Mircom

OpenBAS BUILDING AUTOMATION SYSTEM

System Design Studio User Guide

LT-6675 Rev. 1
April 2020

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

About this guide

This guide covers the System Design Studio version 1.0.2 with firmware versions of 3.04.0 and greater. **Note**: a firmware version of 3.04.0 or greater is required. 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.

Specifications

- Supported OS
  - Windows 7, Windows 10
- Supported architecture
  - 32 bit, 64 bit
- Supported controllers
  - All NX core-based controllers (OpenBAS-XX-**NX**XXX, OpenBAS-HV-LEARN)

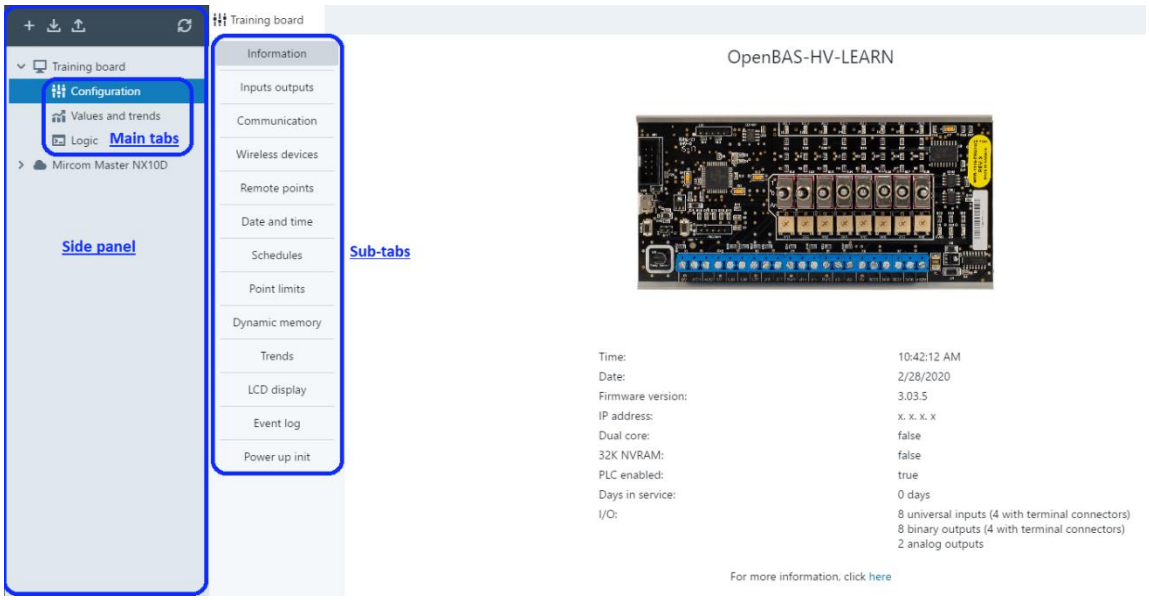
Mircom

Table of Contents

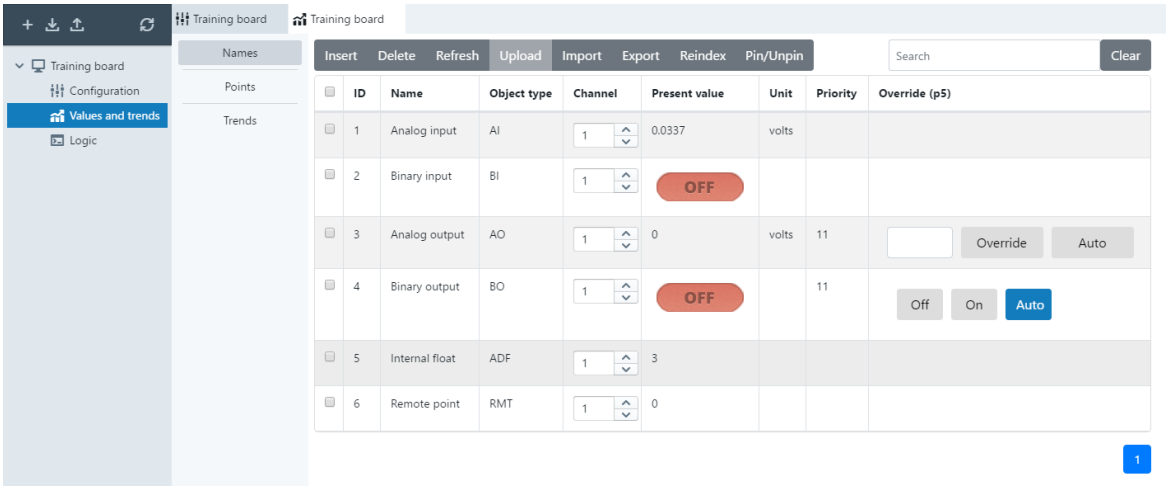
User interface overview

The user interface can be divided into three main sections including the side panel, main tabs and sub-tabs. 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.

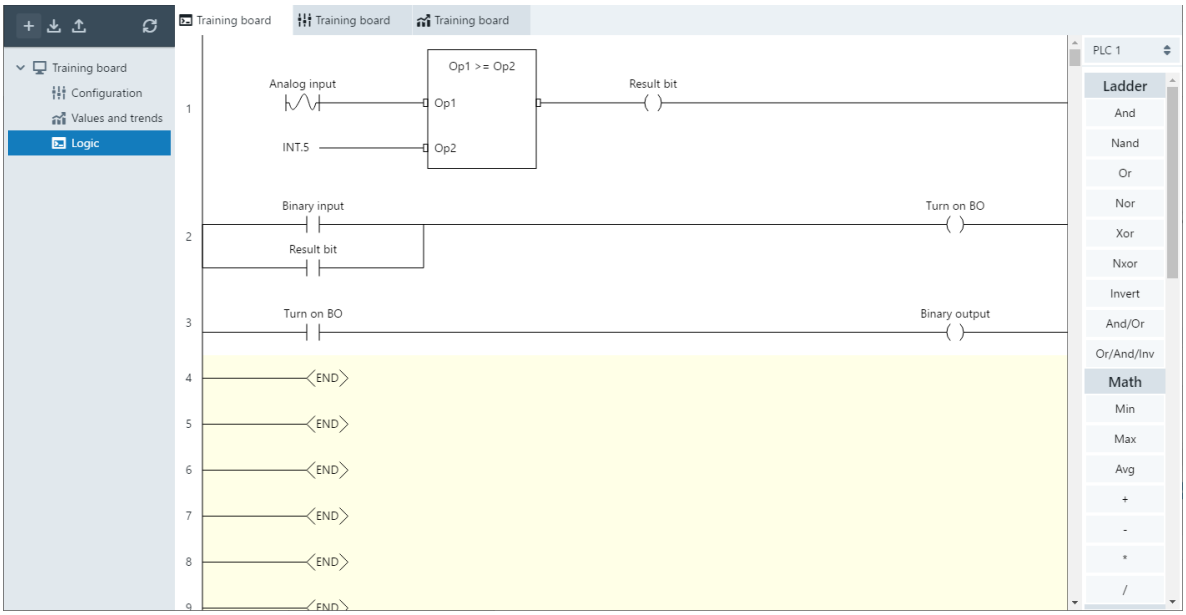
User interface and Configuration main tab



