE] - DC BA

- Since many controllers have more than 1 fieldbus or COM port, it is necessary to understand which COM port will be used to poll for specific remote points. The "managing COM port" is listed beside each remote point under the COM column.
  - For "NX" controllers the managing COM port depends on which COM ports have been configured with master type protocols, as well as the expanded RMT_51+ toggle option in *Configuration->Communication* which allows the user to select whether RMT_51-255 will be managed by COM2 or COM3. For more information see the OpenBAS Remote Point Truth Table.
  - For the OpenBAS-NWK-ETH3, COM1-RMT_1-1000 are managed by COM1 and COM2-RMT-1-1000 are managed by COM2.
- Slave type remote points (aka "slave expansions") allow every physical IO point (AI, BI, AO, BO) of a slave controller to be polled with the use of only 1 remote point.
  - Slave type remote points can only be used in conjunction with device addresses 100, 101, 102 and 103.
  - It is recommended to only use **OpenBAS-HV-NX10** series controllers as the slave expansion controllers in order to take advantage of all 40 UI, 40 BO and 10 AO available in the master controllers internal point database.
  - Configuring a Slave type remote point with address 100 will copy the present values of the IO points from the slave controller (AI_**1-8**, BI_**1-8**, BO_**1-8** and AO_**1-2**) into the master controller's AI_**9-16**, BI_**9-16**, BO_**9-16** and AO_**3-4**. Similarly, configuring a Slave type remote point with address 101 will copy that slave's IO points into the master controller's AI_17-24, BI_17-24, BO_17-24 and AO_5-6 and so on.
    - Configuring a Slave type remote point with address **101**, *without* configuring a slave type remote point with address **100**, will still copy into the master controller's AI_**17-24**, BI_**17-24**, BO_**17-24** and AO_**5-6**. This means the master controller's AI_9-16, BI_9-16, BO_9-16 and AO_3-4 will continue to be unused in this scenario.
    - Do not configure separate remote points for any points that are already included as part of a Slave expansion.
  - Reserved remote points for Slave expansions.
    - When using the OpenBAS-HV-NX10D, RMT_51-54 are reserved for slave expansions when managed by com2 or com3 (Opto22/N2 type), but they are not reserved when managed by COM1 (Modbus type) since only RMT_1-50 can be configured as slave expansions in COM1.
    - When using the OpenBAS-HV-NX10D, RES_FLT_41-44 are reserved for slave expansions.

        - When using the OpenBAS-ACC-32KNV or similar memory expansion, RMT_51 to RMT_54 and RES_FLT_51 to RES-54 are **not** reserved.
- Wireless (WLS) type remote points are used to configure points such as temperature or humidity from an OpenBAS-HV-WLSTH wireless temperature and humidity sensor. The OpenBAS-HV-WLSTH device(s) must first be configured in the Configuration->Wireless devices tab (see Connecting wireless devices).
    - Wireless type remote points are only supported on COM2 managed Remote points 1-255 (see the OpenBAS Remote Point Truth Table).
        - At least COM2 must be set as Optomux Master. Make sure that COM2 shows up in the COM column of the Configuration->Remote points table. If not, please review the OpenBAS Remote Point Truth Table.
    - After selecting WLS from the Object type dropdown, select a point type and the ID of the specific OpenBAS-HV-WLSTH which can be found in Configuration->Wireless devices tab (the first wireless device added would be ID 1 and so on).
- RMT_1 to RMT_10 on "NX" controllers, and when managed by COM2 have an adjustable poll rate or priority setting which can be adjusted in the Configuration->Communication tab.
    - Select RMT_1 to RMT_X (max 10) and then select a multiple of 200ms for the polling period.
    - The controller will stop polling non prioritized remote points whenever the time since the prioritized points were last polled has reached the selected polling period for prioritized remote points. Once the prioritized points have been polled, the controller will continue polling the remote points where it left off.
- Last point to poll
    - Each fieldbus has a "Last point to poll" setting which will cause the controller to stop polling the present value of all remote points after the selected remote point channel, even if they are configured.
    - Note: Similar to the Last point to poll setting, an expanded RMT or RES_FLT point enabled by the OpenBAS-ACC-32KNV memory expansion on "NX" controllers with an address of zero **and** a type of NULL will disable all higher remote point channels for RMT or RES_FLT remote points, respectively.
- Offline points
    - The controller will attempt to poll all configured remote points whether they are online or offline.
    - The controller will wait up to 200ms for a slave controller to respond to each request meaning every offline point will add about ~180ms each to the poll period (cycle time).
        - It is important to note that **all configured remote points must be online for optimal communications**.
- MP remote points on OpenBAS-HV-NXVAV Series (Polling Belimo MP Data pool values)
    - Any of RMT_1-50 can be configured to poll MP data pool values from devices on the MP bus.

- Either COM1 must be set as Modbus Master, or COM2 must be set as Optomux master (if both are set, priority is given to Modbus and RMT_1-50 are managed by COM1).
- Set the Address cell to the MPX address, select the MP data pool Object type, and enter the Data pool ID into the Channel number cell.
  - Addresses MP1 through MP15 are supported in Remote points.
  - Data pool ID's can be found in the respective devices documentation from Belimo.
  - Data pool ID's up to 255 are supported.

Configuring IP remote points

The OpenBAS-NWK-ETH3 has support for 250 IP remote points. Each IP remote point can be configured as a BACnet/IP or Modbus/TCP point. After clicking the Configuration->IP remote points tab, the System Design Studio will begin reading remote point configurations from the controller. IP remote points exist on each ETH3, but their present value is periodically updated with the present value of a point existing on another controller that is located on the same local IP network. Similar to Remote points, it is important to consider the Last point to poll setting.

- First configure the general IP settings found in the Configuration->Communication->IP section.
  - Enable the BACnet/IP or Modbus/TCP clients using the checkboxes.
  - Enable Channel 0 if needed, this only affects BACnet/IP IP remote points.
  - Timeout (s) is how long in seconds (20-255) without an update before each IP remote point will be shown as offline. It is recommended to set this ~1.5 times greater than the total IP remote point cycle time. You can quickly calculate a safe estimate for the IP remote point cycle time as CycleTime = 0.1*Last point to poll.
  - Last point to poll is the highest IP remote point channel number to be polled. In other words, it will cause the controller to stop polling the present value of all IP remote points after the selected IP remote point channel, even if they are configured.
  - Close Modbus sockets after LPTP, when checked, will close and open a new socket after each polling cycle. This is slightly less efficient but may be required by some Modbus/TCP slave devices.
- Use the main drop-down box to switch between viewing the IP_RMT and Sockets tables.
- From the Sockets table, configure the IP device's protocol, IP address, port, and network/address if applicable.
  - For BACnet IP remote points, the Network checkbox allows specifying a BACnet Network number and Address (MAC) when communicating through a BACnet router.
- After configuring Sockets, **be sure to click the Sync protocol button from the IP_RMT table.** This will update the available Object types based on the selected sockets current protocol.
- For each IP remote point needed, select the Socket representing the appropriate device, and select an Object type and Channel number.

- Count is enabled for Modbus/TCP IP remote points and allows a consecutive number of IP remote points to be polled more efficiently. System Design Studio will help visualize this.
- For MODBUS type remote points, The Object type is selected from a drop down, so do not include the Object type number when entering the Channel. Ex. For the MODBUS entity number 400001 (Holding register), select Holding Register from the Object type dropdown and enter 1 for the Channel.
    - MODBUS Remote point Channels can be configured to any integer from 1 to 65535 (inclusive).
    - MODBUS Remote points configured as Input registers or Holding registers can read or Set a Present value to any integer from 0-65535 (inclusive).
- The OpenBAS-NWK-ETH3 supports various multi-byte encodings or float formats when polling Holding Registers or Input Registers over MODBUS/TCP.
    - When configuring a MODBUS/TCP based IP remote point, and selecting Holding Register or Input Register as the Object type, two dropdowns will appear directly beneath the Object type dropdown.
        - The first dropdown allows for various data types to be selected:
            - UINT16 (Default): Unsigned 16-bit integer
            - INT16: Signed 16-bit integer
            - INT32: Signed 32-bit integer
            - UINT32: Unsigned 32-bit integer
            - INT64: Signed 64-bit integer
            - UINT64: Unsigned 64-bit integer
            - Float: 32-bit floating-point number
            - Double: 64-bit floating-point number
        - The second dropdown allows for various byte-orders to be selected (square bracketed bytes only applicable based on data type selected):
            - AB CD - [EF GH] (default)
            - [GH EF] - CD AB
            - BA DC - [FE HG]
            - [HG FE] - DC BA
- BACnet/IP remote points have a configurable priority level within the Override column. This setting is saved on the controller and affects all overrides sent to the specific IP_RMT.

Connecting wireless devices

Wireless devices such as the OpenBAS-HV-WLSTH can be added or configured in the **Configuration->Wireless devices** tab after connecting* an OpenBAS-HV-RF433R wireless receiver module (supported by all NX-based controllers with COM2).

*Do not connect or disconnect the OpenBAS-HV-RF433R or any other accessories or modules while the controller is powered.

- Use the “no” symbol toggle button to enable or disable the wireless receiver.
- Add or configure up to 10 wireless devices.
    - Each wireless device is identified with a group number (1-10) and address (1-199).
    - A green circle indicates the controller is receiving packets from the wireless device and a red circle indicates that there is an issue (Device offline, out of range, wrong group or address).
- Remove a wireless device by hovering over the device’s row and clicking the garbage can icon.
- View all information from a device by hovering over the device’s row and clicking the magnifying glass with a plus symbol.
    - Group: 1-10 (typically used to group wireless devices in a zone).
    - Address: 1-199.
    - Online: true or false (by default, a wireless device is marked offline when no packets are received for 90 seconds).
    - Mode: Off / Auto / Manual.
    - Temperature: displays the temperature sensed by the wireless device.
    - Humidity: displays the relative humidity sensed by the wireless device.
    - Fan speed: 1, 2, and 3 is the fastest (the fan speed to be used when Mode is set to Manual).
    - Occupied temperature setpoint: the temperature setpoint to be used when the zone is occupied (19-27 °C).
    - Unoccupied temperature setpoint: the temperature setpoint to be used when the zone is unoccupied (19-27 °C).
    - Humidity setpoint: the relative humidity setpoint in percent (0-100%).
    - Proportional band: 1°C / 1.5°C / 2°C (the temperature range around the setpoint where the heating, cooling or fan states will not change).
- Individual remote points must be configured (managed by COM2) for each wireless device-based point needed (see Configuring remote points).

Backup to disk

A controller’s entire database can be backed up into a single text-based *.json file and be stored or transferred as needed. For script based controllers, this may be less important. Right click a controller from the side panel and select “**Backup to disk**”. Select whether to backup Logic, Setpoints, Configuration or some combination of all 3 options and select OK. Browse to an appropriate location and type a descriptive file name.

- Since a backup can contain different sets of information (i.e., setpoints only, COM3 settings, PLC 2, expanded remote point configurations etc.), it is important to describe exactly what is included in the file name and be as detailed as possible.
- It is recommended to create a folder structure in Windows Explorer similar to the project structure to prevent confusion. I.e., /project name/building 1/HVAC/slave address 1/xxxxxxLSC.json (LSC for Logic + Setpoints + Configuration).

- Backing up Logic, Setpoints and Configuration separately allows for each part of the controller database to be restored individually as well.
- Selecting Logic will backup PLC1, PLC2* and PLC3* instructions. *When available
- Selecting Setpoints will backup Internal bytes (ADB), Internal Integers (ADI) and Internal floats (ADF).
- Selecting Configuration will backup all settings found within the Configuration tab along with all point names.

Restore from disk

To restore a previously saved backup (*.json) file, right click a controller item from the side panel and select "**Restore from disk**". Browse to and select the appropriate backup file and click Open.

