BACNET MS/TP INTEGRATION KIT FOR PAC PROJECT GUIDE

Form 1907-150316—March 2015

OPTO 22
Automation made simple.

43044 Business Park Drive • Temecula • CA 92590-3614
Phone: 800-321-OPTO (6786) or 951-695-3000
Fax: 800-832-OPTO (6786) or 951-695-2712
www.opto22.com

Product Support Services
800-TEK-OPTO (835-6786) or 951-695-3080
Fax: 951-695-3017
Email: support@opto22.com
Web: support.opto22.com

BACnet MS/TP Integration Kit for PAC Project Guide
Form 1907-150316—March 2015

The information in this manual has been checked carefully and is believed to be accurate; however, Opto 22 assumes no responsibility for possible inaccuracies or omissions. Specifications are subject to change without notice.

Opto 22 warrants all of its products to be free from defects in material or workmanship for 30 months from the manufacturing date code. This warranty is limited to the original cost of the unit only and does not cover installation, labor, or any other contingent costs. Opto 22 I/O modules and solid-state relays with date codes of 1/96 or newer are guaranteed for life. This lifetime warranty excludes reed relay, SNAP serial communication modules, SNAP PID modules, and modules that contain mechanical contacts or switches. Opto 22 does not warrant any product, components, or parts not manufactured by Opto 22; for these items, the warranty from the original manufacturer applies. These products include, but are not limited to, OptoTerminal-G70, OptoTerminal-G75, and Sony Ericsson GT-48; see the product data sheet for specific warranty information. Refer to Opto 22 form number 1042 for complete warranty information.

Opto 22
Automation Made Simple.

Table of Contents

OPTO 22

1: Using the Integration Kit

NOTE: To read the PIC statement for this integration kit, see "A: BACnet PIC Statement" on page 55.

Introduction

The BACnet MS/TP Integration Kit for PAC Project™ (Part # PAC-INT-BAC) enables your Opto 22 PAC system equipped with a SNAP-PAC-S1 or SNAP-PAC-S2 controller running a standard PAC Control strategy to communicate with a BACnet MS/TP network. BACnet is a communications protocol for building automation and control networks. MS/TP is a Master-Slave/Token-Passing specification of BACnet.

NOTE: In order for this integration kit to perform as intended, the S-series controller must be dedicated to BACnet only and the S-series controller and PAC Control strategy must ***not*** *be configured with the redundancy option.*

The integration kit contains the BACnet_Protocol and Read_VAV charts, which contain everything you need to use the BACnet MS/TP protocol in your own PAC Control strategy.

The BACnet Integration Kit meets the BACnet protocol standard 135-2008 version 1 revision 9. It was tested using PolarSoft BACbeat Evaluation/Analysis Tool v1.94.

This guide assumes that you understand how to use PAC Control™ and the BACnet MS/TP protocol, and how to use and configure an S-series controller.

What is Required

You will need the following things:

- A PC with PAC Project 9.3c or newer (Basic or Pro).
- An S-series controller connected to a BACnet MS/TP network

  BACnet MS/TP uses the EIA-485 physical layer. Connection to the BACnet MS/TP network can be made using an S-series controller's port configured as RS-485. For more information, see the following Opto 22 guides:

  – Form 1704, the *PAC Manager User's Guide*

  – Form 1592, the *SNAP PAC S-series User's Guide*

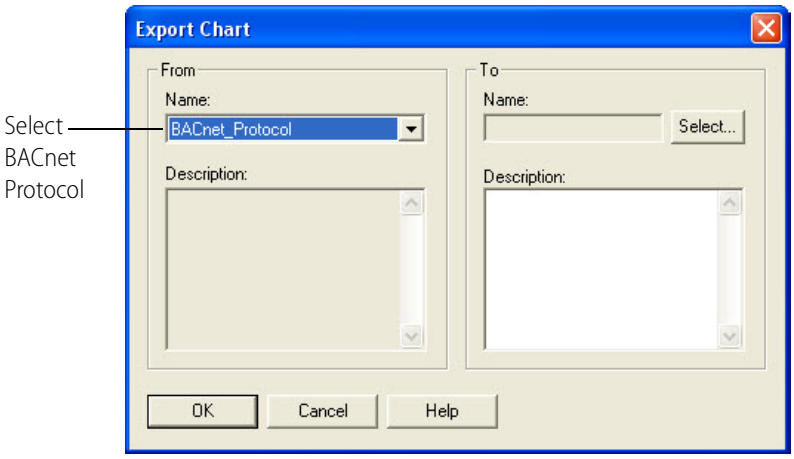
  – Form 1700, the *PAC Control User's Guide*

Exporting and Importing the Charts

In order to use the BACnet_Protocol, Read_VAV charts, and BACnet_Process_ComplexACK, you must first export these charts from the example strategy, and then import them into your own strategy.

Exporting the Charts

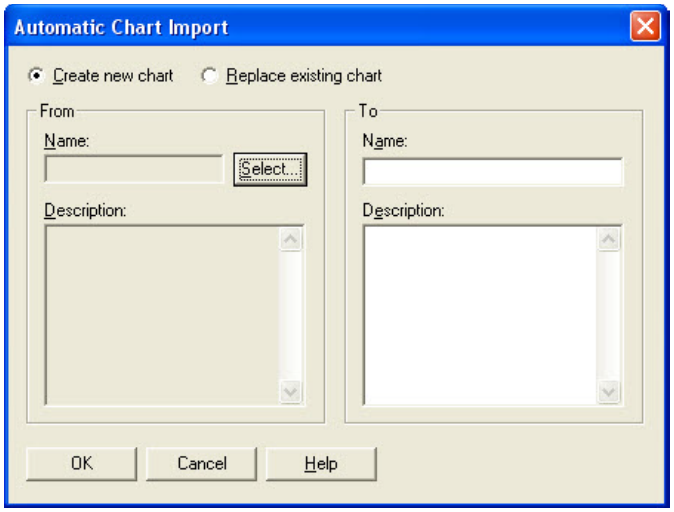
1. Open the zip file, and extract the contents of the zip file to a directory on your hard drive.
2. Start PAC Control, and open the strategy file you just extracted to your hard drive, BACnetInt.idb.
3. Select Chart > Export to open the Export Chart dialog box.



4. In the From combo box, select BACnet_Protocol.
5. Under To, click Select to open the Select Destination dialog box, and then browse to an appropriate directory such as your strategy's directory.
6. Name the export file *BACnet_export*, and then click Save to close the Select Destination dialog box.
7. Click OK to export the file and exit the Export Chart dialog box.
8. Repeat steps 3-7 to export the Read_VAV chart. Name the export file *Read_VAV_export*.
9. Repeat steps 3-7 to export the BACnet_Process_ComplexACK chart. Name the export file *BACnet_Process_ComplexACK_export*.

Importing the Charts

1. Open the strategy you want to use with the BACnet protocol.
2. Select Chart > Import to open the Automatic Chart Import dialog box.



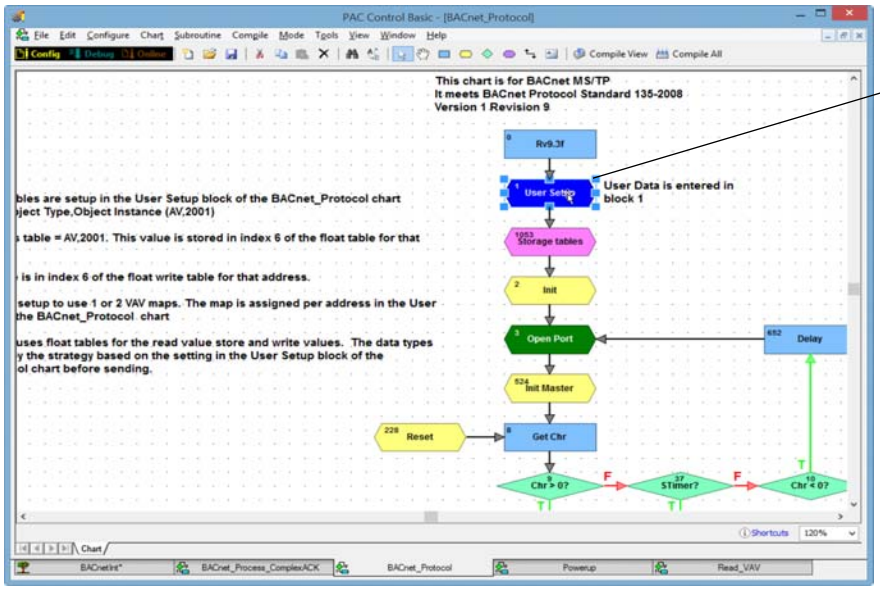
3. With "Create new chart" selected, click Select to open the Select File to Import dialog box.
4. Browse to the directory that contains the BACnet_export file, BACnet_export.cxf.
5. Select the export file, and then click Open.

   The Select File to Import dialog box closes.
6. Under To in the Automatic Chart Import dialog box, enter the name of the chart, *BACnet_Protocol*.
7. Click OK.
8. Repeat steps 2-7 to import the Read_VAV chart. Name the chart *Read_VAV*.
9. Repeat steps 2-7 to import the BACnet_Process_ComplexACK _export.cxf file. Name the chart *BACnet_Process_ComplexACK*.