Values and trends main tab



Logic main tab



Connecting to controllers

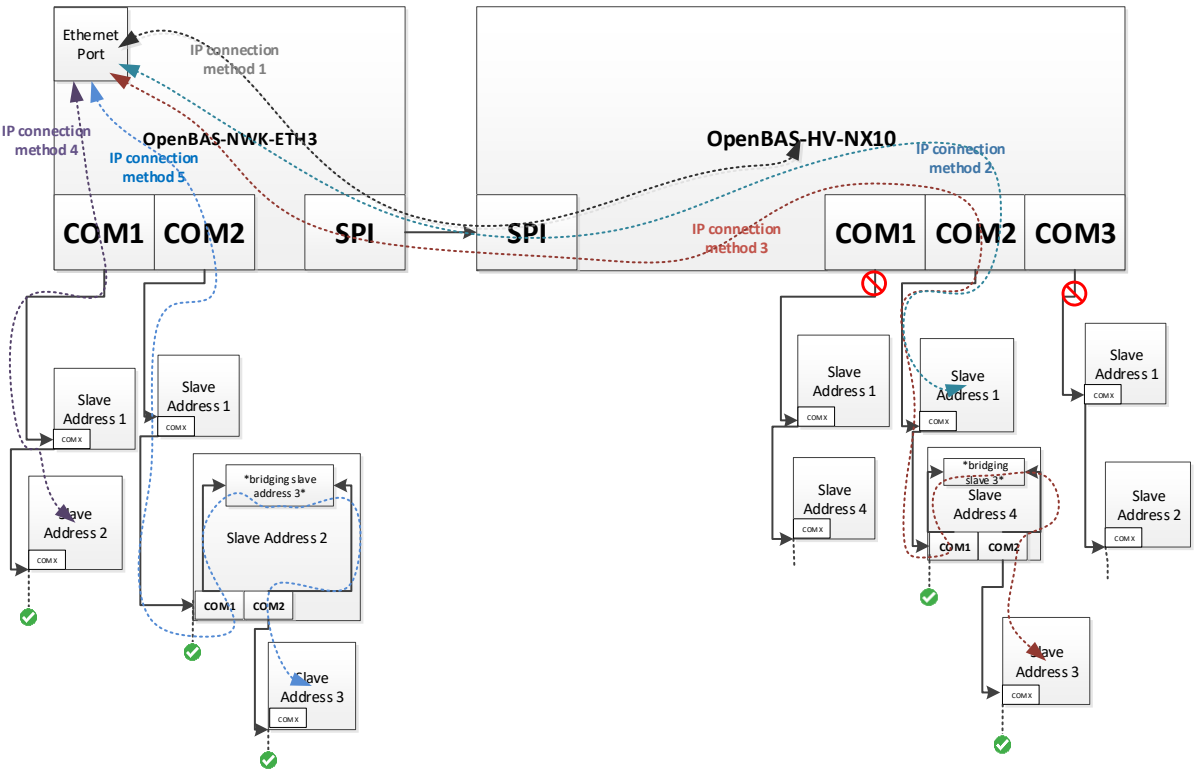
The System Design Studio can connect and communicate with controllers in many ways. In order to understand the different methods of connecting to a controller, a few terms and concepts need to be reviewed. OpenBAS controllers are commonly configured to communicate with each other using the Optomux or Modbus 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 communication methods involve using the master controller to relay messages to one of the slave controllers, as well as relay that slave controller's responses back to the System Design Studio.

- General requirements
    - 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 needs to be connected to the “middle” controller’s COM1 and the middle controllers COM2 needs to be connected to any COM port on the “slave-of-slave” controller.

- Connecting to controllers over IP:
  - Requirements:
    1. PC must be on same network as controller.
    2. The IP address of the respective OpenBAS-NWK-ETH3 is needed.
       1. See LT-6630 for help determining the OpenBAS-NWK-ETH3’s IP address.

System Design Studio IP Connection Options



  - 5 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.

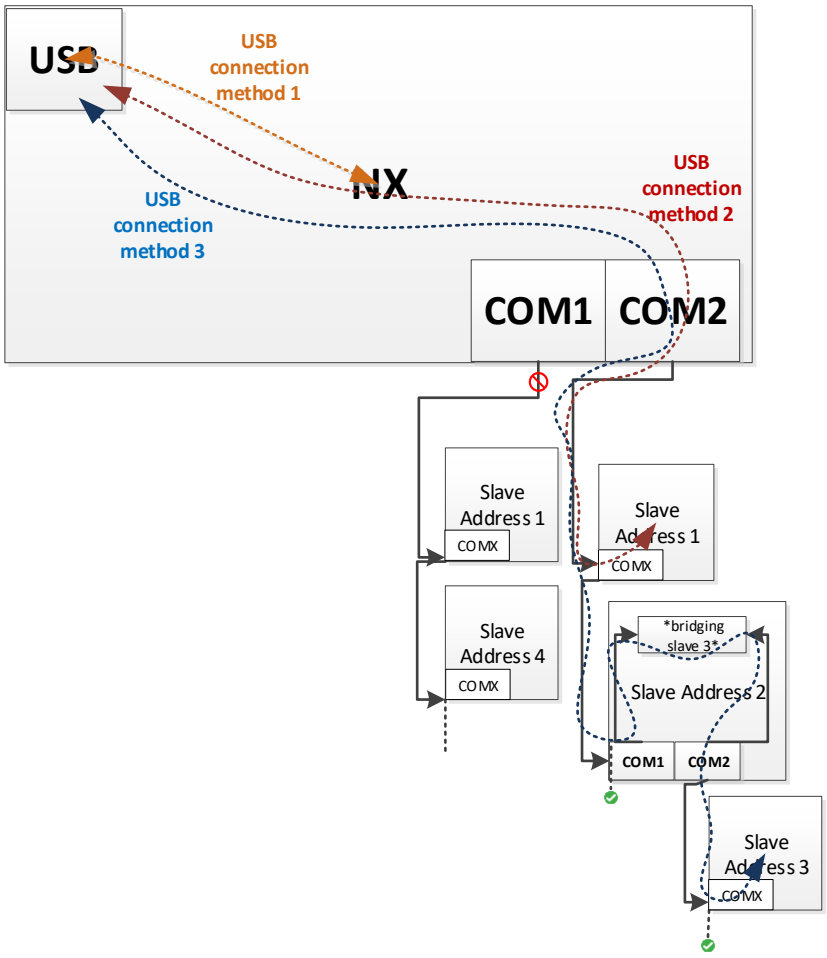
1. Click the “+” icon in the top left-hand corner, fill in the IP address and click “Add”. The IP address should be in the form of “123.123.123.123:1234” or “www.sampleIP.com:8080”