- Restore from disk will restore everything that is in a backup file. For example, if you backed up both Logic and Setpoints into a single file, it would not be possible to restore only Logic, or only Setpoints.
    - Note that restoring a Configuration backup can make a controller go offline since the communication settings may change.
- The database size and controller type that the backup (including Logic, Configuration, or both) was created from must match the database size and controller type of the controller being restored to. For example, a backup including either Logic, Configuration or both that was made from a controller with the standard database can only be restored to a controller with the standard database. Since all database sizes support the same set of Internal bytes (ADB), Internal integers (ADI) and Internal floats (ADF), a backup of Setpoints only can be restored to any NX-based device regardless of whether it is an OpenBAS-HV-NX10D or has a memory expansion accessory.

MiPages for AI (Beta)

Mircom's OpenBAS-NWK-ETH3 user customizable web pages, or MiPages for short, reinvents the user customizable webpages previously supported on the OpenBAS-NWK-ETH3, with a focus on leveraging AI. A simple and intuitive HTML data-attribute notation was developed that allows custom web pages to read and write points from any controller in the system, in real-time. With a provided template, it's easy enough for anyone to use, but why not work faster with AI? Simply attach a background image (or generate one), the MiPages documentation in markdown format, and a points list, and ask your favourite AI to create your web page. See LT-6796 - Mircom OpenBAS MiPages for AI User Guide for more information.

- While the possibilities are endless, we have supplied an example web page in the documentation that is based on a background image with point values overlayed. Each point opens a pop up when clicked that allows the operator to Set, Override or Auto the point.
- Individual forms or buttons can be created for Set, Override and Auto.
- The rate at which point values are updated defaults to 10 seconds but is user adjustable.

- Up to 51 MiPages can be created but must be uploaded as any 8 character (at most) filename with ".htm" extension.
    - The web_idx.ini file must be updated to map from each MiPage file to wf-0.htm through wf-50.htm. The sample file can be found at C:\OpenBAS\Documents\ETH3_WebPagesUser_OpenBAS\ after installing the System Design Studio.
    - Users will be redirected to the file mapped to wf-0.htm upon logging in through a web browser.
    - MiPages and the updated web_idx.ini file can be copied to the ETH3's USB flash memory stick manually or remotely at http://<ETH3_IP>/protect/usbupl.htm; they will go directly into the /_OpenBAS/ folder.
    - Note that due to a legacy feature in the ETH3, every '%' character in a MiPage file must be excaped by doubling it '%%' (two percent signs).
- MiPages can be accessed by browsing or linking to the wf-x.htm files, e.g. http://<ETH3_IP> /protect/wf-xx.htm .
- Sample prompt for AI: *Following the attached MiPages for AI documentation, create a webpage with the attached AHU image as the main focus. Analyze the image, and for each element you find, overlay the appropriate point label and value. Use ETH3 com1 slave 6 for all points. Note that regarding airflow, the top right is the return air, top left exhaust, bottom left is fresh air, and bottom right is supply air. Do your best to place the elements, but make sure they do not overlap. I will fine-tune the locations.*
Make the page look modern and add common sections (specs/title/nav bar etc) you would expect from a BAS interface.
    - We recommend asking AI to do its best to locate the point values over the image, but to make sure they're spread out enough to allow you to fine tune the locations afterwards.
    - The template script in the documentation has a feature where you can click anywhere on the background image, and the coordinates (in %) will be logged to the browsers console for easy fine-tuning by the user. In our testing, this feature was always included in the generated web page, but if you have issues, you can add or fix it manually or ask the AI for help.
    - Either list the points you want or attach a points list. You can export a full or partial points list from the System Design Studio Names table or copy from a script file.
    - Specify which controller(s) you want points to be read from by including an IP address (if not from the ETH3 hosting the MiPages) the connection path (ETH3 com1/com2 etc.), and finally the address for slave controllers.
- For the background image, integrators can use their existing graphics and graphics tools if available, otherwise it is probably fastest to take a photo of the equipment. You can even try using AI to generate your equipment graphics based on a quick hand drawing, however your mileage may vary here (we had the best results using Gemini Flash 2.5).
    - Images must be named in the format image01.jpg (*without* a leading dash): image01.jpg through image50.jpg.
        - Place them in the same folder as the imagecvt.exe utility and run it to compress the images into the gzip format required by the OpenBAS-NWK-ETH3. The resulting compressed images will have a leading dash: -image01.jpg through -image50.jpg. Note that the *.jpg extension is kept for browser compatibility.

        - The imagecvt.exe utility can be found at C:\OpenBAS\Documents\ETH3_WebPagesUser_OpenBAS\ after installing the System Design Studio.
    - Sample prompt for AI: *Based on the attached hand drawing, create a high quality, professional looking 3D graphical representation of an Air Handling Unit (AHU). The graphic should be a side view of a cross section of the AHU. As detailed in the hand drawing, there should be the following elements:*
      rectangular ducts forming a sideways "H" (light grey with 3D shadows)
      3x dampers (use the same icon for all 3)
      1x heating coil (show in red)
      1x cooling coil (show in blue)
      1x fan
      1x return air temp sensor probe
      1x supply air temp sensor probe
      1x static air pressure sensor probe (supply)
      All text and arrows in the drawing are for reference only. Do not include any text in the graphic. Use realistic icons for each element. Do not include any extra elements. All elements should be in the locations shown in the hand drawing. The background can be pure white.

SPI cache

The SPI cache is an internal feature that limits how often the OpenBAS-NWK-ETH3 communicates with the SPI connected NX10 series controller. In certain scenarios, like when reading and writing many SPI-NX points from the ETH3's PLC, the SPI bus can become overloaded and these features can help. These settings don't typically need to be modified unless instructed to by Mircom support.

SMS alerts

SMS or text message alerts can be sent using an OpenBAS-NWK-NXSMS controller. An active standard size SIM card is needed **on a GSM network**. When connecting to an OpenBAS-NWK-NXSMS controller and opening the Configuration main tab, the "**SMS alerts"** sub-tab becomes available. It is important to note that the OpenBAS-NWK-NXSMS controller uses Section three from the Dynamic memory to store the information needed for the SMS alerts and so Trends 13-16, PLC2 and 4KB of script storage are unavailable.

- Up to 20 SMS messages can be configured.
- Use the Enable/Disable option to disable a specific SMS alert without having to clear the configured details.
- RES_BIT channel selects the RES_BIT control point used to trigger the controller to send an SMS alert. SMS alerts will be sent when the present value of the selected RES_BIT changes from 0 (zero) to 1.
- Repeat allows the SMS alert to be resent up to 10 times provided that the present value of the selected control point stays at 0 (zero). Set to 0 (zero) to disable resending the SMS alert and only send the first, original alert.
- Repeat rate determines the time in Minutes between resending each SMS alert and must be between 1 and 60 Minutes.
- Phone number is the cellular phone number associated with the SIM card that is inserted in the OpenBAS-NWK-NXSMS controller. Area or country codes must be entered here if needed.

- Alert text is the actual body of the SMS alert that will be sent to the Phone number. You can create messages up to 64 characters in length.
- The Refresh button can be clicked to reload the SMS configuration from the controller at any time.

Configure from script (EZ-Script)

To compile a script and load the compiled configuration to a controller, right click any controller item (OpenBAS-NWK-ETH3 supported as of SDS 2.0.0 / firmware 3.17.0) from the side panel and select "**Configure from script**". For controllers outside of projects, browse to and select an existing or new script folder (any windows directory). Checking the Quick configuration checkbox will only configure the controller with parts of the script that have changed since it was previously loaded. Alternatively, unchecking Quick configuration allows you to specify whether to configure the controller with Logic, Setpoints, Configuration or some combination of all 3 options. After clicking Configure, a confirmation dialog will appear. To test compile your script, select Compile and click Start. When the Compile process completes, the compilation details will be opened in Notepad. When configuring a controller from script within a project, there is a checkbox to Save script to main Eth3. If there is a USB flash memory stick installed in the Main OpenBAS-NWK-ETH3, it is recommended to leave this checkbox checked. Select Compile + Configure and click Start to configure the controller from your script; a loading bar will appear showing the progress. Once finished, the details will be opened in Notepad. If there is a compiler error, a text file will be opened in Notepad that outlines the error details.

- Script directories are identified by, and must include, a project.nx5 file. This file is created automatically; however one can manually create a project.nx5 file by creating and renaming an empty *.txt file to "project.nx5". Make sure that you set Windows Explorer to show file name extensions otherwise you may erroneously rename your text file to "project.nx5.txt".
- EZ-Scripts can be split into up to 10 separate script files and a definition file. Blank script_1.txt and script_def.txt files will be created automatically if they do not already exist when clicking the Configure from script->Edit button, or [controller right click menu]->Create/Edit script button. When manually creating script files, the file names must begin with "script_1" through "script_10" and be used consecutively as needed. The definition file's name, if used, must begin with "script_def". All script files must have the *.txt extension.
    - No linking of any kind is needed, the compiler will begin by joining all files together (in order, starting with script_def.txt followed by script_1.txt and so on) before proceeding.
    - Users can modify file names so long as they begin with the correct prefix and have the correct *.txt extension.
    - The System Design Studio will show a Warning if it detects any missing script files. For example, if script_7.txt is found, the System Design Studio expects script_1.txt through script_6.txt to exist as well. The script_def.txt file is optional.
- A /Compile/ directory is created within the script directory the first time a script is compiled to store various compilation files. This was added in 1.2.0 to clean up the main script directory and support custom script file names (suffixes).

- An /Archive/ directory is created within the script directory the first time any script sync status is checked. Script files are compressed into zip files and stored in the /Archive/ folder as part of the script sync status checking process.
- When using Notepad++, a custom language file will be installed to support syntax highlighting. **With an EZ-Script file open in Notepad++, click Language->OpenBAS EZ-Script to enable syntax highlighting for that file.**
    - Notepad++ significantly improves workflow and is highly recommended.
- As of SDS 1.3.1, the Notepad++ OpenBAS EZ-Script language comes with support for code folding, the ability to collapse sections of script.
    - **Within any script-comment**, indicate the start of a collapsible section of code with "{" or "fold_start" or "fold start" (without the quotations).
        - Ex: // any comment followed by {
    - **Within any script-comment**, indicate the end of a collapsible section of code with "}" or "fold_end" or "fold end" (without the quotations).
        - Ex: // any comment followed by }
- To change the EZ-script language from English, the first line of script_def.txt (if used) or script_1.txt must include the appropriate language keyword only.
- The first 199 point names defined (keyword DEFINE) will be saved to the controller. The DECLARE keyword can be used instead of DEFINE for point names that should not be loaded to the controller.
    - By default, all channel numbers of an Object type, up to the highest channel number with a point name Defined, will be discoverable through BACnet. Declared point names will not be discovered if their channel number is higher than the highest Defined point name of the same object type.
- See LT-6631 – OpenBAS EZ-Script Programming Manual for more information on EZ-Script programming.

Script status

Within a Project, there are 3 icons that may appear to the right of any controller's name in the side panel related to the controller's script status. No icon implies that no script files exist for the controller. Note that the Main OpenBAS-NWK-ETH3 must have a USB flash memory stick connected for the Script synchronized icon to appear after configuring the controller from script.

- Unknown status,
    - User should right-click the project->Check script status.
- Script unsynchronized,
    - The controller has not been configured from the current script files.
- Script synchronized,

- The current script files match the script files last used to configure the controller from script, from within this same Project.
- Note that this is not a guarantee that the controllers configuration matches the current script files.

Batch programming

As of SDS 2.0.0, batch programming allows users to configure many controllers from script with a single click. Batch programming requires all controllers to be part of a single Project, and Projects heavily rely on a USB flash memory being installed in the main OpenBAS-NWK-ETH3.

Note: If BACnet/MSTP is being used for any of the controllers being configured, it is very important to adjust the Max Info Frames settings of any Master controllers such as a JACE-8000 and the OpenBAS-NWK-ETH3. If the project allows, it would be recommended to set the JACE-8000 Max Info Frames setting to 1, and the OpenBAS-NWK-ETH3 Max Info Frames setting to at least 50 during the Batch programming process.