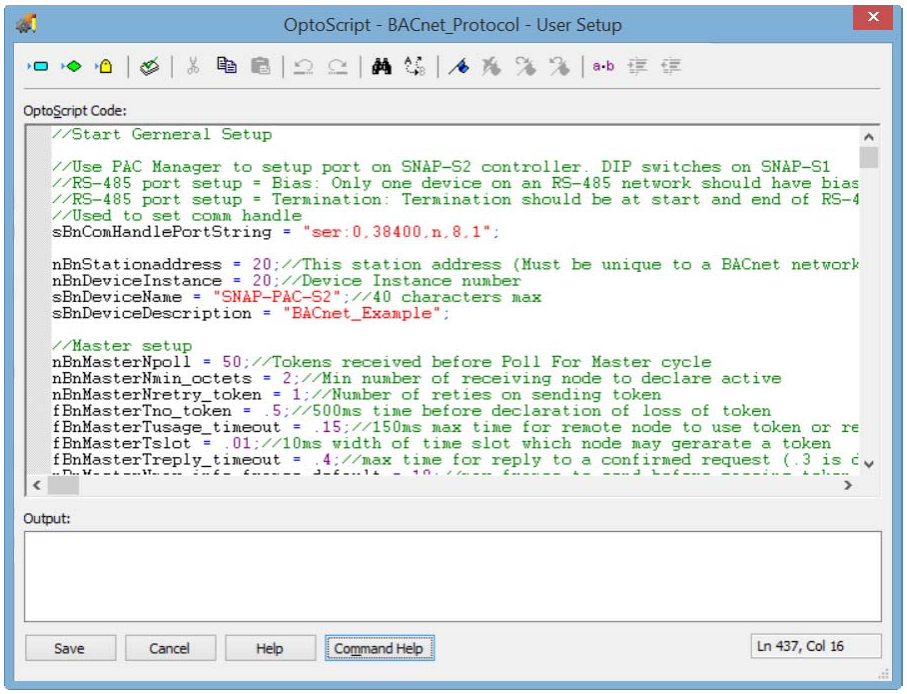
Entering User Setup Parameters

The user setup parameters are entered in Block 1 of the BACnet_Protocol chart.

1. With your strategy (BACnetInt.idb) open in PAC Control, open the BACnet_Protocol chart.
2. Double-click on Block 1, User Setup.



This opens the User Setup script.



3. In the User Setup script, enter the information for your device for each parameter group as described in the following sections:

- "Comm Handle, Station & Device, Master Setup, and Password" on page 5
- "Analog Input" on page 6
- "Analog Output" on page 7
- "Analog Values" on page 8

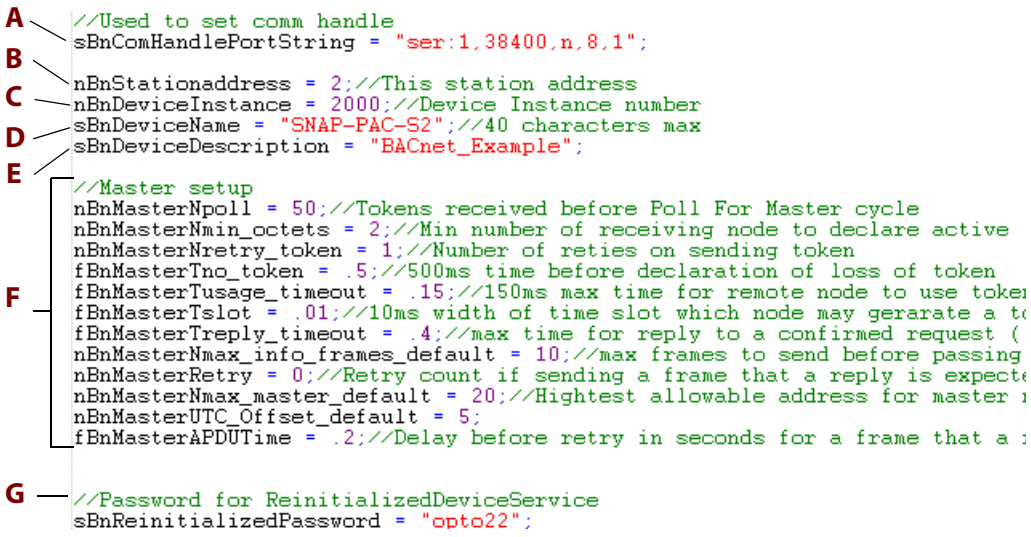
Opto 22 Table Index-to-Object Identifier Offset Defaults

The default values for the index-to-object identifier offsets are as follows:

Object	Offset
Analog Input	0
Analog Output	200
Analog Values	400

Object	Offset
Binary Input	600
Binary Output	800
Binary Value	1000

Comm Handle, Station & Device, Master Setup, and Password



Enter the following information:

A—Comm Handle string

B—Station address

C—Device instance

D—Device name

E—Device description

F—Master setup parameters

G—Password for Reinitialized Device Service

Analog Input

```
//Load analog input points into pointer table that will be used by BACnet.
//Index  to object identifier offset = 0
//Table = potBnAnalogInputs
//Load analog input names used by BACnet
//Table = stBnAnalogInputsObjectName
//Load analog input Engineering units used by BACnet
//Table = ntBnAnalogInputsEngUnits
//Load analog input Event State used by BACnet
//Table = ntBnAnalogInputsEventState
nBnAnalogInputIndexOffset = 0;
potBnAnalogInputs[0] = &aiRoom_Temp;
potBnAnalogInputs[1] = &aiPID_Input;
potBnAnalogInputs[2] = &aiPOT;
stBnAnalogInputsObjectName[0] = "aiRoom_Temp";
stBnAnalogInputsObjectName[1] = "aiPID_Input";
stBnAnalogInputsObjectName[2] = "aiPOT";
ntBnAnalogInputsEngUnits[0] = 64;//degrees-Fahrenheit
ntBnAnalogInputsEngUnits[1] = 56;//pounds-force-per-square-inch
ntBnAnalogInputsEngUnits[2] = 5;//volts
ntBnAnalogInputsEventState[0] = 0;//0=normal  1=fault  2=offnormal  3=high-limit
ntBnAnalogInputsEventState[1] = 0;
ntBnAnalogInputsEventState[2] = 0;
```

A
B
C
D
E

Enter the following information:

A—Index-to-object identifier offset = 0

NOTE: Each object identifier within a single BACnet device must have a unique value. The strategy uses an index-to-object identifier offset. For example the default index-to-object identifier offset for the binary outputs is 800. If you want to read the object name of an output point stored in index 2 of the output name table you would use object identifier 802. (Outputs are loaded into the tables in block 1).

C—Load analog input points into pointer table that will be used by BACnet

D—Load analog input names used by BACnet

E—Load analog input Engineering units used by BACnet

NOTE: There is a partial list of engineering codes listed on the left side of the BACnet_Protocol chart.

F—Load analog input Event State used by BACnet

NOTE: The Event State is set to 0 (normal) by default. The protocol chart will report the event state but has no logic to change the event state. If the state needs to change, it should be included in your strategy. Event states: 0=normal, 1=fault, 2=offnormal, 3=high-limit, 4=low-limit, 5=life-safety-alarm

NOTE: Out Of Service controls if the present state is writable. The protocol chart will determine the Out Of Service state by checking if communication is enabled to the point. Out Of Service = 0 Read Only, Out Of Service = 1 Read /Write

Analog Output

```
//Load analog output points into pointer table.
//Index  to object identifier offset = 200
//Table = potBnAnalogOutputs
//Load analog output names used by BACnet
//Table = stBnAnalogOutputsObjectName
//Load analog output Engineering units used by BACnet
//Table = ntBnAnalogOutputsEngUnits
//Load analog output Event State used by BACnet
//Table = ntBnAnalogOutputsEventState
nBnAnalogOutputIndexOffset = 200;                                          A
potBnAnalogOutputs[0] = &aoMeter;                                          B
potBnAnalogOutputs[1] = &aoPID_Output;
stBnAnalogOutputsObjectName[0] = "aoMeter";                                C
stBnAnalogOutputsObjectName[1] = "aoPID_Output";
ntBnAnalogOutputsEngUnits[0] = 98;//percent                                D
ntBnAnalogOutputsEngUnits[1] = 104;//revolutions-per-minute
ntBnAnalogOutputsEventState[0] = 0;//0=normal  1=fault  2=offnormal  3=high-li  E
ntBnAnalogOutputsEventState[1] = 0;
```

Enter the following information:

A—Index-to-object identifier offset = 200

NOTE: Each object identifier within a single BACnet device must have a unique value. The strategy uses an index-to-object identifier offset. For example the default index-to-object identifier offset for the binary outputs is 800. If you want to read the object name of an output point stored in index 2 of the output name table you would use object identifier 802. (Outputs are loaded into the tables in block 1).

B—Load analog output points into pointer table

C—Load analog output names used by BACnet

D—Load analog output Engineering units used by BACnet

NOTE: There is a partial list of engineering codes listed on the left side of the BACnet_Protocol chart.

E—Load analog output Event State used by BACnet

NOTE: The Event State is set to 0 (normal) by default. The protocol chart will report the event state but has no logic to change the event state. if the state needs to change, it should be included in your strategy. Event states: 0=normal, 1=fault, 2=offnormal, 3=high-limit, 4=low-limit, 5=life-safety-alarm

NOTE: The protocol chart will determine the Out Of Service state by checking if communication is enabled to the point. Out Of Service = 0 Communication is enabled to point, Out Of Service = 1 Communication is disabled to point.

Analog Values

```
//Load analog value points into pointer table.
//Index  to object identifier offset = 400
//Table = potBnAnalogValues
//Load analog value names used by BACnet
//Table = stBnAnalogValuesObjectName
//Load analog value Engineering units used by BACnet
//Table = ntBnAnalogValuesEngUnits
//Load analog value Event State used by BACnet
//Table = ntBnAnalogValuesEventState
//Load analog value Out-Of-Service used by BACnet
//Table = ntBnAnalogValuesOutOfService
A nBnAnalogValueIndexOffset = 400;
B potBnAnalogValues[0] = &fBnSampleAnalogValue0;
  potBnAnalogValues[1] = &fBnSampleAnalogValue1;
  potBnAnalogValues[2] = &fBnSampleAnalogValue2;
  potBnAnalogValues[3] = &fBnSampleAnalogValue3;
C stBnAnalogValuesObjectName[0] = "fBnSampleAnalogValue0";
  stBnAnalogValuesObjectName[1] = "fBnSampleAnalogValue1";
  stBnAnalogValuesObjectName[2] = "fBnSampleAnalogValue2";
  stBnAnalogValuesObjectName[3] = "fBnSampleAnalogValue3";
D ntBnAnalogValuesEngUnits[0] = 95;//no-units
  ntBnAnalogValuesEngUnits[1] = 5;//volts
  ntBnAnalogValuesEngUnits[2] = 95;//no-units
  ntBnAnalogValuesEngUnits[3] = 5;//volts
E ntBnAnalogValuesEventState[0] = 0;//0=normal  1=fault  2=offnormal  3=high-limit
  ntBnAnalogValuesEventState[1] = 0;
  ntBnAnalogValuesEventState[2] = 0;
  ntBnAnalogValuesEventState[3] = 0;
F ntBnAnalogValuesOutOfService[0] = 1;//BACnet Write Control
  ntBnAnalogValuesOutOfService[1] = 0;//Local Control
  ntBnAnalogValuesOutOfService[2] = 1;//BACnet Write Control
  ntBnAnalogValuesOutOfService[3] = 0;//Local Control
```

Enter the following information:

A—Index-to-object identifier offset = 400

NOTE: Each object identifier within a single BACnet device must have a unique value. The strategy uses an index-to-object identifier offset. For example the default index-to-object identifier offset for the binary outputs is 800. If you want to read the object name of an output point stored in index 2 of the output name table you would use object identifier 802. (Outputs are loaded into the tables in block 1).