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 option 1.
    2. Right click the master controller item in the side panel and click “Add slave”.
    3. Select SPI 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 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 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 COM1 or 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 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 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 COM1 or COM2 appropriately from the channel drop down box.

4. Type in the address of the “slave-of-slave” controllers respective COM port and click “Ok”.

- Connecting to controllers over USB
  - 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.

System Design Studio USB Connection Options



  - 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 COM2 is selected from the channel drop down box.
    5. Type in the address of the slave controllers respective COM port and click “Ok”.
3. [1] **USB connection method 3:** Connect to a slave on **COM2** of a slave 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 COM2 is selected from the channel drop down box.
    5. Type in the address of the “slave-of-slave” controllers respective COM port and click “Ok”.

[1] “Slave-of-slave” connections require a minimum baud rate of 19200 on both involved communication busses.

[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 in order to establish a connection. Alternatively, the COM port can be disabled and re-enabled within Windows Device Manager.

Removing and disconnecting controllers

Right click a controller item in the side panel and click “Remove”.

- IP connections will just no longer appear in the side panel.
- USB connections will release or close the COM port in Windows in addition to no longer appearing in the side panel.

Naming controllers

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

- Controller names are limited to 18 characters.
- The controller name is stored in the 200th index/ID of the Point names list within the controller's memory. However, the controller name is hidden from the **Values and trends->Names** table. Knowing this allows the controller name to be displayed on the LCD display (See LCD display, alarms and configuration).

Importing and exporting IP connection lists

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".

- The controller 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. 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 a 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 (which can be managed by different COM ports) 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 communication method 1 in order 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:
    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:
    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.

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 core" 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 standard "NX Core" 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 core" database with the OpenBAS-ACC-32KNV memory expansion (Changes from NX core 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 core 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 **are** available
Remote points (RMT)	1..50 **+51-255**
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. 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, or Trends 1-8 can be traded for +8KB of script/file storage.
    - Section two, or Trends 9-12 can be traded for +4KB of script/file storage or PLC3.
    - Section three, or Trends 13-16 can be traded for +4KB of script/file storage or PLC2.
- Controllers with the OpenBAS-ACC-32KNV memory expansion (includes 16 KB of script/file storage by default).
    - Section one, or Trends 1-8 cannot be traded.
    - Section two, or Trends 9-12 can be traded for +4KB of script/file storage.
    - Section three, or Trends 13-16 can be traded for +4KB of script/file storage.
- OpenBAS-HV-NX10D (includes 16 KB of script/file storage by default).
    - Section one, or Trends 1-8 can be traded for +32KB of script/file storage.
    - Section two, or Trends 9-12 can be traded for +32KB of script/file storage.
    - Section three, or 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.
    - 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
    - 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
  - 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. 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 400 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. There is no need (or option) to specify in software that a universal input is being used as either a binary or analog input, just calibrate and name the respective point. For example, when there is a voltage of 3V DC on universal input 1, analog input 1's (AI_1) present value (if calibrated as a 0-10V 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).