1. Create or associate script files to controllers.
    a. Right click any controller and select Create/Edit script. If no script files exist, this will create an empty script_def.txt and script_1.txt file, and if Notepad++ is installed, this will open the script_1.txt file in Notepad++ with the OpenBAS EZ-Script language pre-selected. Write or copy in your script and save your changes.
        i. Within Notepad++ it is recommended to go to File->Open Folder as Workspace… and browse to and select the Project directory, in order to open a side panel showing your project directory and script folders. This will happen automatically if Notepad++ is first launched by the System Design Studio.
    b. If a script is common among multiple controllers, then once the script has been created and saved, right click the controller with the script, and select Copy script. Then select one or more controllers by left clicking the first controller, and shift + left-clicking the last controller, or ctrl + left-clicking each individual controller. Then right click one of the selected controllers and select Paste script. This will duplicate the copied script file(s) into all selected controllers.
        i. Note that when shift-selecting multiple controllers, all slave controllers whether visible or hidden(collapsed) will still be added to the selection.
    c. Script directories are located within the Project directory and have hardcoded folder names based on the way they are connected, e.g. [project directory]/1.2.3.4-COM1-S001-CTRLRNAME/.
2. Select multiple controllers to be Configured from script.
    a. A range of controllers can be selected by left-clicking the first, and shift + left-clicking the last controller.
    b. Individual controllers can be added or removed from the selection by ctrl + left-clicking controllers.
    c. A counter at the top of the side panel indicates how many controllers are selected.

   d. Note that when shift-selecting multiple controllers, all slave controllers "between" the selection, whether visible or hidden(collapsed), will also be included in the selection.

3. Right click any of the selected controllers and select Configure from script.
   a. The Data to load options are the same as when configuring a single controller from script. *See Configure from script*.
   b. Blue icons identify the selected controllers.
   c. Max number of threads allows for multiple controllers to be Configured from script simultaneously assuming there are multiple OpenBAS-NWK-ETH3 gateways in the project. Each OpenBAS-NWK-ETH3 enables an additional thread, however each additional thread increases the CPU and memory load of the PC, so use with caution.
4. Selecting Start will begin the process of compiling and configuring all selected controllers with their script files.
   a. After each controller is Configured from script, it's script files will be saved to the main OpenBAS-NWK-ETH3's USB flash memory stick along with an up-to-date project file.
   b. As the process finishes with each controller, a document icon beside each controller's loading bar can be clicked to open the compilation results file (or error file).

File transfer

Depending on the memory accessories and Dynamic memory settings, files up to 112 KB in size can be stored in the memory of any "NX" controller. This can be useful for storing the latest copy of a script for example. Bigger files can be transferred to a USB flash memory stick that is connected to an OpenBAS-NWK-ETH3 using the web interface.

- To upload a file to an "NX" controller, first make sure there is enough memory by checking in **Configuration->Dynamic memory**, and then right click the controller item from the side panel and select File transfer.
  - Under Select file(s), select all files to be compressed into a zip file (and later stored on the controller).
  - Under Save as: (in the Upload section) browse to an appropriate location, enter a filename and click Save.
  - Click the Upload button to begin transferring the compressed *.zip file to the controller.
    - Note that if the file is the same as the file previously uploaded to the controller then a warning will appear.
- To download a file from the controller to a PC, right click the controller item from the side panel and select File transfer.
  - Under Save as: (in the Download section), browse to an appropriate location, enter a file name and click Save.
    - Note that if you select an existing *.zip file, after clicking Save, the System Design Studio will check if the file on the controller matches the file on your PC and display a message beside "File match:".
- To transfer a file to a USB flash memory stick connected to an OpenBAS-NWK-ETH3:

- Make sure the USB flash memory stick is formatted as FAT32 and connected to your OpenBAS-NWK-ETH3.
- Browse to http://[ETH3's IP address]/protect/usbupl.htm in a web browser (Google Chrome recommended), choose your file* and click Save to USB.
  - *Files transferred to or from an OpenBAS-NWK-ETH3 must use the 8.3 format with a maximum of 8 characters for the file name and a maximum of 3 characters for the extension.
  - In order to access the web pages, you will need to log in. The default username and password are: Username: admin Password: mircom.
  - The file will be saved into the ROOT/_OPENBAS/ folder.

- To transfer files from a USB flash memory stick connected to an OpenBAS-NWK-ETH3 to your computer:
  - Determine the file name and extension of your file that exists within the ROOT/_OPENBAS/ folder of the USB flash memory stick. If the file was transferred using the web interface this can be determined from looking at the log file:
    - Browse to http://[ETH3's IP address]/protect/usbdwn.htm and select the Load file contents below button (an empty text box will default to the log file, USB_DIR.TXT).
    - In order to access the web pages, you will need to log in. The default username and password are: Username: admin Password: mircom.
  - Type the name of your 8.3 format-named file into the text box and click:
    - Download will download the file to your browsers Download folder.
    - Load file contents below will only be enabled for *.txt, *.htm or *.jsn files and will simply show the contents within the web page.

Logic and the PLC

The main Logic tab allows for ladder logic-based programming through an intuitive, easy to use, drag and drop style interface. By default, opening the Logic tab will begin loading PLC 1 and a drop-down menu at the top right of the screen can be used to switch to PLC 2 or PLC 3 when available. The controller begins processing the 1st instruction of PLC 1 and continues consecutively until an End* instruction is reached. Once an End* instruction is processed in PLC 1, the controller will begin processing the 1st instruction of the next available PLC. If both PLC 2 and PLC 3 are not available, then the controller will begin processing the 1st instruction of PLC 1 again.
*Note that the first End instruction processed after a Call (Subroutine) instruction will return to the next consecutive instruction after the originating Call instruction (see the Call instruction overview for more information).

- Each PLC is filled with End instructions by default, and they must be deleted before adding a new instruction.
- To add an instruction, drag (click and hold) any instruction from the instruction list and drop (release left mouse button) into an empty PLC instruction area.

  - While “dragging”, the mouse icon will change from a No icon to a + icon to indicate that the instruction can be “dropped” at the mouse’s current location.
- To delete an instruction from an instruction rung hover over it and click the circled X icon that appears on the right side.
- Instructions can be disabled (skipped by the controller) without having to delete them by hovering over an instruction and clicking the No icon. A Disabled instruction’s background colour will change and the instruction can be re-enabled by hovering over and clicking the circled checkmark icon.
- To edit or modify an instructions inputs or outputs, hover over and click the pencil icon, modify as needed and then click the checkmark to save the changes (changes will be saved to the controller if in online mode). To revert any changes, click the X icon.
  - **Note:** The channel number text box does not allow invalid numbers such as empty or blank. This means you cannot backspace or clear the channel number before entering a new one. The correct process is to highlight the current Channel number either by double clicking or pressing Shift + (Left or Right arrow keys).
- Live values of the points used in any instruction can be viewed by left clicking the instruction. When multiple instructions are selected (left click the first, Shift left click the last), the first 3 instructions selected will show live values.
  - **Note:** Since the instructions are processed by the controller much faster than the live values are updated, the values may not accurately represent their value at the time the instruction was processed. If the value of a point needs to be known at a specific instruction, the Debug mode and Debug point can be used to halt the controller. See further down in this section for more information.
- Offline mode allows for copying or removing blocks of instructions without affecting the logic currently running on the controller. Offline mode is represented by the rung numbers at the left side of the PLC being highlighted orange, along with an orange Logic tab icon. Each PLC has its own online or offline state which is quickly visible in the Logic tabs right click context menu.
  - To enter offline mode left click (and optionally hold shift key and left click another instruction) to select one or more instructions and then:
    - Drag and drop the selection onto another instruction to copy and paste all selected instruction rungs above the instruction they were “dropped” on. This will increase the total amount of rungs that can be viewed/modified in the current PLC.
    - Right click any instruction that is not currently selected and select Copy above here to copy and paste all selected instruction rungs above the instruction that was right clicked on. This will increase the total amount of Rungs that can be viewed/modified in the current PLC.
    - Right click the selected instructions and click Remove Rung(s). This will decrease the total amount of Rungs that can be viewed/modified in the current PLC.
  - To save changes to the controller and return to online mode, right click the logic tab and select:

        - Refresh current PLC will reload the PLC from the controller and any offline changes will be lost.
        - Upload PLCX to controller will save all offline changes in PLC X to the controller.
- Debug mode allows for all logic processing to halt on the controller when a specific RES_BIT is Set to True. This is useful in a variety of applications including debugging scripts that reuse stack variables in order to view a stack variable's Present value at a specific point in the logic.
    - Debug mode can be enabled from the Logic tab's right click context menu.
    - RES_BIT_246 is the default debug point and will display a default point name in the ladder logic editor. This can be changed but the default point name is hardcoded for RES_BIT_246.
    - Debug mode being enabled is represented by an orange controller name in the side panel with an orange dot beside it.
    - When the controller is halted, the controller's name is red in the side panel with a red square beside it. The Logic tabs icon will also change to red.
    - Debug mode can be disabled from the controller's right click context menu as well as the Logic tab's right click context menu.
    - To resume processing logic without disabling debug mode, select "Clear RB.X and resume" from the Logic tab's right click context menu.
- When configuring a controller from script, it may be hard to follow or understand the resulting PLC instructions within the ladder logic editor, so it is typically recommended to use one or the other.
    - There is a script.map file that gets created inside the [script directory]/Compile/ folder that details the mapping between each line of script and the generated ladder logic rungs.

Logic instruction overview

There are many instructions available for use in the OpenBAS PLC's and they are grouped into Ladder (boolean), Math, Cmp (comparison) and Others categories within the Logic tab's instruction list. Every instruction has its own specific use, and the details can be found below.

Ladder (boolean) type instructions

Basic ladder type instructions treat inputs as either True or False depending on their present value. A present value of less than 1 is treated as False and a present value of 1 or greater is treated as True. Ladder type instructions must use a Result Bit as the output. The output Result Bit will be True (1) or False (zero) depending on the specific instruction and the present value of its configured inputs.

- Make sure to name all configured points in order to avoid using them again accidentally.
- Not all instructions and object types are currently available on the OpenBAS-NWK-ETH3.

And instruction

- Configuration:
    - Select up to four input points of any type.

- Select a Result Bit for the output.

- Functionality:
  - The state of the selected output Result Bit will be True when all configured inputs are True. If any configured inputs are False, the output will be False.
  - Null or unconfigured inputs are treated as True for this instruction.

Nand instruction (inverted And)

- Configuration:
  - Select up to four input points of any type.
  - Select a Result Bit for the output.
- Functionality:
  - The state of the selected output Result Bit will be False when all configured inputs are True. If any configured inputs are False, the output will be True.
  - Null or unconfigured inputs are treated as True for this instruction.

Or instruction

- Configuration:
  - Select up to four input points of any type.
  - Select a Result Bit for the output.
- Functionality:
  - The state of the selected output Result Bit will be True when at least one of the configured inputs is True. If all configured inputs are False, the output will be False.
  - Null or unconfigured inputs are treated as False for this instruction.

Remote online (Or instruction)

In next release this will appear as its own instruction however behind the scenes it reuses code from the Or instruction, and so it can currently be used by dragging an Or instruction into a PLC and configuring according to below.

- Configuration:
  - Select the first input as any Remote point (including Result Float type remote points) to check the online status for and leave the other three inputs as blank/null.
  - Select RES_BIT_245 for the output.
- Functionality:
  - The state of Result Bit 245 will be True when the selected Remote point is online and will be false when the selected Remote point is offline.
  - Typically this is then used to run alternative logic.

Nor instruction (inverted Or)

- Configuration:
  - Select up to four input points of any type.
  - Select a Result Bit for the output.

- Functionality:
    - The state of the selected output Result Bit will be False when at least one of the configured inputs are True. If all configured inputs are False, the output will be True.
    - Null or unconfigured inputs are treated as False for this instruction.

Xor instruction (Exclusive Or)

- Configuration:
    - Select one input point for the two left most logic gates and a second input for the two right most gates. The System Design Studio will automatically configure the corresponding gates as changes are made.
    - Select a Result Bit for the output.
- Functionality:
    - The state of the selected output Result Bit will be False when both configured inputs are False, True when exactly one of the configured inputs is True and False when both configured inputs are True (often remembered with the phrase “One or the other but not both”).
    - Null or unconfigured inputs are treated as False for this instruction.