B—Load analog value points into pointer table

C—Load analog value names used by BACnet

D—Load analog value Engineering units used by BACnet

NOTE: There is a partial list of engineering codes listed on the left side of the BACnet_Protocol chart.

E—Load analog value Event State used by BACnet

NOTE: The Event State is set to 0 (normal) by default. The protocol chart will report the event state but has no logic to change the event state. if the state needs to change, it should be included in your strategy. Event states: 0=normal, 1=fault, 2=offnormal, 3=high-limit, 4=low-limit, 5=life-safety-alarm

F—Load analog value Out-Of-Service used by BACnet

NOTE: Out Of Service controls if the present state is writable. The protocol chart will report the Out Of Service state but has no logic to change the state. if the state needs to change, it should be included in your strategy. Out Of Service = 0 Read Only, Out Of Service = 1 Read /Write

Binary Input

```
//Load binary input points into pointer table that will be used by BACnet.
//Index  to object identifier offset = 600
//Table = potBnBinaryInputs
//Load binary input names used by BACnet
//Table = stBnBinaryInputsObjectName
//Load binary input Event State used by BACnet
//Table = ntBnBinaryInputsEventState
//Load binary input Polarity used by BACnet
//Table = ntBnBinaryInputsPolarity
nBnBinaryInputIndexOffset = 600;                                                    A
potBnBinaryInputs[0] = &diSwitch_D0;                                                B
potBnBinaryInputs[1] = &diSwitch_D1;
potBnBinaryInputs[2] = &diButton_Ct2;
potBnBinaryInputs[3] = &diButton_Ct3;
stBnBinaryInputsObjectName[0] = "diSwitch_D0";                                      C
stBnBinaryInputsObjectName[1] = "diSwitch_D1";
stBnBinaryInputsObjectName[2] = "diButton_Ct2";
stBnBinaryInputsObjectName[3] = "diButton_Ct2";
ntBnBinaryInputsEventState[0] = 0;//0=normal  1=fault  2=offnormal  3=high-limit   D
ntBnBinaryInputsEventState[1] = 0;
ntBnBinaryInputsEventState[2] = 0;
ntBnBinaryInputsEventState[3] = 0;
ntBnBinaryInputsPolarity[0] = 0;//Normal  1 = Reverse                               E
ntBnBinaryInputsPolarity[1] = 0;
ntBnBinaryInputsPolarity[2] = 0;
ntBnBinaryInputsPolarity[3] = 1;
```

Enter the following information:

A—Index-to-object identifier offset = 600

NOTE: Each object identifier within a single BACnet device must have a unique value. The strategy uses an index-to-object identifier offset. For example, the default index-to-object identifier offset for the binary outputs is 800. If you want to read the object name of an output point stored in index 2 of the output name table you would use object identifier 802. (Outputs are loaded into the tables in block 1).

B—Load binary input points into pointer table that will be used by BACnet

C—Load binary input names used by BACnet

D—Load binary input Event State used by BACnet

NOTE: The Event State is set to 0 (normal) by default. The protocol chart will report the event state but has no logic to change the event state. if the state needs to change, it should be included in your strategy. Event states: 0=normal, 1=fault, 2=offnormal, 3=high-limit, 4=low-limit, 5=life-safety-alarm

E—Load binary input Polarity used by BACnet

NOTE: If the polarity is 0 (Normal), then the Active state of the Present Value is also the Active or On state of the physical point. If the polarity is 1 (Reverse), then the Active state of the Present Value is also the InActive or Off state of the physical point.

NOTE: Out Of Service controls if the present state is writable. The protocol chart will determine the Out Of Service state by checking if communication is enabled to the point. Out Of Service = 0 Read Only, Out Of Service = 1 Read /Write

Binary Output

```
//Load binary output points into pointer table that will be used by BACnet.
//Index  to object identifier offset = 800
//Table = potBnBinaryOutputs
//Load binary output names used by BACnet
//Table = stBnBinaryOutputsObjectName
//Load binary input Event State used by BACnet
//Table = ntBnBinaryOutputsEventState
//Load binary output Polarity used by BACnet
//Table = ntBnBinaryOutputsPolarity
nBnBinaryOutputIndexOffset = 800;                                          A
potBnBinaryOutputs[0] = &doAlarm;                                          B
potBnBinaryOutputs[1] = &doLED_D5;
potBnBinaryOutputs[2] = &doLED_D6;
potBnBinaryOutputs[3] = &doLED_D7;
stBnBinaryOutputsObjectName[0] = "doAlarm";                                C
stBnBinaryOutputsObjectName[1] = "doLED_D5";
stBnBinaryOutputsObjectName[2] = "doLED_D6";
stBnBinaryOutputsObjectName[3] = "doLED_D7";
ntBnBinaryOutputsEventState[0] = 0;//0=normal  1=fault  2=offnormal  3=high-limi  D
ntBnBinaryOutputsEventState[1] = 0;
ntBnBinaryOutputsEventState[2] = 0;
ntBnBinaryOutputsEventState[3] = 0;
ntBnBinaryOutputsPolarity[0] = 0;//Normal  1 = Reverse                      E
ntBnBinaryOutputsPolarity[1] = 0;
ntBnBinaryOutputsPolarity[2] = 1;
ntBnBinaryOutputsPolarity[3] = 0;
```

Enter the following information:

A—Index-to-object identifier offset = 800

NOTE: Each object identifier within a single BACnet device must have a unique value. The strategy uses an index-to-object identifier offset. For example the default index-to-object identifier offset for the binary outputs is 800. If you want to read the object name of an output point stored in index 2 of the output name table you would use object identifier 802. (Outputs are loaded into the tables in block 1).

B—Load binary output points into pointer table that will be used by BACnet

C—Load binary output names used by BACnet

D—Load binary input Event State used by BACnet

NOTE: The Event State is set to 0 (normal) by default. The protocol chart will report the event state but has no logic to change the event state. if the state needs to change, it should be included in your strategy. Event states: 0=normal, 1=fault, 2=offnormal, 3=high-limit, 4=low-limit, 5=life-safety-alarm

E—Load binary output Polarity used by BACnet

NOTE: If the polarity is 0 (Normal), then the Active state of the Present Value is also the Active or On state of the physical point. If the polarity is 1 (Reverse), then the Active state of the Present Value is also the InActive or Off state of the physical point.

NOTE: The protocol chart will determine the Out Of Service state by checking if communication is enabled to the point. Out Of Service = 0 Communication is enabled to point, Out Of Service = 1 Communication is disabled to point.

Binary Values

```
//Load binary value points into pointer table.
//Index  to object identifier offset = 1000
//Table = potBnBinaryValues
//Load binary value names used by BACnet
//Table = stBnBinaryValuesObjectName
//Load binary value Event State used by BACnet
//Table = ntBnBinaryValuesEventState
//Load binary value Out-Of-Service used by BACnet
//Table = ntBnBinaryValuesOutOfService
A  nBnBinaryValueIndexOffset = 1000;
B  potBnBinaryValues[0] = &nBnSampleBinaryValue0;
   potBnBinaryValues[1] = &nBnSampleBinaryValue1;
   potBnBinaryValues[2] = &nBnSampleBinaryValue2;
   potBnBinaryValues[3] = &nBnSampleBinaryValue3;
C  stBnBinaryValuesObjectName[0] = "nBnSampleBinaryValue0";
   stBnBinaryValuesObjectName[1] = "nBnSampleBinaryValue1";
   stBnBinaryValuesObjectName[2] = "nBnSampleBinaryValue2";
   stBnBinaryValuesObjectName[3] = "nBnSampleBinaryValue3";
D  ntBnBinaryValuesEventState[0] = 0;//0=normal  1=fault  2=offnormal  3=high-limit
   ntBnBinaryValuesEventState[1] = 0;
   ntBnBinaryValuesEventState[2] = 0;
   ntBnBinaryValuesEventState[3] = 0;
E  ntBnBinaryValuesOutOfService[0] = 0;//Local Control
   ntBnBinaryValuesOutOfService[1] = 1;//BACnet Write Control
   ntBnBinaryValuesOutOfService[2] = 0;//Local Control
   ntBnBinaryValuesOutOfService[3] = 1;//BACnet Write Control
```

Enter the following information:

A—Index-to-object identifier offset = 1000

NOTE: Each object identifier within a single BACnet device must have a unique value. The strategy uses an index-to-object identifier offset. For example the default index-to-object identifier offset for the binary outputs is 800. If you want to read the object name of an output point stored in index 2 of the output name table you would use object identifier 802. (Outputs are loaded into the tables in block 1).