- Go to **Configuration->Inputs Outputs** and select the **Universal input** option from the main dropdown box.
    - Under the **Analog input calibration** header, 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 a user to better match the real time value of a multimeter or other measuring device.
    - For noisy or unstable inputs, under the **Universal 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 in **Configuration->Inputs outputs->General** which affects all 40 analog inputs. 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.
    - The "Invert local binary input logic" option will change binary input 1 through 8 to use "active-low" logic which is useful when all of 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).
    - "OpenBAS-HV-NXSF CT input noise level" within *Configuration->Inputs outputs->General*, 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. After installation, configure all AI's that are connected to CTs as "12-bit ADC" type inputs and find the maximum of their present values, while the measured circuits are off. Use this maximum value as the "OpenBAS-HV-NXSF CT input noise level" and then don't forget to configure each Analog input back to "Current Transformer (1000:1)" type inputs.

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

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 within the controller that they physically exist.
    - Set each points’ 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 device, the local Analog outputs can be configured with a default present value in case 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 in the “Local output state when offline” section in **Configuration->Inputs outputs** (select General from the main drop down box).
- 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 200 ms 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). Similar to “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 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” 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.

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 load 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, type or select a

Name, Object type and Channel, and then click Save. 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 clicking on their current name under the “Name” column and clicking outside of the text box or pressing enter.
- 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 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 a script or clicking upload.
- See “Debugging and testing logic” for more information on the “Pin/Unpin” button.

Limiting point values

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're 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 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".
    - Limit the number of controllers on any bus to a maximum of 32. Using the above poll-rate assumption of 25 points per second and an average of 20 points per controller, there would be 640 points in total which would take around 26 seconds for each polling cycle. Also, separating large field buses into smaller field buses decreases risk since less controllers will go offline should any single controller or bus have an issue.
    - Use one of the following baud rates depending on the protocol:

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

- 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 of these settings must match (aside from Address, which must be unique) for every controller on a fieldbus.
        - Note that the address of a COM port that is configured with a Master type protocol is ignored.

- "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.
- 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 block" 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 block" setting). Any "assign" instructions located **after** the Xth 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.
    - The "BACnet MSTP" protocol is only available on COM1 and has some additional settings that appear when selected.
        - "Max master" is a setting that must 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.
            - Typically, this should be set to the number of controllers on a bus + 5. Adding 5 will slow down communications slightly however it will save a lot of time and labour if more devices ever need to be added to a bus since this setting must be configured on each device.
        - "Discovery" allows you to select the range of each point type that will be discoverable over BACnet.
            - 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 a script or go to *Values and trends->Names* and click the "Reindex" button to manually reindex them.
        - "NPDU" can be set to local when connecting to an NAE/NCE for improved communications.
        - "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.
        - Note that when COM1 is configured for the BACnet MSTP protocol, all 16 BACnet priorities become available for Binary Outputs however they are all mapped to the four OpenBAS priorities (BACnet priority level 1,5,9 and 11). Changing the protocol on COM1 to BACnet MSTP requires the controller to be restarted.
    - 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.
        - "Modbus master write coil function" will either use the "write single coil" or "write multiple coils" Modbus function/message when writing to a coil (binary input/binary output/result bit).
        - "Modbus slave write single coil action" will treat received "write single coil" functions/messages as either regular Set or Override messages.
        - "Modbus slave write multiple coils action" will treat "write multiple coils" functions/messages as either regular Set or Override messages.

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 and Set/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.
    - 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 with the existing pinned points.
- Use the buttons in the Override column to command output points in order to test logic as needed.

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 important to perform a factory reset after following the proper connection procedure.

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 core-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**.