Nxor instruction (Also known as Xnor, inverted Exclusive Or)

- Configuration:
    - Select one input point for the two left most logic gates and a second input for the two right most gates. The System Design Studio will automatically configure the corresponding gates as changes are made.
    - Select a Result Bit for the output.
- Functionality:
    - The state of the selected output Result Bit will be True when both configured inputs are False, False when exactly one of the configured inputs is True and True when both configured inputs are True.
    - Null or unconfigured inputs are treated as False for this instruction.

Invert instruction (Also known as Not)

- Configuration:
    - Select one input point of any type.
    - Select a Result Bit for the output.
- Functionality:
    - The state of the selected output Result Bit will be True when the input is False and False when the input is True.
    - A null or unconfigured input is treated as False for this instruction.

And/Or instruction (A·B+C·D)

- Configuration:

  - Select four input points of any type for A and B on the top line, and C and D on the bottom line.
  - Select a Result Bit for the output.
- Functionality:
  - The state of the selected output Result Bit will be True when A and B are both True, or when C and D are both True (or any three or four inputs are True).
  - A single null or unconfigured input on either line will be treated as True but two null or unconfigured inputs on either line will be treated as False.

Or/And/Inv instruction (And/Or inverted)

- Configuration:
  - Select four input points of any type for A and B on the left, and C and D on the right.
  - Select a Result Bit for the output.
- Functionality:
  - The state of the selected output Result Bit will be True when at least one of A or B are False, and at least one of C or D are False. If A and B are both True, the output will be False. Similarly, if C and D are both True then the output will be False.
  - A single null or unconfigured input will be treated as True. If both A and B are left null or unconfigured, they will be treated as False. Similarly, if both C and D are left null or unconfigured, they will be treated as False.

Math type instructions

Math type instructions perform some type of calculation on the configured inputs and store the result in a Result Float type point.

- Note that if a Remote point is offline, the value used for calculations will be 0 (zero).
- Null or unconfigured inputs will be ignored in the calculation.
- The Int point type can be used as any constant integer between 0 and 254. Select a value where you would select a channel number for any other point type. For example, Int.5 will be considered as a point with a present value of 5.
- Make sure to name all configured points in order to avoid using them again accidentally.

Min instruction (minimum)

Find the minimum value.

- Configuration:
  - Select up to four input points of any type.
  - Select a Result Float for the output.
- Functionality:
  - The present value of the selected output Result Float will equal the present value of the configured input with the lowest present value.

Max instruction (maximum)

Find the maximum value.

- Configuration:
  - Select up to four input points of any type.
  - Select a Result Float for the output.
- Functionality:
  - The present value of the selected output Result Float will equal the present value of the configured input with the highest present value.

Avg instruction (average)

Calculate an average.

- Configuration:
  - Select up to four input points of any type.
  - Select a Result Float for the output.
- Functionality:
  - The present value of the selected output Result Float will equal the average present value of all configured inputs.

"+" instruction (addition)

Calculate a sum.

- Configuration:
  - Select up to four input points of any type.
  - Select a Result Float for the output.
- Functionality:
  - The present value of the selected output Result Float will equal the total sum of all configured inputs or addends.

"-" instruction (subtraction)

Calculate a difference.

- Configuration:
  - Select up to four input points of any type.
  - Select a Result Float for the output.
- Functionality:
  - The 2^{nd} ,3^{rd} and 4^{th} inputs, the subtrahends, will be subtracted from the first input, the minuend, and the result (difference) will be stored in the selected output Result Float.
  - When configuring less than four inputs, the System Design Studio will automatically arrange them so that the first configured instruction is used as the minuend or starting number.

"*" instruction (multiplication)

Calculate a product.

- Configuration:
  - Select two input points of any type.
  - Select a Result Float for the output.
- Functionality:
  - The inputs (the multiplicand and the multiplier) will be multiplied together and the result or product will be stored in the selected output Result Float.
  - Leaving either input null or unconfigured will cause the instruction to be skipped by the controller.

"/" instruction (division)

Calculate a quotient.

- Configuration:
  - Select two input points of any type.
  - Select a Result Float for the output.
- Functionality:
  - The first input, the dividend, will be divided by the second input, the divisor, and the result (quotient) will be stored in the selected output Result Float.
  - If the present value of the 2nd input (divisor) is 0 (zero) the output will be 0 (zero).
  - Leaving either input null or unconfigured will cause the instruction to be skipped by the controller.

Cmp (comparison) type instructions

Comparison type instructions compare the present value of two operands (Op1 and Op2) and store either a True (1) or False (0) in the selected output Result Bit.

- Leaving any inputs as null or unconfigured will cause the instruction to be skipped by the controller.

">" instruction (greater than)

Determine whether Op1 is greater than Op2.

- Configuration:
  - Select two input points of any type for Op1 and Op2.
  - Select a Result Bit for the output.
- Functionality:
  - The state of the selected output Result Bit will be True when the present value of Op1 is greater than the present value of Op2 and False otherwise.

"<" instruction (less than)

Determine whether Op1 is less than Op2.

- Configuration:
  - Select two input points of any type for Op1 and Op2.
  - Select a Result Bit for the output.
- Functionality:
  - The state of the selected output Result Bit will be True when the present value of Op1 is less than the present value of Op2 and False otherwise.

"=" instruction (equal)

Determine whether Op1 is exactly equal to Op2.

- Configuration:
  - Select two input points of any type for Op1 and Op2.
  - Select a Result Bit for the output.
- Functionality:
  - The state of the selected output Result Bit will be True when the present value of Op1 is exactly equal to the present value of Op2 and False otherwise.

">=" instruction (greater than or equal to)

Determine whether Op1 is greater than, or exactly equal to Op2.

- Configuration:
  - Select two input points of any type for Op1 and Op2.
  - Select a Result Bit for the output.
- Functionality:
  - The state of the selected output Result Bit will be True when the present value of Op1 is greater than or exactly equal to the present value of Op2 and False otherwise.

"<=" instruction (less than or equal to)

Determine whether Op1 is less than, or exactly equal to Op2.

- Configuration:
  - Select two input points of any type for Op1 and Op2.
  - Select a Result Bit for the output.
- Functionality:
  - The state of the selected output Result Bit will be True when the present value of Op1 is less than or exactly equal to the present value of Op2 and False otherwise.

"!=" instruction (not equal to)

Determine whether Op1 is not exactly equal to Op2.

- Configuration:
  - Select two input points of any type for Op1 and Op2.
  - Select a Result Bit for the output.

- Functionality:
    - The state of the selected output Result Bit will be True when the present value of Op1 is not exactly equal to the present value of Op2 and False otherwise.

Other instructions

All instructions that do not fall under Ladder, Math, or Cmp are located under the Other header within the Logic tab instruction list.

- Make sure to name all configured points in order to avoid using them again accidentally.
- Many of these instructions require an index to a consecutive number of points. It is important to name and Set these points appropriately from within the **Values and trends** tab.

Label instruction

Label instructions are often used to identify what a group or section of instructions are used for. In EZ-Script, Labels are required for Jump and Call (Subroutine) instructions.

- Configuration:
    - Enter up to nine characters into the text box.
- Functionality:
    - Label instructions are for reference only.

Long label instruction

Long label instructions can only be created from script but they can be viewed and deleted from within the Logic tab. A Long label instruction uses multiple consecutive instructions to store up to 105 characters (32 characters max between spaces).

- Configuration:
    - Not applicable
- Functionality:
    - The first instruction used will display the entire label and the following instructions will remain blank.
    - The disable and edit buttons do nothing for each instruction rung involved in the long label instruction.
    - Click the delete button of any involved instruction rung to delete the entire label.

Timer instruction

After dragging in a timer instruction, hover over “TIMER” and a drop down will appear with two additional options, OSCILLATOR and OSCILLATOR(F). The TIMER option creates a 100ms, or 1000ms countdown timer that is typically used to trigger an action when it reaches 0 (zero). The OSCILLATOR option is like the TIMER option except it decrements every 50ms and has an extra output for Result Bit or Binary Output toggling. OSCILLATOR(F) allows a Result Bit or Binary Output to toggle at a rate between 0 (zero) and 10Hz depending on the input percentage value (0.0-100.0). When using the Timer

instruction, **be sure to consider the first loop of the PLC after a power cycle**, since Result Bit's are by default, cleared to 0 (zero).

- TIMER (1s or 1/10s countdown timer)
  - Configuration:
    - *SW* (switch)
      - Select a Result Bit to be used as the timers control bit. The System Design Studio will automatically select the appropriate output TMR point (RES_BIT_21-36 are the control bits for TMR_1-16 respectively).
    - *Output*
      - Select a TMR point to store the output value. The System Design Studio will automatically select the appropriate control bit (RES_BIT_21-36 are the control bits for TMR_1-16 respectively).
    - *RV* (reload value point)
      - Select an Internal integer (ADI) to store the reload value.
      - The full ADI value range is supported (up to 65535 seconds).
    - *Itvl* (interval)
      - Select a decrement interval of either 100ms or 1000ms.
  - Functionality:
    - The present value of the TMR output will be set to the present value of the selected reload value point **each time this instruction is processed, and the control bit is True (1).** The present value of the TMR output will start decrementing by 1 at the interval selected when the control bit is False (zero), and until it reaches 0 (zero).
    - For various use cases, the control bit can be left at 0 (zero) and the TMR point can be Set directly. For example when using the PLC to Set the TMR point directly, the TMR value will instantly be changed whereas Setting the control bit to True will only cause the TMR to be reloaded when the PLC processes a TIMER instruction configured for the respective TMR point.
- OSCILLATOR (0.05s countdown timer with output toggling)
  - Configuration:
    - *SW* (switch)
      - Select a Result Bit to be used as the timers control bit. The System Design Studio will automatically select the appropriate output TMR (RES_BIT_21-36 are the control bits for TMR_1-16 respectively).
    - *Output*
      - Select a TMR point to store the output value. The System Design Studio will automatically select the appropriate control bit (RES_BIT_21-36 are the control bits for TMR_1-16 respectively).
    - *Itvl* (interval)

- Select any point whose present value will be used to control the pulse interval (where the selected pulse output point will toggle every [interval/20] seconds).
- This value is limited to 1000, or 50 seconds.

▪ *PULS* (pulse output)

- Select a Binary Output or Result Bit to toggle every [interval/20] seconds. *Binary outputs are restricted from changing faster than 2.5Hz.

○ Functionality:

▪ The present value of the TMR output will be instantly set to the present value of the selected interval point when it reaches 0 (zero).
▪ The present value of the TMR output will decrement by 1 every 50ms when the control bit is True (1) and will stop decrementing when the control bit is False (zero).
▪ The selected pulse output will toggle every time the present value of the TMR output reaches 0 (zero).

▪ OSCILLATOR(F) (percentage to frequency/pulse converter)

○ Configuration:

▪ *SW* (switch)

- Select a Result Bit to be used as the timers control bit. The System Design Studio will automatically select the appropriate output TMR (RES_BIT_21-36 are the control bits for TMR_1-16 respectively).

▪ *Output*

- Select a TMR point to store the output value. The System Design Studio will automatically select the appropriate control bit (RES_BIT_21-36 are the control bits for TMR_1-16 respectively).

▪ *Freq* (frequency)

- Select any point whose present value will be used as a percentage (0.0-100.0) to control the pulse frequency between 0 (zero) and 10 Hz.

▪ *PULS* (pulse output)

- Select a Binary Output or Result Bit to toggle at the current frequency. *Binary outputs are restricted from changing faster than 2.5Hz.

○ Functionality:

▪ When the control bit is True (1), the selected pulse output will toggle at the rate determined by the frequency input (entered as an integer from 0-100) multiplied by 10 Hz. The duty cycle is roughly 50%.
▪ For reference, the TMR output value is calculated and reset based on the present value of the frequency point when the timer reaches 0 (zero). The TMR present value is decremented every 0.05 seconds when the control bit is True until it reaches 0 (zero).

Jump instruction

The Jump instruction allows for either permanent or conditional skipping of instructions. Conditional Jump instructions allow for If-Then-Else type logic.