B—Load binary value points into pointer table

C—Load binary value names used by BACnet

D—Load binary value Event State used by BACnet

NOTE: The Event State is set to 0 (normal) by default. The protocol chart will report the event state but has no logic to change the event state. if the state needs to change, it should be included in your strategy. Event states: 0=normal, 1=fault, 2=offnormal, 3=high-limit, 4=low-limit, 5=life-safety-alarm

E—Load binary value Out-Of-Service used by BACnet

NOTE: Out Of Service controls if the present state is writable. The protocol chart will report the Out Of Service state but has no logic to change the state. if the state needs to change, it should be included in your strategy. Out Of Service = 0 Read Only, Out Of Service = 1 Read /Write

Master Remote Property Tables

```
//Master Remote Property Tables
SetVariableFalse(nBnDeviceMasterAI);//True or False = Store data
potBnDeviceNameAI[1] = &stBnDeviceNameAI1;
potBnDeviceNameAI[2] = &stBnDeviceNameAI2;
potBnDeviceEventStateAI[1] = &ntBnDeviceEventStateAI1;
potBnDeviceEventStateAI[2] = &ntBnDeviceEventStateAI2;
potBnDeviceOutOfServiceAI[1] = &ntBnDeviceOutOfServiceAI1;
potBnDeviceOutOfServiceAI[2] = &ntBnDeviceOutOfServiceAI2;
potBnDevicePresentValueAI[1] = &ftBnDevicePresentValueAI1;
potBnDevicePresentValueAI[2] = &ftBnDevicePresentValueAI2;
potBnDeviceStatusFlagsAI[1] = &ntBnDeviceStatusFlagsAI1;
potBnDeviceStatusFlagsAI[2] = &ntBnDeviceStatusFlagsAI2;
potBnDeviceUnitsAI[1] = &ntBnDeviceUnitsAI1;
potBnDeviceUnitsAI[2] = &ntBnDeviceUnitsAI2;
```

The Master Remote Property Tables are used to store data from devices after using the ReadProperty or ReadPropertyMultiple subroutine. There are tables for each object type. There is a variable to disable storage for each type. The default is disabled. The Destination Address is used to select the table from the pointer tables.

If data is to be stored, the length of the pointer table will need to be set to the highest destination address + 1. If data storage for a device is not needed, just leave that pointer index empty. The length of the storage tables should be set to the highest instance number, + 1. The storage tables are used in the blocks named Store Data of the Complex ACK Section.

In the User Setup, the tables are loaded into pointer tables. When the ReadProperty subroutine is used, the destination and instance number are set. After the response is processed, a table will be selected for the property from the pointer table using the destination as the index to load. The response data will be stored in the table using the instance number as the index.

For example, if the ReadProperty subroutine is used to read the object name of the analog input with instance number of 25 of the device at address 5, the strategy will select the string table loaded in the pointer table for the analog input names at index 5. The name will be stored at index 25 of the string table moved from the pointer table.

Device Map Table

```
//Device Map table
//MNB-V1
stBnDeviceMap1[0] = "AI,1";
stBnDeviceMap1[1] = "AI,2";
stBnDeviceMap1[2] = "AI,3";
stBnDeviceMap1[3] = "AI,4";
stBnDeviceMap1[4] = "AI,17";
stBnDeviceMap1[5] = "AO,2";
stBnDeviceMap1[6] = "AV,2001";
stBnDeviceMap1[7] = "AV,2002";
stBnDeviceMap1[8] = "BV,2002";
stBnDeviceMap1[9] = "BV,2001";
stBnDeviceMap1[10] = "NA";
stBnDeviceMap1[11] = "NA";
stBnDeviceMap1[12] = "NA";
stBnDeviceMap1[13] = "MV,10101";
stBnDeviceMap1[14] = "AV,10106";
stBnDeviceMap1[15] = "AV,10107";
stBnDeviceMap1[16] = "MV,10201";
stBnDeviceMap1[17] = "AV,10206";
stBnDeviceMap1[18] = "AV,10207";
stBnDeviceMap1[19] = "MV,10301";
stBnDeviceMap1[20] = "AV,10306";
stBnDeviceMap1[21] = "AV,10307";
stBnDeviceMap1[22] = "AV,10401";
stBnDeviceMap1[23] = "AV,10451";
stBnDeviceMap1[24] = "MV,10452";
stBnDeviceMap1[25] = "BV,11801";
stBnDeviceMap1[26] = "AV,11850";
stBnDeviceMap1[27] = "AV,11851";
stBnDeviceMap1[28] = "AV,11852";
stBnDeviceMap1[29] = "MV,11853";
stBnDeviceMap1[30] = "AV,13313";
stBnDeviceMap1[31] = "AV,13326";
stBnDeviceMap1[32] = "AV,13332";
stBnDeviceMap1[33] = "BV,13353";
stBnDeviceMap1[34] = "AV,13354";
//The length of this table is last index + 2
//NA = No data stored at these indexes
```

(some text removed)

```
//Each map table is loaded in the pointer table
//There is a table loaded for each address. Note that the same ma
potBnDeviceMap[1] = &stBnDeviceMap1;
potBnDeviceMap[2] = &stBnDeviceMap2;
```

The device map table is used to map the object type and instance to an index of the storage tables. The map tables are loaded in a pointer table.

In the above tables the present value of ai,17 at address 2 would be stored in table ftBnDevicePresentValueAI2 at index 4.

The sequence for ReadProperty or ReadPropertyMultiple is after the subroutine loads the send buffer table. The command is sent when this controller has the token.

When the response is received the BACnet_Protocol chart will enable the BACnet_Process_ComplexACK chart. After the chart parses the response it checks if storage for that object type is enable and the map pointer table is not null. If it passes the test it loads the map table from the pointer table and does a lookup to find the matching entry. If a match is found the storage table is loaded from the pointer table and the data is stored at the index of the match.

Object Types and Supported Properties

The following tables show the supported BACnet object types and the properties supported for each object type:

Each table also includes the property data type, the identifier number, and whether the property is read or read/write.

Analog Input

BACnetObjectType = 0

Properties Supported	Property Data Type	Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	REAL	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R
Out_Of_Service	BOOLEAN	81	R
Units	BACnetEngineeringUnits	117	R

Analog Output

BACnetObjectType = 1

Properties Supported	Property Data Type	Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	REAL	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R

Properties Supported	Property Data Type	Identifier	Read/Write
Out_Of_Service	BOOLEAN	81	R
Units	BACnetEngineeringUnits	117	R

Analog Value

BACnetObjectType = 2

Properties Supported	Property Data Type	Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	REAL	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R
Out_Of_Service	BOOLEAN	81	R
Units	BACnetEngineeringUnits	117	R

Binary Input

BACnetObjectType = 3

Properties Supported	Property Data Type	Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	BACnetBinaryPV	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R
Out_Of_Service	BOOLEAN	81	R
polarity	BACnetPolarity	84	R

Binary Output

BACnetObjectType = 4

Properties Supported	Property Data Type	Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	BACnetBinaryPV	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R
Out_Of_Service	BOOLEAN	81	R
polarity	BACnetPolarity	84	R

Binary Value

BACnetObjectType = 5

Properties Supported	Property Data Type	Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
Present_Value	BACnetBinaryPV	85	R/W
Status_Flags	BACnetStatusFlags	111	R
Event_State	BACnetEventState	36	R
Out_Of_Service	BOOLEAN	81	R

Device

BACnetObjectType = 8

Properties Supported	Property Data Type	Identifier	Read/Write
Object_Identifier	BACnetObjectIdentifier	75	R
Object_Name	CharacterString	77	R
Object_Type	BACnetObjectType	79	R
System_Status	BACnetDeviceStatus	112	R
Vender_Name	CharacterString	121	R
Vendor_Identifier	Unsigned16	120	R
Model_Name	CharacterString	70	R
Firmware_Revision	CharacterString	44	R
Application_Software_Version	CharacterString	12	R
Protocol_Version	Unsigned	98	R
Protocol_Revision	Unsigned	139	R
Protocol_Sevices_Supported	BACnetServicesSupported	97	R
Protocol_Object_Type_Supported	BACnetObjectTypesSupported	96	R
Object_List	Sequence of BACnetObjectIdentifier	76	R
Max_APDU_Length_Supported	Unsigned	62	R
Segmentation_Supported	BACnetSegmentation	107	R
Local_Time	Time	57	R
Local Date	Date	56	R
UTC_Offset	Integer	119	R/W
APDU_Timeout	Unsigned	11	R
Number_Of_APDU_Reties	Unsigned	73	R
Max_Master	Unsigned	64	R/W
Max-Info_Frames	Unsigned	63	R/W
Device_Address_Binding	Sequence of BACnetAddressBinding	30	R
Database_Revision	Unsigned	155	R

Master Subroutines

This section describes the master subroutines included in the kit and what each subroutine does, including the prompts for each subroutine.

The kit includes the following subroutines:

Who-Is

Loads send table with Who-Is request. The Protocol chart will send it when it has the token.

Prompt	Type	Description
Parameter Table	Integer 32 Table	Index 1 = Destination Address (0-255) Index 2 = Low Device Instance (0-4194303) 0 = all instances Index 3 = High Device Instance (0-4194303) 0 = all instances
MasterLockFlag	Integer 32	Used by strategy (nBnMasterLockFlag1)
WaitingTransmit	Integer 32	Used by strategy (nBnMasterDataFrameWaitingTransmit)
SendDataTable	Integer 32 Table	Used by strategy (ntBnMasterSend_Data)
SendIndexTable	Integer 32 Table	Used by strategy (ntBnMasterSendIndexTable)
Put Status In	Integer 32	Subroutine system errors

Get Name of Binded Devices

Loads the send table with ReadProperty Object Name request for each device in the binding table. The Protocol chart will send it when it has the token.

Prompt	Type	Description
BindingInstanceT	Integer 32 Table	Used by strategy (ntBnMasterBindingInstance)
BindingDestT	Integer 32 Table	Used by strategy (ntBnMasterBindingDestination)
SNET Table	String Table	Used by strategy (stBnMasterSNETSLENSADR)
Binding Index	Integer 32	Used by strategy (nBnMasterBindingIndex)
Invoke ID	Integer 32	Used by strategy (nBnMasterInvokeID)
MasterLockFlag	Integer 32	Used by strategy (nBnMasterLockFlag1)
WaitingTransmit	Integer 32	Used by strategy (nBnMasterDataFrameWaitingTransmit)
SendDataTable	Integer 32 Table	Used by strategy (ntBnMasterSend_Data)
SendIndexTable	Integer 32 Table	Used by strategy (ntBnMasterSendIndexTable)
Put Status In	Integer 32	Subroutine system errors

ReadProperty

Loads the send table with ReadProperty request. The Protocol chart will send it when it has the token.