- 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).
- 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.
- Alarm messages are limited to 16 characters.
- "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.
- "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).
- "Extra byte to read" is a legacy feature that is no longer supported.
- "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.
- 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.
    - 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).
    - Analog and Binary inputs are read only and cannot be Set or Overridden by the user. For testing and debugging purposes, modify the input calibration to achieve the required input value.
- 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 take any decimal value in the range -99999.9999 to 99999.9999.
  - The value of each ADF point will remain even if power is disconnected or lost.
  - Warning: ADF points are stored in eeprom 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 any 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 eeprom 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 state 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 eeprom 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 state 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 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 (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_40-246 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_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 state 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 core-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)
    - View/set the present value and point names of the first 40* result float type points in the point database.
        - *Expanded result float points are typically used as remote points and are not available in the RES_FLT table (see 'Configuring remote points').
            - Expanded RES_FLT_**41**-255 are available on the OpenBAS-HV-NX10D.
            - Expanded RES_FLT_**51**-255 are available on any NX core-based controller when paired with the OpenBAS-ACC-32KNV or similar memory expansion.
    - 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.
    - Result floats (1-40) are typically used to store the result of a PLC instruction and can take any value between -99999.9999 and 99999.9999.
    - Unlike the eeprom-based internal bytes, integers and floats (ADB, ADI, ADF), RES_FLT_1-40 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) to make sure that remote points that are configured to Set eeprom-based points such as ADB, ADI or ADFs are not written too frequently as this can permanently damage the controllers 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 (1-40) 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 core-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 NX core-based controller.
  - An undefined mode indicates that the timer has not been configured in the PLC.
  - TMR points are read only and cannot be Set or overridden 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)
  - View the present value and point names of the first 50* RMT type remote points.
    - *The present value of all remote points, including 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 core-based controller paired with the OpenBAS-ACC-32KNV or similar memory expansion.
  - Remote points need to be configured for their present value to be copied from a point on another controller through a fieldbus (see Configuring remote points).
  - The present value of RMT_1-50 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 written too frequently as this can permanently damage the controllers 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 remote points

Remote points can be configured from within the **Configuration->Remote points** tab. Depending on the controller and memory expansions, there can be two types of remote points available which are RMT_1-255 points, and RES_FLT_41-255 (*RES_FLT_1-40 are PLC result registers only and cannot be configured as remote points). 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 all RMT type 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 in order for the remote points to show an online status. In addition, it's important to consider the remote point's managing COM port as well as the Last point to poll setting.

- Use the main drop-down box to switch between viewing RMT and RES_FLT type remote points when available.
- Enter and select the device Address, Object type and Channel to configure a remote point.
- Since many controllers have more than 1 fieldbus or COM port, it is necessary to identify 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.
    - 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.
- 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 in order to take advantage of all 40 UI, 40 BO and 10 AO available in each 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 must first be configured in the Configuration->Wireless devices tab (see Connecting wireless devices).
    - Wireless type remote points are only supported on COM2 (see 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.
- RMT_1 to RMT_10 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: 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.
- Offline points
    - The controller will attempt to poll all configured remote points whether they are online or offline.
    - The controller will wait up to 200 ms 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.

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 core-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. 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’s important to describe exactly what’s included 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.
- 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 that the backup (including Logic, Configuration or both) was created from must match the database size 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 core-based device regardless of whether it's an OpenBAS-HV-NX10D or has a memory expansion accessory.

Configure from script (EZ-Script)

To compile a script and load the compiled configuration to a controller, right click a controller item from the side panel and select "**Configure from script**". Browse to and select a project directory containing at least one script file. By default, Quick configuration will be selected which 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 configure Logic, Setpoints, Configuration or some combination of all 3 options. After clicking Configure, a confirmation dialog will appear. To configure the controller, click Start and a loading bar will appear followed by a notification saying the configuration from script was successful. If there is a compiler error, multiple text files will open that outline the error details.

- Project directories are identified by, and must include, a project.nx5 file. This file is created automatically when clicking the Edit button (requires Notepad++). If needed, manually create a project.nx5 file by creating and renaming a basic *.txt file to "project.nx5". Make sure that you

set Windows Explorer to show file 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 files will be created automatically if they do not already exist when clicking the Edit button (requires Notepad++). When manually creating script files, the file names must be “script_1.txt” through “script_10.txt” and be used consecutively as needed. The definition file, if used, must be named “script_def.txt”.
    - No linking of any kind is needed, the compiler will begin by joining all files together (in order) before proceeding.
- When using Notepad++, a custom language file will be installed to allow for syntax highlighting. With an EZ-Script file open in Notepad++, click Language->OpenBAS EZ-Script to enable syntax highlighting for that file.
- To change the EZ-script language from English, the first line of script_def.txt (if used) or script_1.txt must include only the appropriate language keyword.
- Note: There is a known bug where any compiler error pop ups may open “behind” the System Design Studio window, with the loading bar not progressing. If the loading bar does not progress, be sure to minimize the System Design Studio to see if there is an error pop up “behind” it.
- See LT-6631 – EZ-script manual for more information on EZ script programming.