- Configuration:
  - Select up to one input point of any type.
  - Click the gate to toggle whether the jump is conditional on the input point being True or False.
  - Enter a number between 1 and 200 for the number instructions to jump.
- Functionality:
  - When the gate is not inverted, the controller will skip the configured number of instructions when the input is True (>= 1) and will continue to the next consecutive instruction when the input is False (< 1).
  - When the gate is inverted, the controller will skip the configured number of instructions when the input is False and will continue to the next consecutive instruction when the input is True.
  - A null or unconfigured input will always skip the configured number of instructions regardless of whether the gate is inverted or not.

Call instruction (subroutine call)

Call instructions can save space (PLC instructions) and simplify logic when requiring the same functionality more than once in any program. A Call instruction allows a consecutive set of instructions located at any point in the PLC to be "jumped to" and processed whenever needed. This instruction is similar to what various programming languages refer to as functions or methods. As of SDS 1.3.1, Call instructions can now call subroutines that exist in other PLC's.

- Configuration:
  - Select up to one input point of any type.
  - Click the gate to toggle whether the call is conditional on the input point being True or False.
  - Select the PLC where the subroutine instructions will be.
  - Enter the rung number of the first instruction of the subroutine (it is recommended that the first instruction of a subroutine is a Label instruction).
  - Create the subroutine:
    - Must be located after the main END instruction of any available PLC.
    - Must end in an END instruction.
- Functionality:
  - When the gate is not inverted, the controller will jump to the configured PLC and rung number only when the input is True (>= 1) and will continue to the next consecutive instruction when the input is False (< 1).
  - When the gate is inverted, the controller will jump to the configured rung number only when the input is False and will continue to the next consecutive instruction when the input is True.

- After jumping to an instruction due to a Call, the controller will continue to process instructions consecutively until an END instruction is processed, and then return to the next consecutive instruction following the originating Call instruction.
- A null or unconfigured input will always jump to the configured instruction number regardless of whether the gate is inverted or not.
- Nested Call instructions are not supported. This means you cannot call a subroutine from within another subroutine.

End instruction

The End instructions causes the controller to skip the remaining instructions in the current PLC.

- Configuration:
  - None.
- Functionality:
  - When an End instruction is processed, the controller will begin processing the first instruction of the next consecutive and available PLC and return to the first instruction of the first PLC when all PLC's have been processed.
  - The only exception is when an End instruction is processed following a Call instruction. In this scenario, the controller will begin processing the next consecutive instruction following the originating Call instruction.

LightG instruction (lighting group)

Lighting groups allow for simplified logic when controlling many lighting circuits with similar scheduling requirements. Lighting groups allow multiple Binary outputs to be controlled with a single input. Lighting groups are not necessarily restricted to lighting control applications.

- Configuration:
  - Select between Result Bit 1 through 20 and Binary Output 41 through 60 as the input control bit.
  - Select up to eight Binary Outputs to be controlled together, as a group.
- Functionality:
  - When this instruction is processed, all configured Binary Outputs will be Set on (priority 9) when the control bit is True (1) and Set off when the control bit is False (zero).
  - Lighting groups are commonly used with Schedules and up to 10 schedules can be created to control each lighting group control bit.
  - Result Bit's 1 through 20 are reserved for lighting group control bits and can be substituted with Binary Outputs 41 through 60 respectively when priority-based control is needed. Binary Outputs 41 through 60 are internal registers only (no relays are mapped by default) that support the four priority levels (*however all 16 BACnet priorities are mapped when COM1 is set to BACnet MSTP protocol).

- A schedule configured to control RES_BIT_1 will not affect a lighting group configured with BO_41 as the control bit. Similarly, a schedule configured to control BO_41 will not affect a lighting group configured with RES_BIT_1 as the control bit.

Assign instruction (controlling outputs)

The assign instruction is most used for the control of outputs within the PLC by copying the present value from one point to another. The assign instruction allows for two separate copy operations within a single instruction. The OpenBAS-NWK-ETH3 can copy to or from points on an SPI connected NX10 series controller as well. For assigning more than two outputs in a single instruction, see the Multiple Assign instruction overview below. Warning: Do not assign an analog signal (any constantly changing value) to any EEPROM-based points (ADB, ADI, ADF). An EEPROM-based point can only be Set around 1,000,000 times before the respective memory block may fail. The ladder editor will prefix these instructions output points with "(Avoid)" for ADB, ADI, and ADF object types as a reminder.

- Configuration:
  - For OpenBAS-NWK-ETH3 only, select SPI-NX or ETH3 scope for each input and output.
  - Select up to two points of any type for the inputs.
  - Select up to two writeable (AI/BI/TMR not supported) points for the outputs.
- Functionality:
  - When this instruction is processed, the present value of each input point will be copied into the present value of the respective output point.
  - Null or unconfigured inputs or outputs will cause the respective input or output to be set as null or unconfigured as well and the respective copy operation will be ignored.

Multiple Assign instruction

The Multiple Assign instruction allows the present value of up to 40 consecutive points of any type to be copied into another 40 consecutive points of any type. Warning: Do not assign an analog signal (any constantly changing value) to any EEPROM-based points (ADB, ADI, ADF). An EEPROM-based point can only be Set around 1,000,000 times before it may fail. The ladder editor will prefix these instructions output points with "(Avoid)" for ADB, ADI, and ADF object types as a reminder.

- Configuration:
  - *#* (number of points to copy)
    - Select the number of points to copy the same way a point channel would be selected. For example, Int.10 would be interpreted as a point with a present value of 10.
  - *src ptr* (source pointer)
    - Select the first of any consecutive points whose present value will be copied from.
  - *out ptr* (output pointer)
    - Select the first of any writeable (AI, BI, TMR not supported) consecutive points whose present value will be copied to.
- Functionality:

- When this instruction is processed, the present value of the configured input points will be copied into the present value of the configured output points.
- For example, with # set to Int.3, *src ptr* set to RB.1 and *out ptr* set to BO_1, the present value of BO_1, BO_2 and BO_3 will be Set (priority 9) with the present values of RES_BIT_1, RES_BIT_2, and RES_BIT_3, respectively.

Totalizer instruction (pulse accumulator and energy metering)

After dragging in a Totalizer instruction, hover over ACCUMULATOR and a drop down will appear with two additional options which are ACCUM/PERIOD (accumulation per period), and ENERGY ACCUM (energy accumulation). The ACCUMULATOR option allows for indefinite pulse counting whereas the ACCUM/PERIOD is based on a time period and will store the total pulse count for the current and previous period only. Both ACCUMULATOR and ACCUM/PERIOD are commonly used in conjunction with flow meters. The ENERGY ACCUM option is typically used for calculating energy usage in kilowatt hours and will integrate (calculate the "area under the curve") an input every 10 minutes and add the result to the existing value.

- ACCUMULATOR
  - Configuration:
    - *src* (source)
      - Select any binary type of point (BI, BO, RES_BIT) for the pulse input source.
        - Use BI_1 and/or BI_2 for high frequency signals up to 250Hz (15,000 pulses per minute).
        - Use BI_3-8, BO_1-8 and RES_BIT_1-255 for signals up to 10Hz (600 pulses per minute).
        - Use BI_9-40, or BO_9-60 for signals up to 1Hz* (60 pulses per minute). *The supported signal frequency for these points is highly dependent on the number of configured remote points and many other factors.
    - *EVENT* (signal edge)
      - Select *To Zero*, *To One*, or *Any* to count on falling edge, rising edge or both falling and rising edges, respectively.
    - *Total*
      - Select any Result float to store the total number of the selected *EVENT* that occurs on the selected *src*.
    - *EEPROM*
      - Select any Internal float (ADF) to store a copy of the present value of *Total*.
  - Functionality:
    - The present value of *Total* will increment by one every time the selected *EVENT* occurs on the selected *src* signal.
    - If the present value of *Total* is less than *EEPROM* (usually during a power cycle), it will be Set to that value.

        - The present value of *EEPROM* will be Set with the present value of *Total* every five minutes.
        - To clear or reset an accumulation, first Set the *EEPROM* point's present value to zero and then Set the *Total* point's present value to zero. If you try to Set the present value of *Total* first, it will be Set back to the present value of *EEPROM*.
- ACCUM/PERIOD (accumulation per period)
    - Configuration:
        - *src* (source)
            - Select any binary type of point (BI, BO, RES_BIT) for the pulse input source.
                - Use BI_1 and/or BI_2 for high frequency signals up to 250Hz (15,000 pulses per minute).
                - Use BI_3-8, BO_1-8 and RES_BIT_1-255 for signals up to 10Hz (600 pulses per minute).
                - Use BI_9-40, or BO_9-60 for signals up to 1Hz* (60 pulses per minute). *The supported signal frequency for these points is highly dependent on the number of configured remote points and many other factors.
        - *EVENT* (signal edge)
            - Select *To Zero*, *To One*, or *Any* to count on falling edge, rising edge or both falling and rising edges, respectively.
        - *P/min* (Period in minutes)
            - Select between 1, 5, 10, 15, 20, 30, or 60 minute periods.
        - *Total*
            - Select any Result float to store the number of selected signal edges during the current period.
        - *LastP* (previous period total)
            - Select any Result float to store the total number of selected signal edges of the previous period.
    - Functionality:
        - The present value of *Total* will increment by one every time the selected *EVENT* occurs on the selected *src* signal.
        - When the period has elapsed (*Total* has been accumulating for the selected period), the present value of *LastP* will be Set with the present value of *Total* and then *Total* will be reset to zero.
- ENERGY ACCUM (energy accumulator)
    - Configuration:
        - *src* (source)
            - Select any Result float for the power input.
                - Use the Math type instructions to calculate Power = Current*Voltage*Power Factor. If voltage and/or power factor are

unavailable, constants can be used for estimation. Alternatively, using a current input (in Amps) will calculate Amp hours (A h).

- *Total*
  - Select any Result float to store the total sum of all samples of the source input during the current 10-minute period.
- *samples*
  - Select any Result float to store the total number of samples that have been summed into *Total* during the current period.
- *energy*
  - Select any Internal float (ADF) to store the accumulation of the power input integrations (i.e., kWh for kW input, Wh for W input, A h for A input).

- Functionality:
  - When this instruction is processed, the present value of *samples* will increment by 1 and the present value of the power input will be added to the present value of *Total*.
  - Every 10 minutes, the present value of *Total* and *samples* will be used to calculate the energy usage over that period which is then added to the present value of *energy*. Both *Total* and *samples* will then be Set to zero.

PropCtrl instruction (proportional control)

The PropCtrl instruction is often used for temperature or pressure reset control strategies. This instruction will adjust an output between *MIN* and *MAX* depending on another value such as the outside air temperature. The configurable integration constant acts as a smoothing function on the output and is often used to prevent machine cycling.

Configuration:

- *PV* (process variable or feedback)
  - Select any point for the process variable or feedback input.
- *setPoint* and *band* (proportional band)
  - Select any two consecutive Result floats (RES_FLT), Internal bytes (ADB), Internal integers (ADI), or Internal floats (ADF) to store the control setpoint and the proportional band, respectively.
    - The System Design Studio will automatically adjust each input appropriately as the other input changes.
    - *Setpoint* is the desired value of the parameter being controlled.
    - Proportional band is the typical error range or expected deviation of the controlled parameter from the setpoint.
- *MIN* (minimum), *MAX* (maximum), and *INT* (integration constant)
  - Select any three consecutive Result floats (RES_FLT), Internal bytes (ADB), Internal integers (ADI), or Internal floats (ADF) to store the minimum and maximum allowed output values as well as the integration or smoothing constant, respectively.