Prompt	Type	Description
Parameter Table	Integer 32 Table	Index 1 = Destination Address (0-255) Index 2 = Object Identifier Index 3 = Object Instance (0-4194303) Index 4 = Property Identifier Index 5 = Not used Index 6 = DNET 0 = Local Index 7 = Octet 1 of IP Index 8 = Octet 2 of IP Index 9 = Octet 3 of IP Index 10 = Octet 4 of IP Index 11 = UDP port
Invoke ID	Integer 32	Used by strategy (nBnMasterInvokeID)
MasterLockFlag	Integer 32	Used by strategy (nBnMasterLockFlag1)
WaitingTransmit	Integer 32	Used by strategy (nBnMasterDataFrameWaitingTransmit)
SendDataTable	Integer 32 Table	Used by strategy (ntBnMasterSend_Data)
SendIndexTable	Integer 32 Table	Used by strategy (ntBnMasterSendIndexTable)
Put Status In	Integer 32	Subroutine system errors

WriteProperty

Loads the send table with WriteProperty request. The Protocol chart will send it when it has the token.

Prompt	Type	Description
Parameter Table	Integer 32 Table	Index 1 = Destination Address (0-255) Index 2 = Object Identifier Index 3 = Object Instance (0-4194303) Index 4 = Property Identifier Index 5 = Application Data Type Index 6 = DNET 0 = Local Index 7 = Octet 1 of IP Index 8 = Octet 2 of IP Index 9 = Octet 3 of IP Index 10 = Octet 4 of IP Index 11 = UDP port Index 16 = Priority 0 = not used
Index 17 = Relinquish 1 = true 0 = false	Invoke ID	Integer 32
Used by strategy (nBnMasterInvokeID)	MasterLockFlag	Integer 32
Used by strategy (nBnMasterLockFlag1)	WaitingTransmit	Integer 32
Used by strategy (nBnMasterDataFrameWaitingTransmit)	SendDataTable	Integer 32 Table
Used by strategy (ntBnMasterSend_Data)	SendIndexTable	Integer 32 Table
Used by strategy (ntBnMasterSendIndexTable)	StringValue	String
String to write		
Octet String = hexadecimal octets 2 digits, e.g. 01C0		
Character String = ANSI		
Bit String = 0s and 1s		
Date = mm/dd/yyy/dow Monday = 1 FF = Unspecified		
Time = hh:mm:ss:ms FF = Unspecified		
Object Identifier = Object Type,Instance Number	Int32Value	Integer 32
Integer to write		
Boolean = 1 or 0		
Unsigned		
Integer		
Enumerated	FloatValue	Float
Float to write		
Real	Put Status In	Integer 32
Subroutine system errors		

SubscribeCOV

Loads the send table with SubscribeCOV and cancels subscription requests. The Protocol chart will send it when it has the token.

Prompt	Type	Description
Parameter Table	Integer 32 Table	Index 1 = Destination Address (0-255) Index 2 = Object Identifier Index 3 = Object Instance (0-4194303) Index 4 = Property Identifier Index 5 = Not used Index 6 = DNET 0 = Local Index 7 = Octet 1 of IP Index 8 = Octet 2 of IP Index 9 = Octet 3 of IP Index 10 = Octet 4 of IP Index 11 = UDP port Index 12 = Process Identifier Index 13 = Issue Confirmed Notification 1 = Confirmed Index 14 = Lifetime in seconds 0 = forever Index 15 = Cancel subscription 1 = cancel Resets index 15 to 0 after execution
Invoke ID	Integer 32	Used by strategy (nBnMasterInvokeID)
MasterLockFlag	Integer 32	Used by strategy (nBnMasterLockFlag1)
WaitingTransmit	Integer 32	Used by strategy (nBnMasterDataFrameWaitingTransmit)
SendDataTable	Integer 32 Table	Used by strategy (ntBnMasterSend_Data)
SendIndexTable	Integer 32 Table	Used by strategy (ntBnMasterSendIndexTable)
Put Status In	Integer 32	Subroutine system errors

Get Address From Binded Devices

Used by PAC Display interface (MMI) to fill in the address by selecting the binding index.

Prompt	Type	Description
Reference Index	Integer 32	Index of device in binding table
BindingInstanceT	Integer 32 Table	Used by strategy (ntBnMasterBindingInstance)
BindingDestT	Integer 32 Table	Used by strategy (ntBnMasterBindingDestination)
SNET Table	String Table	Used by strategy (stBnMasterSNETSLENSADR)
Binding Index	Integer 32	Used by strategy (nBnMasterBindingIndex)
Parameter Table	Integer 32 Table	Used by strategy (ntBnMasterParameterTable)
Put Status In	Integer 32	Subroutine system errors

Get Data Type

Used by MMI to auto select data type for the selected property.

Prompt	Type	Description
Parameter Table	Integer 32 Table	Used by strategy (ntBnMasterParameterWTable)
PropertyToData	Integer 32 Table	Used by MMI (ntBnMasterPropertyToDataType)
DataTypeDesc	String Table	Used by MMI (stBnMasterApplicationDataTypeDest)
MMI Data Type	String	Used by MMI (sBnMasterMMIDataType)
MMI P Table	String Table	Used by MMI (stBnMasterMMIPPrompt)
MMI Prompt	String	Used by MMI (sBnMasterMMIPPrompt)
MMIWrite	Integer 32	Used by MMI (sBnMasterMMIWrite)
Put Status In	Integer 32	Subroutine system errors

Get Data Type Manual

Used by the MMI to auto select data type manually.

Prompt	Type	Description
Parameter Table	Integer 32 Table	Used by strategy (ntBnMasterParameterWTable)
DataTypeDesc	String Table	Used by MMI (stBnMasterApplicationDataTypeDest)
MMI Data Type	String	Used by MMI (sBnMasterMMIDataType)
MMI P Table	String Table	Used by MMI (stBnMasterMMIPPrompt)
MMI Prompt	String	Used by MMI (sBnMasterMMIPPrompt)
MMIWrite	Integer 32	Used by MMI (sBnMasterMMIWrite)
Put Status In	Integer 32	Subroutine system errors

NOTE: Parameters are listed as Used by strategy (variable) or Used by MMI (variable). The listed variable must be used.

Remove Binded Device

Removes a device for the binding table.

Prompt	Type	Description
DeleteIndex	Integer 32	Used by strategy (nBnMasterRemoveBindingIndex)
BindingInstanceT	Integer 32 Table	Used by strategy (ntBnMasterBindingInstance)
BindingNameT	String Table	Used by strategy (stBnMasterBindingName)
BindingDestT	Integer 32 Table	Used by strategy (ntBnMasterBindingDestination)
SNET Table	String Table	Used by strategy (stBnMasterSNETSLENSADR)
Binding Index	Integer 32	Used by strategy (nBnMasterBindingIndex)
MasterLockFlag	Integer 32	Used by strategy (nBnMasterLockFlag1)
Put Status In	Integer 32	subroutine system errors

Load ReadPropertyMultiple Tables

Used by MMI to load tables for ReadPropertyMultiple. Loads one object per call.

Prompt	Type	Description
Object ID	Integer 32	Object Identifier
Instance Number	Integer 32	Object Instance (0-4194303)
Property	Integer 32	Property Identifier
Invoke ID	Integer 32	Used by strategy (nBnMasterInvokeID)
Object ID Table	Integer 32 Table	Index 0 = 999 = Clear last entry in tables
		Index 0 = 9999 = Clear all entries in tables
Instance Table	Integer 32 Table	Index 0 = Last index used
Property Table	Integer 32 Table	Index 0 = Not used
MasterLockFlag	Integer 32	Used by strategy (nBnMasterLockFlag1)
Put Status In	Integer 32	Subroutine system errors

ReadPropertyMultiple

Loads the send table with ReadPropertyMultiple request. The Protocol chart will send it when it has the token.

Prompt	Type	Description
Parameter Table	Integer 32 Table	Index 1 = Destination Address (0-255)
		Index 2 = Not used
		Index 3 = Not used
		Index 4 = Not used
		Index 5 = Not used
		Index 6 = DNET 0 = Local
		Index 7 = Octet 1 of IP
		Index 8 = Octet 2 of IP
		Index 9 = Octet 3 of IP
		Index 10 = Octet 4 of IP
		Index 11 = UDP port
Invoke ID	Integer 32	Used by strategy (nBnMasterInvokeID)
MasterLockFlag	Integer 32	Used by strategy (nBnMasterLockFlag1)
WaitingTransmit	Integer 32	Used by strategy (nBnMasterDataFrameWaitingTransmit)
SendDataTable	Integer 32 Table	Used by strategy (ntBnMasterSend_Data)
SendIndexTable	Integer 32 Table	Used by strategy (ntBnMasterSendIndexTable)
Object ID Table	Integer 32 Table	Index 0 = Not used
Instance Table	Integer 32 Table	Index 0 = Last index used
Property Table	Integer 32 Table	Index 0 = Not used
Put Status In	Integer 32	subroutine system errors

Master Subroutines Responses

The master subroutine responses are listed here separately because the master subroutines only send the commands. Each response is processed by the protocol chart. This section shows which variables in the response data are stored.

I-Am Service

I-Am is received and stored by the Protocol chart.

Object	Description
(ntBnMasterBindingInstance)	Received device instance number
(ntBnMasterBindingSegmentation)	Received device segmentation
(ntBnMasterBindingVendor)	Received device vendor number
(ntBnMasterBindingDestination)	Received device destination address
(ntBnMasterBindingAPDULength)	Received device APDU maximum length
(stBnMasterSNETSLENSADR)	Received device SNET, SLEN and SADR

Get Name of Binded Devices

ComplexACK is received by the Protocol chart.

Object	Description
(stBnMasterBindingNames)	Object names stored for correct index based on instance number location of the table (ntBnMasterBindingInstance)

ReadPropertyACK

ComplexACK is received by the Protocol chart.