File transfer

Depending on the controller, memory accessories and Dynamic memory settings, files up to 112 KB in size can be stored in the memory of the controller. This can be useful for storing the latest copy of a script for example.

To upload a file to the 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.

1. Under Select file(s), select all files to be compressed into a zip file (and later stored on the controller).
2. Under Save as: (in the Upload section) browse to an appropriate location, enter a filename and click Save.
3. Click the Upload button to begin transferring the compressed *.zip file to the controller.
    a. 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.

1. Under Save as: (in the Download section), browse to an appropriate location, enter a file name and click Save.

a. 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:”.

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 Sub (Subroutine) instruction will return to the next consecutive instruction after the originating Sub instruction (see the Sub 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, 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 to the controller. To revert any changes, click the X icon.
- When configuring a controller from EZ-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.

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 it's configured inputs.

- Make sure to name all configured points in order to avoid using them again accidentally.

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 are True. If all configured inputs are False, the output will be False.
  - Null or unconfigured inputs are treated as False for this instruction.

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 2nd ,3rd and 4th 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 Sub (Call) instructions.

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

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).

- 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 (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.
    - *Itvl* (interval)
      - Select a decrement interval of either 100 ms 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).

- 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).
    - *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.

Sub instruction (subroutine call)

Sub instructions can save space (PLC instructions) and simplify logic when requiring the same functionality more than once in any program. A subroutine allows a consecutive set of instructions to be called and processed whenever needed.

- Configuration:
  - Select up to one input point of any type.
  - Click the gate to toggle whether the subroutine call is conditional on the input point being True or False.
  - Enter the 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 but within the same PLC as the Sub call.
- Must end in an END instruction.

- Functionality:
  - When the gate is not inverted, the controller will jump to the configured instruction 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 instruction 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 Sub 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 Sub instruction.
  - A null or unconfigured input will always jump to the configured instruction number regardless of whether the gate is inverted or not.

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 Sub instruction call. In this scenario, the controller will begin processing the next consecutive instruction following the originating Sub call.

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 commonly 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. 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 requiring controller replacement.

- Configuration:
  - 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.

- 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 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 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 0 (zero) and 100 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 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 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.
            - *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.
    - 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 (ie. 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 (ie. 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)
    - Select any point for the process variable or feedback input.

    dt (time derivative)

- 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)
  - Select any 10 consecutive Internal floats (ADF) to store the above parameters respectively.
  - The System Design Studio will automatically adjust each input appropriately as the other inputs change.
  - *pGain* is the multiplier of the proportional term. A higher value will decrease rise time but increase overshoot.
  - *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.
  - *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.
  - *Min* is the minimum allowable value of the output. The output will never go below this value (prior to either scaling option).
  - *Max* is the maximum allowable value of the output. The output will never go above this value (prior to either scaling option).
  - ***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*).
  - *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.
  - *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.
  - *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.

        - *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)
        - 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)
        - Select any Result float to store the controller's output.
- Functionality:
    - 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, 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.
    - 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).
    - If this PID instruction will be skipped or disabled at times, it is recommended to clear (Set to zero) the accumulated integral term and the previous error (1st and 2nd indexed Result floats, see rfPtr).

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 (ie. 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 similar to 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 calculator)

This instruction when processed will output the index of the machine or stage with the least runtime that is currently enabled.

- Configuration:
    - *stg cnt* (stage count)
        - Select the number of 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 enabled 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 runtime, and that is enabled.
- Functionality:
    - When this instruction is processes, each enabled runtime will be compared, and the output point will be Set with the index of the enabled runtime with the lowest value.
    - Typically, this instruction is used with a schedule and a jump or sub instruction in order to only run this instruction when needed. For example:
        - Create a schedule to Set RES_BIT_40 to 1, once 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 assignment 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. By default, there are 4 special instructions available which are for calculating a random number (123), modulus (124), square root (125) or super heat (126).

- 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)

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.