        - The System Design Studio will automatically adjust each input appropriately as the other input changes.
        - *MIN* and *MAX* should be set between zero and 100 for Analog outputs, otherwise they can be any valid IEEE-754 32 bit float value.
        - *INT* can be set to zero to disabled output smoothing.
    - (output)
        - Select any Result float or Analog output for the output actuator.
- Functionality:
    - The present value of the output will vary proportionally between the present values of the configured minimum and maximum points as the process variable (feedback) varies between the difference of the proportional band and the setpoint.
    - For example, with a minimum of zero, a maximum of 100, a setpoint of 50 and a proportional band of **+**10, the output will be zero (minimum) when the feedback is <= 50 (*setpoint*) and the output will be 100 (*maximum*) when the feedback is >= 60 (*setpoint* + *band*). The output will vary proportionally between 0 (minimum) and 100 (maximum) as the feedback varies from 50 (*setpoint*) to 60 (*setpoint* + *band*).
        - The logic can be flipped by making the proportional band negative. In the above example, changing the proportional band from +10 to **-**10 would cause the output to be zero (*minimum*) when the feedback is >= 50 (*setpoint*) and the output to be 100 (*maximum*) when the feedback is <= 40 (*setpoint* + *band*), and the output would vary proportionally in between 50 and 40.
    - The present value of the *INT* point, the integration or smoothing constant, affects how long it will take for the output to match the proportionally calculated value. Every second, a fraction of the remaining difference between the proportionally calculated value and the previous output will be added to the previous output. For example, if the proportionally calculated output is 10 but the current output is 5, an INT value of 2 will cause the output to increase to 7.5 (current output + difference/INT or 5 + (10-5)/2 = 7.5) after the first second, and 8.75 after the 2nd second and so on.
    - When the output is an Analog output, the minimum and maximum present values will be ignored if they are Set with values less than zero or greater than 100 as these are the absolute limits.

AHU/RTU instruction (air handling unit or roof top unit)

This instruction allows for bang-bang (also known as 2-step, on/off or hysteresis) control of a heating or cooling stage with run/enable control, minimum on and off time control, and minimum inter-stage time control (when more than one of these instructions are used together for multi-staged systems).

- Configuration:
    - *run*
        - Select any point to control whether this stage is enabled (>= 1) or disabled (< 1).

        - Typically, this will be a Result bit used as the output in a previous AND instruction to make sure the stage is enabled, the fan is on, and there is no low- or high-pressure alarm.
    - *setPt* (setpoint), *Diff* (difference), *TMRoff* (timer off), *TMRon* (timer on), *TMRitr* (timer inter-stage), *mode*
        - Select any 6 consecutive Internal bytes (ADB), Internal integers (ADI), Internal floats (ADF) or Result floats to store the above parameters, respectively.
        - The System Design Studio will automatically adjust each input appropriately as the other inputs change.
        - *Setpoint* is the desired value of the parameter being controlled.
        - *Difference* (also referred to as proportional band or deadband) is the range, centered around the *setpoint*, where the output stage will remain in its current state.
        - *Timer off* is an optional index between 1 and 16 referring to the timer point to be used for the minimum off time for this stage.
            - The timer instruction must be configured separately as either 1s or 1/10s countdown timer.
            - Set the value of the respective point to zero to disable.
        - *Timer* on is an optional index between 1 and 16 referring to the timer point to be used for the minimum on time for this stage.
            - The timer instruction must be configured separately as either 1s or 1/10s countdown timer.
            - Set the value of the respective point to zero to disable.
        - *Timer inter-stage* is an optional index between 1 and 16 referring to the timer point to be used for the inter-stage time (when more than one of these instructions are used together for multi-staged systems).
            - The timer instruction must be configured separately as either 1s or 1/10s countdown timer.
            - Set the value of the respective point to zero to disable (for each instruction when using more than one for multi-staged systems).
            - Use the same inter-stage timer for each instruction when using more than one for multi-staged systems.
        - *Mode* should be set to zero for heating and 1 for cooling.
    - *PV* (process variable)
        - Select any point for the process variable of feedback point.
    - (output)
        - Select any Binary output or Result bit for the output stage.
- Functionality:
    - When the present value of the *run* point is zero, the present value of the output point will be zero.

- When enabled in heating mode (*mode* = 0), the output stage will turn on (Set to 1) when the process variable is less than or equal to the [*setpoint* – *difference*/2] and the output stage will turn off (Set to zero) when the process variable is greater than or equal to the [setpoint + difference/2].
- When enabled in cooling mode (*mode* = 1), the output stage will turn off (Set to zero) when the process variable is less than or equal to the [*setpoint* – *difference*/2] and the output stage will turn on (Set to 1) when the process variable is greater than or equal to the [setpoint + difference/2].
- When configured, the *timer on* and *timer off* count down timers will be Set with their configured reload values each time the stage is turned on and off respectively. The timers prevent the stage from turning off or on when the *timer on* and *timer off* points are not equal to zero, respectively. If a change of state is required before the respective timer reaches zero, the output state will not change (at least until the next time the instruction is processed).
- When configured, the *inter-stage timer* will be Set to its configured reload value each time the stage is turned on. The *inter-stage timer* prevents a stage from turning on while the timer is not equal to zero. When configured on multiple AHU/RTU instructions for multi-staged systems, this timer prevents all stages from being commanded on at the same time with a configurable time between each stage.

HourCnt instruction (hour counter)

This instruction will accumulate the amount of time in hours that an input is True (1). This is commonly used for keeping track of machine or stage runtimes.

- Configuration:
  - *Src* (source)
    - Select any point to be used as the source input. This is typically a point that represents the status of a machine or stage.
  - *Cnt* (count)
    - Select any Result float to store the current 5-minute partial hour count.
  - *EE* (eeprom)
    - Select any Internal float (ADF) to store the total accumulated runtime in hours that the input was True (1).
- Functionality:
  - *Count* will accumulate the runtime while *source* is True (1) until it reaches 5 minutes (0.0833 hours) and then *count* will be added to the present value of *eeprom*, and *count* will be reset to zero.

Alt/Pll instruction (alternate and parallel)

This instruction will alternate the use of between 2 and 8 machines (pumps, air handlers, fans, chillers etc.) each time there is a “call” with optional machine-running feedback/monitoring and individual machine enables. The “call” can be in the form of an on/off binary signal or an analog signal such as

when a temperature drops below a threshold. The parallel feature can be enabled to allow the machines or stages to run at the same time, with a configurable time between when each one starts. In this case, the leading machine or stage will alternate each time there is a new "call".

- Configuration:
  - *fbIn* (feedback input)
    - Set to On to require feedback within a configurable amount of time that the machine or stage has turned on. This requires an extra input for each machine or stage, and a timer.
  - *extIn* (external enable inputs)
    - Set to On to allow each machine or stage to be enabled or disabled on demand. This requires an extra input for each machine or stage.
  - *aPV* (analog process variable)
    - Set to On when using an analog signal for the process variable or feedback. This requires that *setpoint* and *prBand* are Set to the appropriate values.
  - *PL* (parallel)
    - Set to On when the machines or stages are intended to run simultaneously. This requires a timer to prevent each machine or stage from turning on at the same time.
  - *UP/DN* (increment or decrement)
    - Select Up to increment the *leader* each time there is a "call" or Down to decrement the *leader* each time there is a "call". The present value of *leader* will wrap around when incremented or decremented passed the last machine or stage.
    - Note: Do not manually change the *leader* when the instruction is running (i.e., there is a "call").
  - *input 1* (first feedback or enable input)
    - Select the first Binary input, Result bit, or Remote point of a consecutive number of points to be used for the feedback inputs, enable inputs or both.
    - When using both feedback inputs and enable inputs, the first [number of stages] inputs will be used for the feedback and then the enable inputs will follow.
    - For example, with 4 stages, *fbIn* set to On, *extIn* set to On, and *input 1* set to BI_1, BI_1-4 will be used for the feedback inputs of the 4 stages respectively and BI_5-8 will be used for the enable inputs of the 4 stages, respectively.
  - *stage#* (number of stages), *leader*, *sTMR* (stage timer), *aTMR* (alarm timer), *setPt* (setpoint), *prBand* (proportional band)
    - Select any 6 consecutive Internal bytes (ADB), Internal integers (ADI), or Internal floats (ADF) to store the above parameters, respectively.
    - The System Design Studio will automatically adjust each input appropriately as the other inputs change.
    - *Stage#* is the number of machines or stages to be controlled.
    - *Leader* identifies which machine or stage will turn on during the next "call".

- When *PL* is On, *Leader* identifies the first machine or stage that will turn on.

▪ *sTMR* is an index to the countdown timer to be used between stages when *PL* is On.
- The timer instruction must be configured separately as either a 1s or 1/10s countdown timer.
- When using both *sTMR* and *aTMR*, *aTMR* must be configured for less time than *sTMR*.
- It is highly recommended to name or reserve this point even when not in use in case it is needed later.
- Note that in firmware 3.03.5 or older, *sTMR* cannot be 0 (zero) regardless if *PL* is On or not, when using the feedback inputs.

▪ *aTMR* is an index to the countdown timer to be used for the time before checking the feedback input after a machine or stage has been commanded on, when *fbIn* is On.
- Timer instruction must be configured separately as either 1s or 1/10s countdown timer.
- When using both *sTMR* and *aTMR*, *aTMR* must be configured for less time than *sTMR*.
- It is highly recommended to name or reserve this point even when not in use in case it is needed later.

▪ *Setpoint* is the desired value of the parameter being controlled and is only used when *aPV* is set to On.
- This setpoint is intended to be a constant and care should be taken to not Set this point more than once every 5 minutes (see ADB, ADI and ADF overviews in Viewing and understanding the point database).
- It is highly recommended to name or reserve this point even when not in use in case it is needed later.

▪ *prBand* (also referred to as deadband) is the range below (less than) the setpoint that is allowed before initiating a “call” and is only used when *aPV* is set to On.
- It is highly recommended to name or reserve this point even when not in use in case it is needed later.

○ PV (process variable)
▪ Select any point for the process variable or feedback input. Typically, Binary inputs or Result bits are used when *aPV* is set to Off and Analog inputs or Result floats are used when *aPV* is set to On.

○ (output)
▪ Select the first Binary output or Result bit of a consecutive number of points to be used as the outputs.
▪ For example, with 4 stages and BO_1 selected as the output, BO_1-4 will be commanded for the 4 stages, respectively.

- Functionality:

- When using a binary process variable, the current *leader* will turn on when the process variable is zero (i.e., there is a "call"). When the process variable changes to 1, the current *leader* will turn off and then *leader* will increment or decrement according to *UP/DN*.
- When using an analog process variable (*aPV* is On), the current *leader* will turn on when the process variable is less than the [*setpoint* – *pBand*]. When the process variable becomes greater than or equal to the setpoint, the current *leader* will turn off and then *leader* will increment or decrement according to *UP/DN*.
- When *PL* (parallel) is On, *sTMR* will begin counting down to zero when each machine or stage is commanded on. If there is still a "call" when *sTMR* reaches zero, the next consecutive output will be commanded on. This process will continue until all outputs are commanded on or the setpoint has been reached (or binary input changes to 1).
- When using the feedback inputs, *aTMR* will begin counting down to zero when any machine or stage is commanded on. If the respective feedback input is zero when *aTMR* reaches zero, the stage will be commanded off and the next consecutive stage will be commanded on.
- When using the enable inputs, any machine or stage whose enable input is 0 (zero) will be skipped.

In Range instruction (and hysteresis/deadband)

Depending on the configuration, this instruction provides a boolean (on/off) output when the input is either within boundaries, outside of boundaries, crossing above boundaries, or crossing below boundaries. After dragging the In Range instruction into a PLC, hover over "In Range" and a dropdown box will appear with an option for Hysteresis. Use In Range to check if an input is within or outside of boundaries and use Hysteresis for simple deadband control.

- In Range
  - Configuration:
    - *pv* (process variable)
      - Select any point for the process variable or input.
    - *high*
      - Select any point for the high/upper boundary.
    - *low*
      - Select any point for the low boundary.
    - *incl bnd* (inclusive boundaries)
      - Set as True to include the configured boundary values during the in-range check or false to exclude them.
    - *invert*
      - Set as true to invert the output.
    - (output)
      - Select a Result bit to store the output.
  - Functionality:

        - When *incl bnd* is true, the non-inverted output will be True (1) when *PV* is both greater than or equal to *low*, and less than or equal to *high*, otherwise the output will be False (0).
        - When *incl bnd* is false, the non-inverted output will be True (1) when *PV* is both greater than *low*, and less than *high*, otherwise the output will be False (0).
- Hysteresis
    - Configuration:
        - *pv* (process variable)
            - Select any point for the process variable or input.
        - *high*
            - Select any point for the high/upper boundary.
        - *low*
            - Select any point for the low boundary.
        - *incl bnd* (inclusive boundaries)
            - Set as true to include the configured boundaries during the hysteresis check or false to exclude them.
        - (output)
            - Select a Result bit to store the output.
    - Functionality:
        - When *incl bnd* is true, the output will change to True (1) when the *PV* becomes greater than or equal to *high* and the output will change to False (0) when the *PV* becomes less than or equal to *low*.
        - When *incl bnd* is false, the output will change to True (1) when the *PV* becomes greater than *high* and the output will change to False (0) when the *PV* becomes less than *low*.