Object	Description
(nBnReceiveDeviceInteger)	Received property that uses signed, unsigned integer and Boolean.
(fBnReceiveDeviceFloat)	Received property that uses real numbers.
(sBnReceiveDeviceString)	Received property that use bit strings, character strings, octet strings, date, time and object identifiers.
(stBnReceiveDeviceString)	Received property that use bit strings, character strings, octet strings, date, time and object identifiers in a list.

WritePropertyACK

SimpleACK is received by the Protocol chart.

Object	Description
(sBnReceiveDeviceString)	Received status. Ok or error description

COV Notification Service

Notifications are received by the Protocol chart.

Object	Description
(stBnReceiveCOVMessage)	Received notification messages both confirmed and unconfirmed.

ReadPropertyMultipleACK

ComplexACK is received by the BACnet_Process_ComplexACK chart.

Object	Description
(stBnDeviceAddressRPM)	Stored in order received

2: PAC Display Example

The BACnetVAV PAC Display example project is included in the integration kit zip file to help you get started using the PAC Control charts and subroutines.

1. Make sure the BACnetInt.idb strategy is running on your control engine.
2. Open the example project, BACnetVAV.UUI, in PAC Display Configurator.
3. Select File > Save Project and Load Runtime.

   The project's Main window opens.
4. See the following sections to configure the Main window and the VAV Poll window.

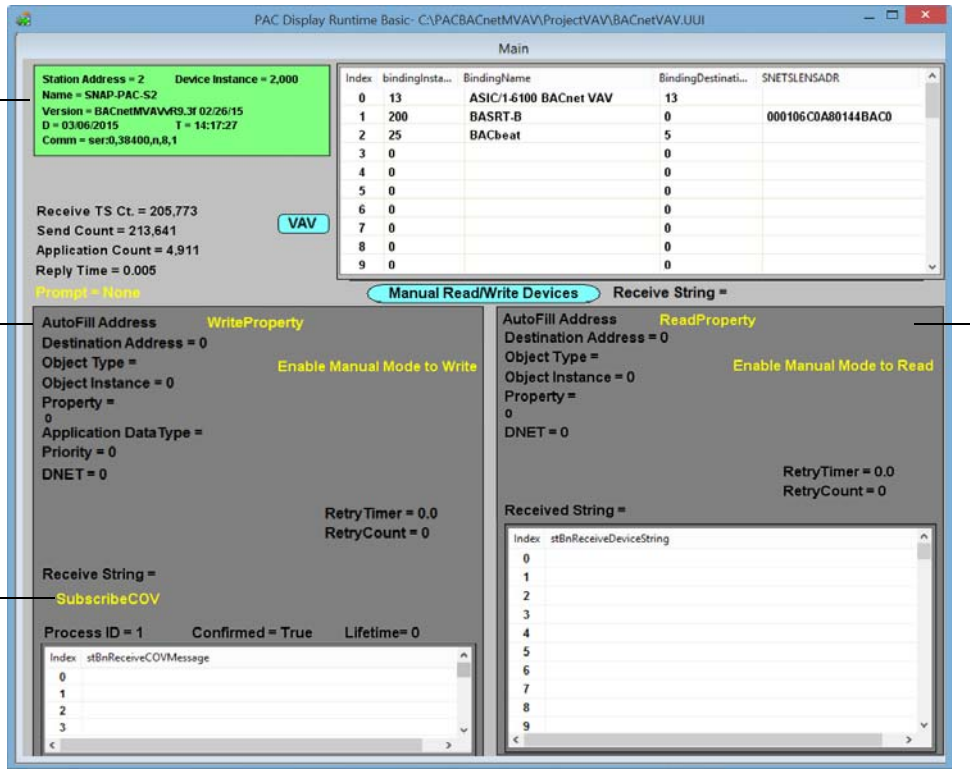
   "Configuring the Main Window" on page 28

   "Configuring the VAV Poll Window" on page 36

Configuring the Main Window

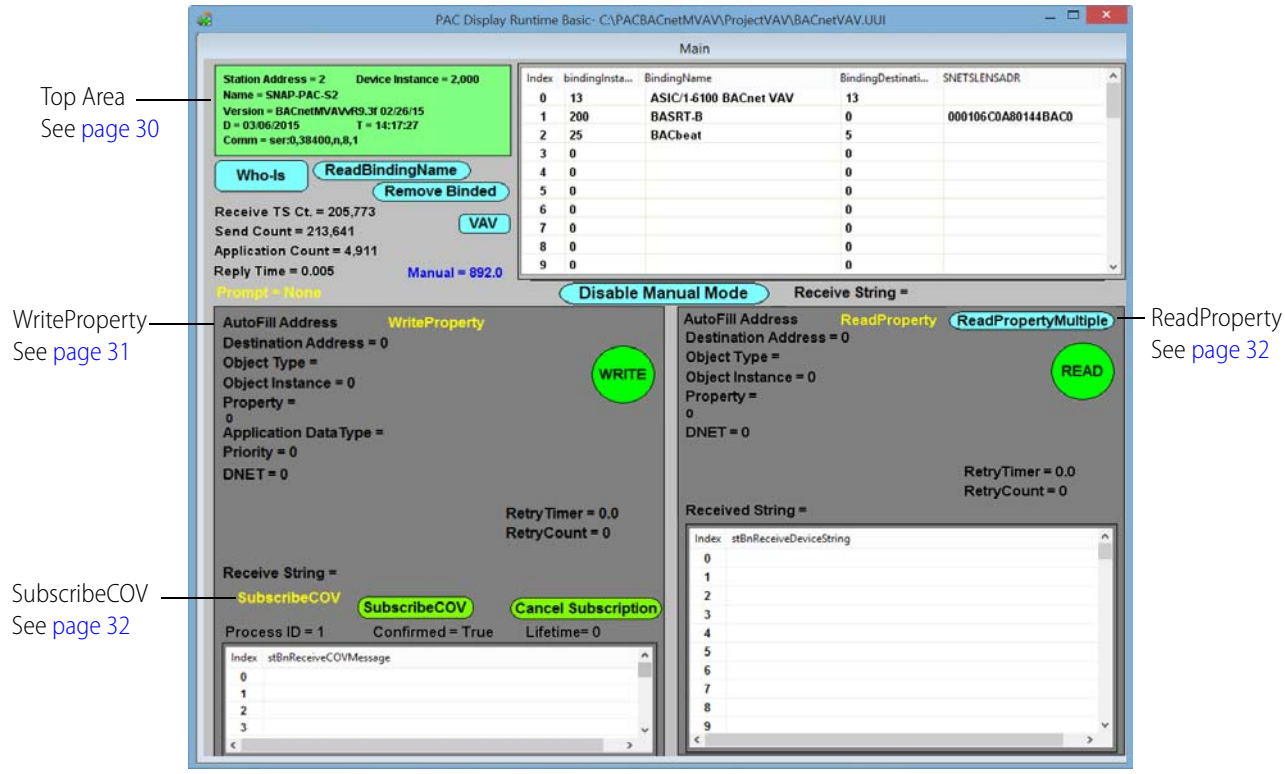
The Main Window can be displayed either with the manual mode disabled or enabled as shown here.

Main Window with Manual Mode Disabled



Main Window with Manual Mode Enabled

When manual mode is enabled, the READ, WRITE and ReadPropertyMultiple buttons are visible so you can manually read and write objects. To enable manual mode, click Manual Read/Write Devices in the middle of the page. Polling is disabled while in manual mode. A timer will disable manual mode after 600 seconds of inactivity if the operator forgets to disable manual mode.



Configure the project parameters on the Main window as described in the following sections.

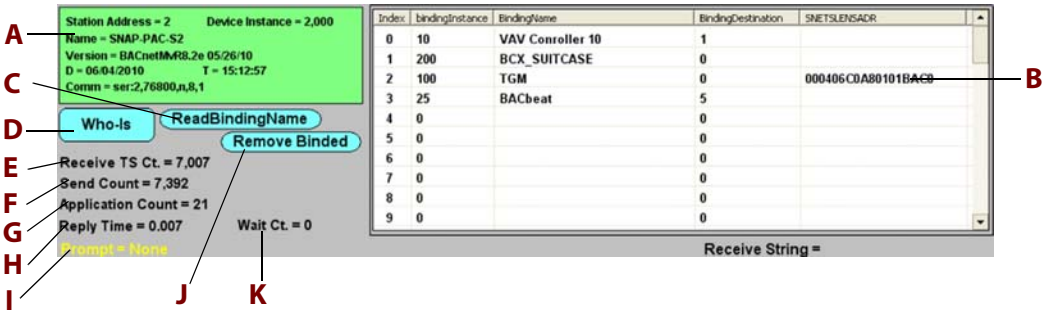
- "Top Area" on page 30
- "WriteProperty" on page 31
- "SubscribeCOV Area" on page 32
- "ReadProperty Area" on page 32
- "Selecting a Property (PropertyIden Window)" on page 33

Also see the following sections for information on windows you can open from the Main window:

- "Selecting a Property (PropertyIden Window)" on page 33
- "ReadPropertyMultiple Window" on page 35

Top Area

Configure the parameters in this section as necessary.



A—*Top Left.* The Station Address, Device Instance, Device Name, Comm Handle string, Date and time are displayed. These are read only.

B—*Top Right.* The binding tables list the device instance, device name, destination address, Source Network (SNET), Source MAC Layer Address Length (SLEN), and the Source MAC Layer Address (SADR).

C—*ReadBindingName button.* Starts a subroutine that will get the device name for devices in the binding table.

D—*Who-Is button.* Opens the Who-Is window for you to enter the Destination Address, the Low Device Instance, and the High Device Instance.

Destination Address: The device address the Who-Is command is sent to. An address of 255 denotes broadcast.

Low Device Instance and High Device Instance number: Each device in the network has an instance number. This is a filter that limits the number of devices responding to the command. If the low and high are left at 0 then all devices will respond.

E—*Receive TS CTs.* Received frames for this address.

F—*Send Count.* Sent frames from this address.

G—*Application Count.* Frames sent to application layer.

H—*Reply Time.* The time between receiving an application request from a device and the complete response.

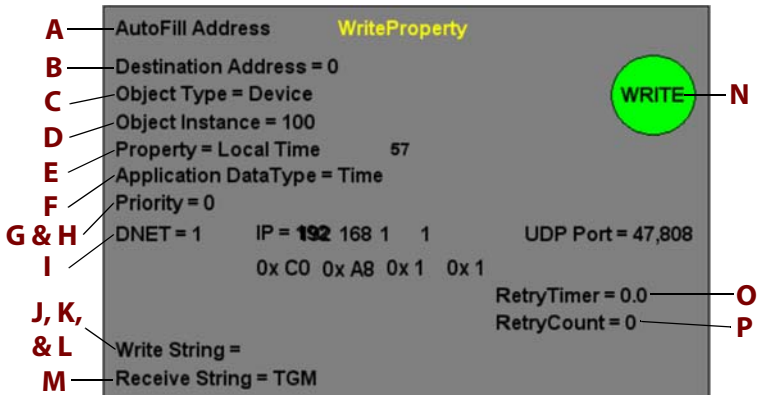
I—*Prompt.* As objects are selected on the display pages the prompt will have information on what is needed.

J—*Remove Binded button.* Starts a subroutine that will remove an entry for the binding table.

K—*Wait Ct.* The frames waiting to be transmitted.

WriteProperty

Configure Destination Address, Object Type, Object Instance, Property, Application Data Type, and DNET. Use Autofill Address to load the Destination Address, Object (Device) Instance, and DNET.



A—*AutoFill Address.* Allows you to select an index from the binding table. It provides the destination address, Object instance and Destination Network (DNET) if needed.

B—*Destination Address.* Manually enter the destination address.

C—*Object Type.* Opens the Object Type Write window. Select an object type and the window closes.

D—*Object Instance.* Enter the object instance number.

E—*Property.* Opens the PropertyIden window. Select a property and the window closes. To select a proprietary property, see "Selecting a Property (PropertyIden Window)" on page 33.

F—*Application Data Type.* Opens the application data type window. If you select a standard property, the data type is selected automatically.

G—*Priority.* Enter the priority. 0 = not used

H—*Relinquish.* Click to toggle between True and False. This parameter is visible when needed.

I—*DNET (Destination Network).* Enter 0 for no DNET. A value greater than 0 allows you to enter the IP address and UDP port.

J—*Write Integer .* Used to write properties that use unsigned integers and Boolean. This parameter is visible when needed.

K—*Write Float.* Used to write properties that use real numbers. This parameter is visible when needed.

L—*Write String.* Used to write properties that use bit strings, character strings, octet strings, date, time and object identifiers. This parameter is visible when needed.

M—*Receive String.* The returned string.

N—*Write button.* Starts WriteProperty subroutine.

O—*RetryTimer.* Delay time before retry. Display only.

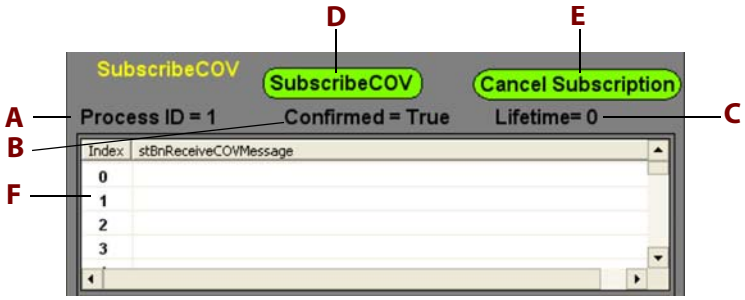
P—*RetryCount.* Retry count. Display only.

SubscribeCOV Area

If supported by the BACnet device, you can subscribe to Change of Value (COV) reporting from the device by configuring parameters under WriteProperty and SubscribeCOV.

For WriteProperty, configure Destination Address, Object Type, Object Instance, and DNET. For more information, see "WriteProperty" on page 31.

For SubscribeCOV, configure Process ID, Confirmed, and Lifetime.



A—*Process ID.* Enter the ID number

B—*Confirmed.* True for Issue confirmed Notification.

C—*Lifetime.* Time in seconds. 0 = forever.

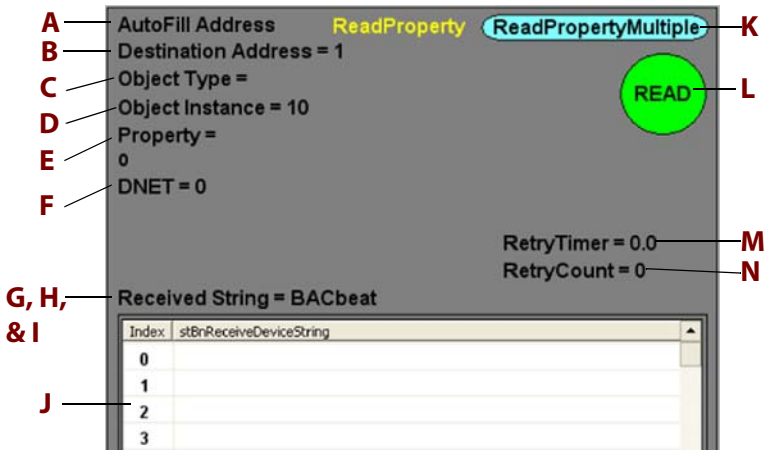
D—*SubscribeCOV.* Starts SubscribeCOV subroutine.

E—*Cancel Subscription.* Starts SubscribeCOV subroutine.

F—*COV Table Box.* Lists the received COV messages.

ReadProperty Area

Configure Destination Address, Object Type, Object Instance, Property, and DNET. Use the Autofill Address to load the Destination Address, Object (Device) Instance, and DNET.



A—*AutoFill Address.* Allows you to select an index from the binding table. It will fill in the Destination Address, Object instance, and DNET (see below) if needed.

B—*Destination Address.* Manually enter the address.

C—*Object Type.* Opens object type window. Select an object type and the window closes.

D—*Object Instance.* Enter object instance number.

E—*Property.* Opens the property window. Select a property and the window closes. To select a proprietary property, see "Selecting a Property (PropertyIden Window)" on page 33.

F—*DNET (Destination Network).* Enter 0 for no DNET. A value greater than 0 allows you to enter the IP address and UDP port.

G—*Received Integer.* Received property that uses signed, unsigned integer, and Boolean. This parameter is visible when needed.

H—*Received Float.* Received property that uses real numbers. This parameter is visible when needed.

I—*Received String.* Received property that uses bit strings, character strings, octet strings, date, time and object identifiers. This parameter is visible when needed.

J—*Received String Table box.* Received property that uses bit strings, character strings, octet strings, date, time and object identifiers in a list.

K—*ReadPropertyMultiple button.* Opens the ReadPropertyMultiple window. See "ReadPropertyMultiple Window" on page 35.

L—*Read.* Starts ReadProperty subroutine.

M—*RetryTimer.* Delay time before retry. Display only.

N—*RetryCount .* Retry count. Display only.

Selecting a Property (PropertyIden Window)

The PropertyIden window becomes visible when you select Property under either ReadProperty or WriteProperty.

To select a non-proprietary property:

1. In PAC Display, open the example project, BACnetVAV.UUI.
2. On the Main window under ReadProperty, click Property.

   The PropertyIden window opens.
3. Select a property.

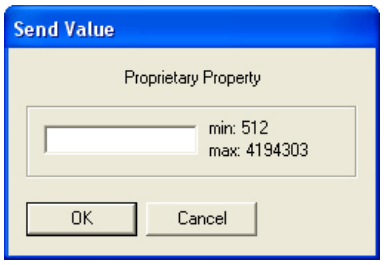
   The window closes.

To select a proprietary property:

1. In PAC Display, open the example project, BACnetVAV.UUI.
2. On the Main window under ReadProperty, click Property.

   The PropertyIden window opens.

3. Select Proprietary to open the Send Value dialog box.



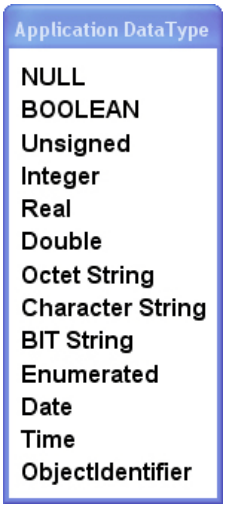
4. Enter the number of the proprietary property.

   Proprietary numbers are assigned by the device manufacturer, and listed in the manufacturer's Protocol Implementation Conformance (PIC).

The remaining instructions are for WriteProperty only.

5. Under WriteProperty, click Application Data Type.

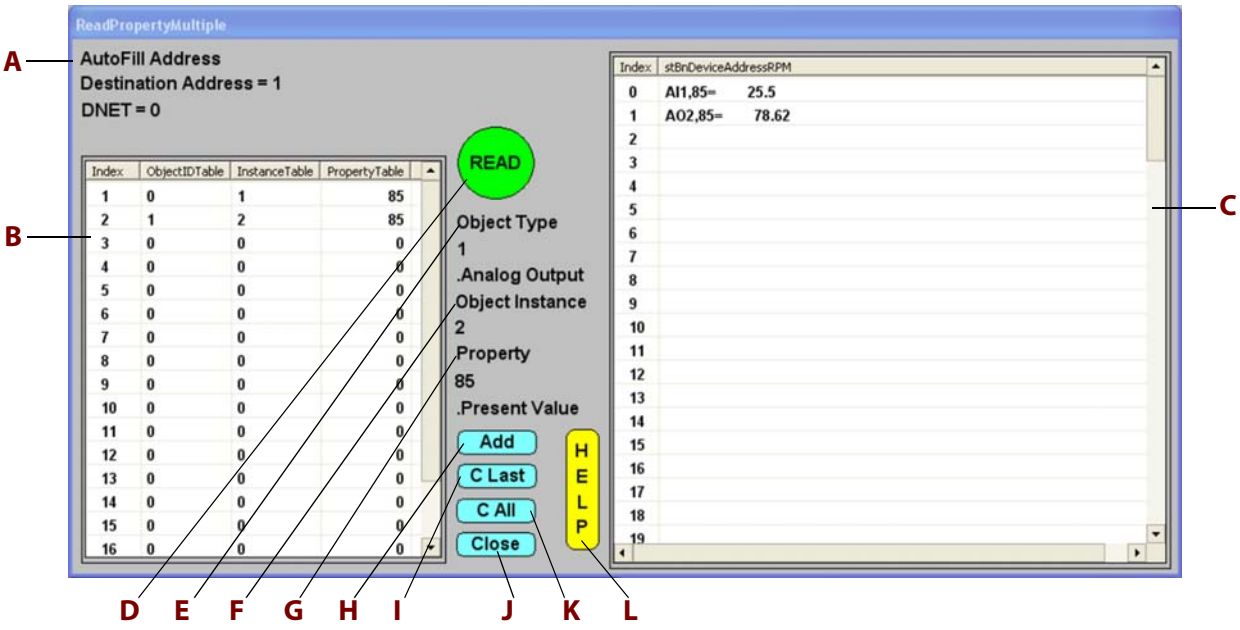
6. On the Application DataType dialog box, select the data type.