Start/Stop instruction

This instruction creates a common latch configuration that allows two momentary switches (a start button and a stop button) to turn on and off an output respectively with an optional emergency stop switch. The term latch refers to the fact that even after releasing the start button, the output will remain on until the stop button is pressed.

- Configuration:
    - *Start*
        - Select any point for the *Start* input. Typically, this is either a Binary input or a Result bit.
    - *Stop*
        - Select any point for the *Stop* input. Typically, this is either a Binary input or a Result bit.
    - *FStop* (force stop)
        - Select any point for the force stop or emergency stop input.

- (output)
    - Select any Result bit for the output.
- Functionality:
    - When *FStop* is False (0) the output will be False (0).
    - When *FStop* is True (1), the output will become True after the *Start* input is read as True (while this instruction is being processed) and remain True until the *Stop* input is read as True in which case the output will remain False.
    - To disable the force or emergency stop feature, Set *FStop* to Int.1.
    - If a single emergency stop button is wired to multiple machines, use the same *FStop* point for each Start/Stop instruction.

Limit instruction

This instruction will limit any input value between an upper and lower boundary by storing the limited value into a Result float.

- Configuration:
    - *pv* (process variable)
        - Select any point for the process variable input.
    - *min* (minimum)
        - Select any point to store the minimum allowed value.
    - *max* (maximum)
        - Select any point to store the maximum allowed value.
    - (output)
        - Select any Result float to store the limited output value.
- Functionality:
    - If *pv* is less than *min*, the output will be Set to *min*.
    - If *pv* is greater than *max*, the output will be Set to *max*.
    - If *pv* greater than or equal to *min* and less than or equal to *max*, the output will be Set to the present value of *pv*.

PID instruction (proportional, integral, derivative controller)

This instruction creates a feedback controller with proportional, integral and derivative terms and many additional features. A common example of a feedback controller is the cruise control system in a car. With constant power, a car ascending a hill would slow down however the cruise control system takes the current speed as the feedback, compares against the desired speed, and adjusts the power so that the car reaches the desired speed quickly and without overshooting.

- Configuration:
    - *pv* (process variable)
        1. Select any point for the process variable or feedback input.
    - *dt* (time derivative)

1. Select any Result float to store the amount of time that has passed since the instruction was last processed.

- *pGain* (proportional gain), *iGain* (integral gain), *dGain* (derivative gain), *min* (minimum), *max* (maximum), *rfPtr* (Result float pointer), *pidScale*, *oScale* (output scale), *offset*, *stPt* (setpoint)
    1. Select any 10 consecutive Internal floats (ADF) to store the above parameters, respectively.
    2. The System Design Studio will automatically adjust each input's channel appropriately as the other inputs change.
    3. *pGain* is the multiplier of the proportional term. A higher value will decrease rise time but increase overshoot.
    4. *iGain* is an optional multiplier of the integral term. A higher value will eliminate steady state error and decrease rise time however it will also increase overshoot, increase settling time and degrade stability.
    5. *dGain* is an optional multiplier of the derivative term. A higher value will decrease overshoot, and decrease settling time however the general stability will begin degrading if this value gets too big.
        - *It is strongly recommended to Set this gain to zero due to various complications.*
    6. *Min* is the minimum allowable value of the output. The output will never go below this value (prior to either scaling option).
    7. *Max* is the maximum allowable value of the output. The output will never go above this value (prior to either scaling option).
    8. ***rfPtr*** is an index to the first of 4 consecutive Result floats (If *rfPtr* is configured to ADF_6, then Setting ADF_6 with a present value of 1 will cause the instruction to use RES_FLT_1-4):
        - The 1st indexed Result float will store the accumulating integral term.
        - The 2nd indexed Result float will store the previous error.
        - The 3rd indexed Result float will be used as a variable setpoint when *fStPt* is set to Off. This is useful when using this instruction in conjunction with the PropCtrl instruction for reset control strategies.
        - The 4th indexed Result float will store the PID scaled output (the instructions main output multiplied by *pidScale*).
    9. *pidScale* is multiplied by the instructions output and the product is stored in the 4th indexed Result float. A value of zero will prevent the controller from Setting the 4th indexed Result float however it is still recommended to name and reserve the point in case it is needed in the future.
    10. *oScale* is multiplied by the instructions output and the product is stored as the new output. Note that the output will remain at zero if this point's present value is zero.
    11. *Offset* is added to the setpoint prior to calculating error regardless of whether *fStPt* is set to On or Off. This allows the default output to be configured when there is zero error. This is intended for Proportional-Only control.

12. *stPt* is used as the fixed setpoint when *fStPt* is set to On. This point will never be Set by the instruction when fStPt is set to Off however it is still recommended to name and reserve this point in case it is needed in the future.

- *fStPt* (fixed setpoint)
  1. Set to On for the instruction to use *stPt* as a fixed setpoint or set to Off for the instruction to use the 3rd indexed Result float (see rfPtr) as a variable setpoint.
- (output)
  1. Select any Result float to store the controller's output.

- Functionality:
  - This instruction is implemented as a reverse acting controller also referred to as an "Up-Up" controller. This means that when the controlled output goes up, the process variable also goes up.
    1. This instruction uses the parallel or ideal form with simple gains and each terms action is based on the error:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt} e(t)$$

    2. Note that manually changing a fixed setpoint can therefore cause an instant, possibly large change in output. It is recommended to implement a switch to disable the instruction according to the bumpless note below and use the switch to disable the instruction before changing the setpoint.
  - By default, the error is calculated as [setpoint – *pv]* and so a *pv* less than the setpoint will make the output positive and vice versa. To invert this logic, for a direct acting or "Up-Down" controller, use two "*" (multiplication) instructions to multiply the feedback input and setpoint input by -1 and use the resulting products as the new *pv* and setpoint, respectively.
    1. Regarding the note below on bumpless control, it is important to calculate (and invert if needed) the setpoint *after* the PID instruction so that the first PID calculation will have the setpoint equal to the process variable. This assumes the setpoint has been initialized or Set to the process variable (feedback input) at power up and any time the instruction is skipped.
  - There is a built-in integral windup prevention that will only accumulate the integral term when the output is within the *min* and *max* limits, or the output is at *max* but the integral term is decreasing, or the output is at *min* but the integral term is increasing. This prevents the integral term from increasing to very high or low numbers when the system is not connected or running that would otherwise cause significant overshoot (when first connecting or running the system).
    1. **Note that the min and max limits are enforced on the output prior to either scaling with pidScale or oScale.** Because of this, it may make sense to do any scaling in a separate multiplication instruction for increased visibility.

- **Note on bumpless control:** When a controller with a PID instruction first powers on, especially if this PID instruction will be skipped or disabled at times, there are a few things that should be done to prevent "bumps". The below 3 steps will make sure the PID instruction's first calculation will be equal to whatever value the output may be manually Set to. This will allow the PID instruction to ramp from the manually commanded output to the calculated output according to the gains selected by the user when the instruction is re-enabled. These calculations should be done at first power on, as well as any time the PID instruction is skipped:
    1. Clear (Set to zero) the previous error (2nd indexed Result float, see rfPtr).
    2. Continually Set the setpoint (whether using the fixed stPt or the variable 3rd indexed Result float) equal to the PV (process variable/feedback input). This will zero out the error and the derivative term due to an unknown dt (time since previous calculation).
    3. Set the accumulating integral term (1st indexed Result float) equal to the output divided by the integral gain (iGain). For example PIDInt = PIDOut / PIDKI .
- There is an example script located at C:\OpenBAS\Examples\14-2-PID-Bumpless.txt

PropStager instruction (proportional stager)

This instruction will turn on the required number of stages depending on the process variable input starting from the first stage of the current leader machine. This instruction supports up to 8 machines with 4 stages each for a total of 32 stages. This instruction has additional options for main stage enables, main stage hour counting, and main or sub-stage feedback monitoring and alarming.

- Configuration:
    - *run*
        - Select any point to control when this instruction should run. If the selected point's present value is zero, all output Result bits will be Set to zero.
    - *pv* (process variable)
        - Select any point to be used as the required number of running stages input. Often this is the output of a PID instruction and should be limited between zero and 32 depending on the number of stages to be controlled.
    - *timer*, *stg cnt* (stage count), *substg cnt* (sub-stage count), *stg ptr* (stage pointer), *stg ldr* (stage leader), *out ptr* (output pointer), *hc ptr* (hour counter pointer), *fb ptr* (feedback pointer), *alm ptr* (alarm pointer), *config* (configuration)
        - Select any 10 consecutive Internal bytes (ADB), Internal integers (ADI) or Internal floats (ADF) to store the above parameters, respectively.
        - The System Design Studio will automatically adjust each input appropriately as the other inputs change.
        - *timer* is an index to the timer used for both the minimum time before turning on any 2 stages and the time before alarm when using the feedback inputs.

- The timer instruction must be configured separately as either a 1s or 1/10s countdown timer.

- *stg cnt* should be set to the number of main stages or machines to be controlled and must be between 1 and 8 inclusively.
- *Substg cnt* should be set to the number of sub-stages or stages per machine and must be between 1 and 4 inclusively.
- *Stg ptr* is an index to consecutive Internal bytes (ADB) used for the main stage enables. When set to zero, no Internal bytes will be used.
  - The number of consecutive Internal bytes used is equal to the present value of *stg cnt*.
- *stg ldr* should be set to the index of the main stage or machine to be turned on first and must be between 1 and the present value of *stg cnt*. Often this will be the output of a LeaderCal instruction or Set manually.
- *out ptr* is an index to the first of consecutive Result bits that represent the state of each stage. Typically, these output Result bits will be directly or indirectly assigned to Binary outputs whose relays are physically wired to the individual stages.
- *hc ptr* is an index to the first of [*stg cnt*] consecutive Result bits that can be used as inputs to separate HourCnt instructions for each main stage.
  - See *config* below for information on how to enable.
- *fb ptr* is an index to the first of consecutive Result bits that can be used as stage status inputs for alarming purposes where True indicates a stage is running.
  - The number of Result bits used depends on *config* below and whether the feedback inputs are disabled, enabled for main stages, or enabled for all sub-stages and will use [zero], [*stg cnt*], or [*stg cnt* * *substg cnt*] Result bits, respectively.
- *alm ptr* is an index to the first of consecutive Result bits that can be used as stage alarm outputs where True indicates a stages feedback was False after the *timer* expired.
  - The number of Result bits used depends on *config* below and whether the alarm outputs are disabled, enabled for main stages, or enabled for all sub-stages and will use zero, [*stg cnt*], or [*stg cnt* * *substg cnt*] Result bits, respectively.
- *config* is used to select the required mode of this instruction:
  - Set to 0 to disable all three sets of hour counter outputs, feedback inputs, and alarm outputs.
  - Set to 1 to enable the main stage hour counter outputs only.
  - Set to 2 to enable main stage feedback inputs and main stage alarm outputs.
  - Set to 3 to enable the main stage hour counter outputs, main stage feedback inputs and main stage alarm outputs.
  - Set to 6 to enable feedback inputs and alarm outputs for all sub-stages.

- Set to 7 to enable feedback inputs and alarm outputs for all sub-stages as well as main stage hour counter outputs.

- *stg out#* (stage output number)
  - Select any Result float to store the current number of stages that have been commanded on.

- Functionality:
  - The present value of *leader* determines which machine's stages (I.e. which of the indexed Result bits from *out ptr*) will be commanded on first. For example, with *stg cnt* Set to 4, *substg cnt* Set to 4, *leader* Set to 4, and *out ptr* Set to 41 (for RES_BIT_41), RES_BIT_53-56 would be commanded on when 1-4 stages are needed respectively and RES_BIT_41-52 would be commanded on when 5-16 stages are needed, respectively.
    - Note that changing the leader will only affect the outputs when there is a change in the number of currently required stages.
  - When the *pv* becomes greater than or equal to 1, the first stage of the first machine (determined by *leader)* will turn on and will only turn off when pv drops below 0.5 (or the machine is disabled, or the *run* input is False). The second stage will be commanded on when the *pv* becomes greater than or equal to 2 and will not turn off until the *pv* drops below 1.5 and so on.
  - The *run* input like an emergency stop will allow the outputs to turn on when True but will instantly Turn all outputs off when False.
  - The main stage enable inputs can be used when *stg ptr* is not equal to zero. If a main stage enable input is zero when there is a change in the *pv,* then all sub-stages of the respective main stage will be commanded off and the next available stages will be commanded on as needed.
  - Each time a stage is commanded on, the *timer* will begin counting down and no more stages will be commanded on or off until the timer expires.
  - If feedback and alarm is enabled when the *timer* expires, each main or sub-stage (depending on the value of *config*) that is commanded on will have the appropriate feedback input checked. If the feedback input is False, then the respective alarm output will be True and vice-versa.
    - Alarms are non-latching meaning once the timer has expired, the alarm output will be True if the feedback is False but will still become False at any time if the feedback becomes True.
    - An alarm output will stay True when the respective output is turned off.
    - If the feedback is already True when the stage is first turned on, the alarm output will be True until the timer expires.
    - Alarm outputs can be paired with Start/Stop instructions to create latching alarms in order to help identify any stages that are cycling or taking too long to turn on.
  - When the main stage hour counter outputs are enabled, the respective Result bit will be True when at least one of a main stage's sub-stages have been commanded on (regardless

of the feedback and alarm points) and False when none of the sub-stages of the respective main stage have been commanded on.

- The hour counter outputs are typically paired with HourCnt instructions to accumulate the total runtime.
- The hour counter outputs can be used in an AND instruction with the feedback if required.

LeaderCal instruction (leader and lagger calculator)

This instruction, when processed, will consider up to 8 machines/stages and depending on the configuration, will output the index of the machine or stage with the least or greatest runtime of the ones that are currently enabled. After dragging the LeaderCal instruction into a PLC, hover over "Leader Cal" and a dropdown box will appear with an option for Lagger. Use Leader Cal to output the index of the machine with the least runtime and use Lagger to output the index of the machine with the greatest runtime.

- Configuration:
  - *stg cnt* (stage count)
    - Select up to 8 stages to compare runtimes for, the same way a point channel would be selected. For example, INT.10 would be interpreted as a point with a present value of 10.
  - *rt ptr* (runtime pointer)
    - Select the first of [*stg cnt*] consecutive points of any type whose present values represent runtimes. The runtimes should be in the same order as the enable flags.
  - *flg ptr* (enable flag pointer)
    - Select the first of [*stg cnt*] consecutive points of any type whose present values represent the enable status of each stage. The enable status points should be in the same order as the runtimes.
  - (output)
    - Select any non-binary writeable point (AO, ADB, ADI, ADF, RES_FLT, RMT) to store the index of the stage or machine with the least/greatest runtime, and that is enabled.
- Functionality:
  - When this instruction is processed, each enabled runtime will be compared, and the output point will be Set with the index of the enabled runtime with the lowest (leader) or greatest (lagger) value.
  - Often the enable status points are calculated using boolean instructions based on whether the stage is currently running or if the stage has any alarms for example.
  - Typically, this instruction is used with a schedule and a jump or Call instruction to only run this instruction when needed such as once per week. For example:
    - Create a schedule to Set RES_BIT_40 to 1, once a week.

        - Create an inverted jump instruction conditional on RES_BIT_40 to jump 2 instructions in PLC1, instruction 1.
        - Create the leader calculator instruction in PLC1, instruction 2.
        - Create an assign instruction to assign INT.0 to RES_BIT_40 in PLC1, instruction 3.
    - If all the stages are disabled, the output will not change.

Staggered Start/Stop instruction

This instruction allows for up to 40 stages to be turned on and/or off in a staggered fashion. This is commonly used when multiple stages or machines have significant start up power requirements and staggering each stage therefor minimizes the peak load. Significant peaks in power usage can cause many problems and some utility companies even bill based on the peak usage.

- Configuration:
    - *pv* (process variable)
        - Select any point whose present value will be used to initiate the starting or stopping of the configured stages.
    - *Timer, stg cnt* (stage count), *stg ldr* (stage leader), *enReg ptr* (enable register pointer), *mode*
        - Select any 5 consecutive Internal bytes (ADB), Internal integers (ADI) or Internal floats (ADF) to store the above parameters, respectively.
        - The System Design Studio will automatically adjust each input appropriately as the other inputs change.
        - *Timer* is an index to the timer used for the time between turning on or off any 2 consecutive stages.
            - The timer instruction must be configured separately as either a 1s or 1/10s countdown timer.
        - *Stg cnt* should be Set to the number of stages or machines to be turned on or off and must be between 1 and 40 inclusively.
        - *Stg ldr* should be Set to the index of the stage which should be turned on first and must be between 1 and [*stg cnt].*
        - *enReg ptr* is an index to the first of [*stg cnt*] consecutive Internal bytes (ADB) to be used as individual enables for each stage. Set *enReg ptr* to zero to disable.
        - *Mode* is used to configure the mode of operation:
            - Set to 1 for staggered start and simultaneous stop.
            - Set to 2 for simultaneous start and staggered stop.
            - Set to 3 for staggered start and staggered stop.
    - *stg ptr* (stage pointer)
        - Select the first of [*stg cnt*] consecutive Binary output, Result bit, Result float or Remote type points for the output stages.
    - *cur stg* (current stage)
        - Select any Result float to store the current number of stages that are commanded on.

- Functionality:
    - When configured for staggered start and *pv* becomes True, the configured stages will begin turning on, separated by the timer, starting from the leader, and incrementing upwards. For example, with *stg cnt* set to 8 and *stg ldr* set to 4, when *pv* becomes True, the 4th indexed output point will be commanded on first followed by the 5th, 6th, 7th, 8th, 1st, 2nd and finally the 3rd indexed output point.
    - When configured for staggered stop and *pv* becomes False, the configured stages will begin turning off, separated by the timer, in reverse order from how they were turned on (unless *stg ldr* changed).
    - When configured for simultaneous start, all outputs will turn on when *pv* becomes True and when configured for simultaneous stop, all outputs will turn off when *pv* becomes False.
    - When the ADB enables are configured, the stage outputs will be skipped during a start if their respective enables are False.
        - With 3 stages and only the 2nd stage disabled, the instruction will turn on the 3rd stage when the timer expires after turning on the 1st stage. In other words, the instruction does not waste time waiting for the timer to expire for stages that are disabled.
        - Stages can be disabled or re-enabled at any time. Note that enabling multiple stages after PV has already been Set to 1 will command the outputs on without regard for the timer.

Special instructions

The special instruction allows for new instructions to be used or tested without updating the System Design Studio (firmware will need to be updated). After dragging in a Special instruction, hover over "Instr:100" and a dropdown will appear allowing a number between 100 and 126 to be selected. As of firmware version 3.17.0, there are 3 special instructions available which are for calculating a random number (123), modulus (124), and square root (125). **The super heat (126) instruction was removed in firmware version 3.17.0.**

- General configuration:
    - *stpt ptr* (setpoint pointer)
        - Select the first of any type of [*stpt cnt*] consecutive points typically used for setpoints.
    - *stpt cnt* (setpoint count)
        - Select the number of consecutive points being used for setpoints starting from *stpt ptr*. For example, INT.10 would be interpreted as a point with a present value of 10.
    - *pv (*process variable*)*
        - Select any point typically used as the main input or process variable.
    - *(other input)*
        - Select any point as required by the specific Special instruction.
    - *(output)*

        - Select any writeable point (AI, BI not supported) to store the output.
- General functionality*:*
    - The functionality depends on the selected special instruction number.

Random number special instruction

This instruction can be selected using the Special instruction with *Instr:123* selected from the main dropdown. This instruction will generate a pseudo-random number between 0 (zero) and 1.98, depending on both time and the current number of instructions being processed per second.

- Configuration:
    - *stpt ptr* (setpoint pointer)
        - Select any point to pass the instruction validation, it will not be used.
    - *stpt cnt* (setpoint count)
        - Select any number, this will not be used.
    - *pv (*process variable*)*
        - Select any point to pass the instruction validation, it will not be used.
    - *(other input)*
        - Leave as null since this will not be used.
    - *(output)*
        - Select any available **RES_FLT** to store the pseudo-random number.
        - Note: do not use any Internal setpoints (ADB, ADI, ADF) unless this instruction is only being processed less than once every 5 minutes.
- Functionality*:*
    - The selected output Result float will be updated with a pseudo-random number between 0 (zero) and 1.98 every time this instruction is processed.

Modulus special instruction (division remainder)

This instruction can be selected using the Special instruction with *Instr:124* selected from the main dropdown. This instruction will output the remainder after a division.

- Configuration:
    - *stpt ptr* (setpoint pointer)
        - Select any point to pass the instruction validation, it will not be used.
    - *stpt cnt* (setpoint count)
        - Select any number, this will not be used.
    - *pv (*process variable*)*
        - Select any point to be used as the dividend.
    - *(other input)*
        - Select any point to be used as the divisor.
    - *(output)*
        - Select any Result float to store the modulus or remainder.

        - Note: do not use any Internal setpoints (ADB, ADI, ADF) unless this instruction is only being processed less than once every 5 minutes.
- Functionality*:*
    - Both the dividend and the divisor will be rounded down to the nearest integer before being divided and then the selected output point will be Set with the Modulus or remainder each time this instruction is processed.

Square root special instruction

This instruction can be selected using the Special instruction with *Instr:125* selected from the main dropdown. This instruction will output the square root of the input or process variable.

- Configuration:
    - *stpt ptr* (setpoint pointer)
        - Select any point to pass the instruction validation, it will not be used.
    - *stpt cnt* (setpoint count)
        - Select any number, this will not be used.
    - *pv (*process variable*)*
        - Select any point to calculate the square root of.
    - *(other input)*
        - Leave as null since this point will not be used.
    - *(output)*
        - Select any Result float to store the square root.
        - Note: do not use any Internal setpoints (ADB, ADI, ADF) unless this instruction is only being processed less than once every 5 minutes.
- Functionality*:*
    - When this instruction is processed the square root of the selected process variable will be calculated using the Babylonian method with up to 15 iterations.

Superheat special instruction (for R22 refrigerant)

No longer available as of firmware 3.17.0. Do not upgrade if using.

This instruction can be selected using the Special instruction with *Instr:126* selected from the main dropdown. This instruction will calculate the saturation (boiling) temperature in Fahrenheit of R22 refrigerant measured at a certain pressure in PSIG.

- Configuration:
    - *stpt ptr* (setpoint pointer)
        - Select any point to pass the instruction validation, it will not be used.
    - *stpt cnt* (setpoint count)
        - Select any number, this will not be used.
    - *pv (*process variable*)*
        - Select any point to pass the instruction validation, it will not be used.
    - *(other input)*

        - Select any point to be used as the measured pressure in PSIG or pounds per square inch gauge (with respect to atmospheric pressure).
    - *(output)*
        - Select any Result float to store the saturation temperature in degrees Fahrenheit.
        - Note: do not use any Internal setpoints (ADB, ADI, ADF) unless this instruction is only being processed less than once every 5 minutes.
- Functionality*:*
    - This instruction will determine the saturation (boiling) temperature within a range of -40 to +79 degrees Fahrenheit of a measured pressure in PSIG for systems using R22 refrigerant.
    - This instruction is typically used with a "-" (subtraction) instruction to subtract the calculated saturation temperature (based on the pressure of the suction line) from the measured suction line temperature, to calculate superheat.
    - Alternatively, this instruction can be used with a "-" (subtraction) instruction to subtract the measured liquid line temperature from the calculated saturation temperature (based on the pressure of the liquid line), to calculate subcooling.