The dialog box closes.

ReadPropertyMultiple Window

This window is opened by clicking the ReadPropertyMultiple button in the ReadProperty area of the Main window. See "ReadProperty Area" on page 32.



A—*AutoFill Address.* Allows you to select an index from the binding table. It will fill in the Destination Address, Object instance, and DNET (see below) if needed.

B—*Left tables.* List the objects to read.

C—*Right table.* Lists responses in the order received.

D—*Read button.* Starts a subroutine ReadPropertyMultiple.

E—*Object Type.* Opens the Object Type window.

F—*Object Instance.* Enter the object instance number.

G—*Property.* Opens the property select window.

H—*Add button .* Starts a subroutine to add object type, instance number, and property to the left tables.

I—*C Last button .* Starts a subroutine to remove the last entry in left tables.

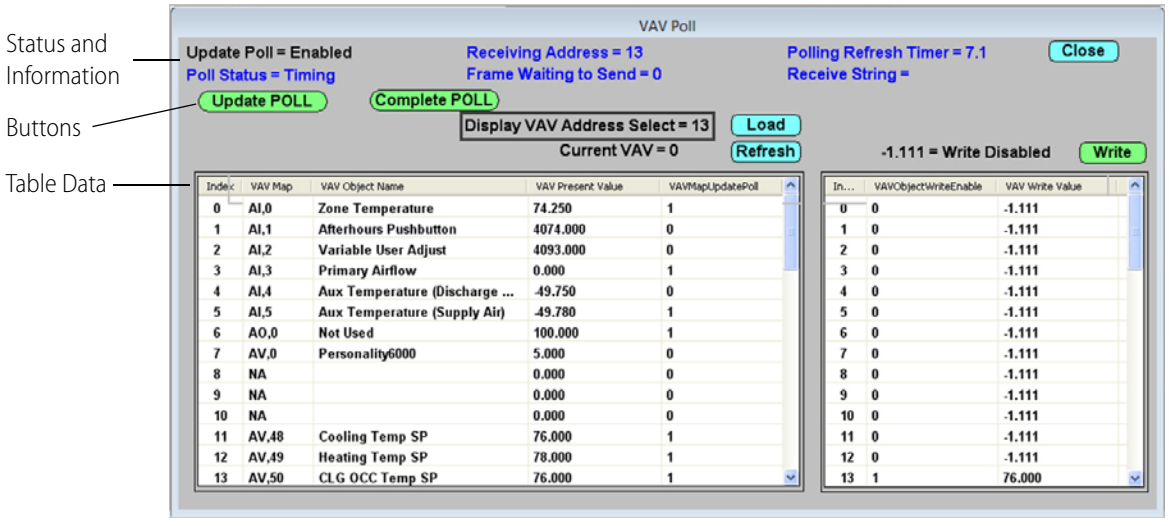
J—*Close button.* Closes the window.

K—*C All button.* Starts a subroutine to Clear the tables.

L—*Help button.* Opens a help window.

Configuring the VAV Poll Window

1. Click the VAV button on the Main window to open the VAV Poll window.
   The data will load after the complete poll. Then it will refresh after each Update poll.
2. Configure the parameters on the VAV Poll window as follows.



Status and Information

Update Poll—Toggle between Auto Polling enabled / disabled.

Poll Status—Current Status of Read_VAV chart.

Receiving Address—Address of last packet received.

Frame Waiting to Send—Number of frames in buffer waiting to send.

Polling Refresh Timer—Countdown timer to Update Poll.

Receive String—Lists any error messages from the BACnet device or Timeout if the BACnet device does not respond. It may be blank for some commands.

Display VAV Address Select—Select an address to display data.

Current VAV—Current address of data displayed.

Buttons

Update Poll—Execute update poll.

Complete Poll—Execute complete poll.

Load—Refresh with new address data.

Refresh—Refresh data from controller read table. Normally data is refreshed after the Update Poll.

Write—Write data to the selected address.

VAVObjectWriteEnabled—Enable write to an object. 1 = enabled. Write button will write to every object with 1.

VAV Write Value—Value to write. If write to an object is disabled it will be set to -1.111 by strategy.

Table Data

VAV Map—Device map for select address.

VAV Object Name—Object names for selected address.

VAVMapUpdatePoll—Objects included in Update poll. 1 = Enable.

VAVObjectWriteEnable—Object included in Write. 1 = Enable.

VAV Write Value—Value to write. If write to an object is disabled it will be set to -1.111 by strategy.

3: VAV Controller Polling Chart (Read_VAV)

The VAV Controller Polling chart (Read_VAV) polls from 1 to 45 VAV (Variable Air Volume) controllers or any BACnet device. The Object type and instances are mapped in the BACnet Protocol chart User Setup block. The strategy supports 4 maps.

A map can be assigned to each address. At startup the example strategy reads every object instance in the map for each address. The value data is stored in a float table for each address and the object name is stored in a string table for each address.

Object instances for each map can be assigned to the update poll. The update poll repeats at a settable interval.

The example strategy can write to each VAV controller. The values to write are stored in a float table for each address. If the value to write is not a float, the strategy corrects the data type before sending to the VAV controller.

User Setup blocks are provided in the BACnet_Protocol and Read_VAV charts for you to set up the example strategy for your system. See "Entering User Setup Parameters" on page 3 and "VAV Map Tables" on page 39.

To obtain the example strategy: On the Opto 22 website, search for PACBACnetVAVExample, and then download zip file. Open the file and extract the contents to a directory on your hard drive.

In This Chapter:

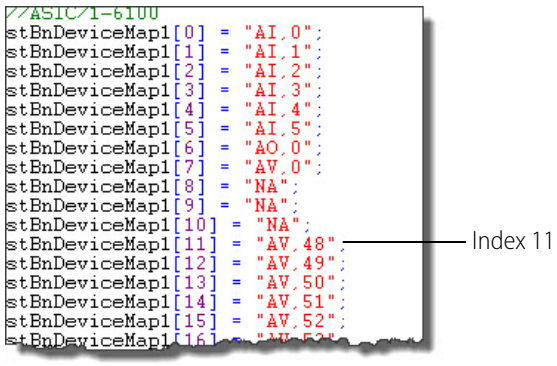
VAV Map Tables

To set up the VAV map tables, open the User Setup block of the BACnet_Protocol chart.

If store data is enabled for the readProperty or readPropertyMultiple master subroutines, this will store an instance number at the index number in this table.

The format of each index is Object Type,Object Instance (e.g., AV,48).

For example, if you want to store an analog value of 48 at index 11 of the analog value storage tables, enter (stBnDeviceMap1[11] = "AV,48";). The write value is in index 11 of the float write table for that address.



The strategy is setup to use 4 VAV maps. The map is assigned per address in the User Setup block of the BACnet_Protocol chart

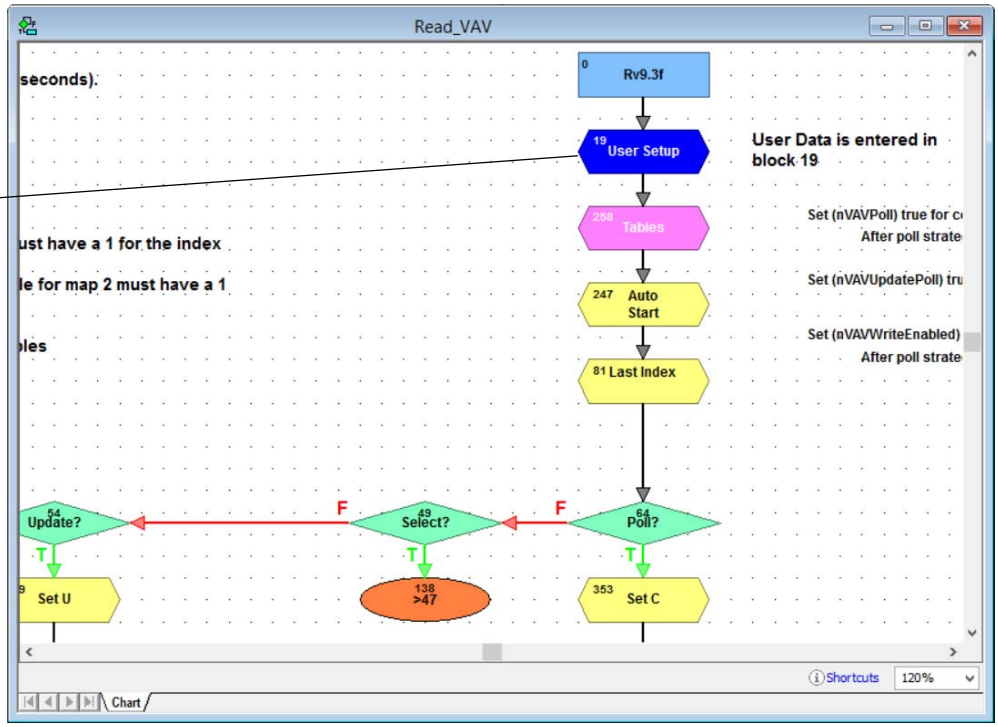
The strategies use float tables for the read and write values for each address. The data types are corrected by the strategy based on the setting in the User Setup block of the VAV_Poll chart before sending.

Entering User Setup Parameters

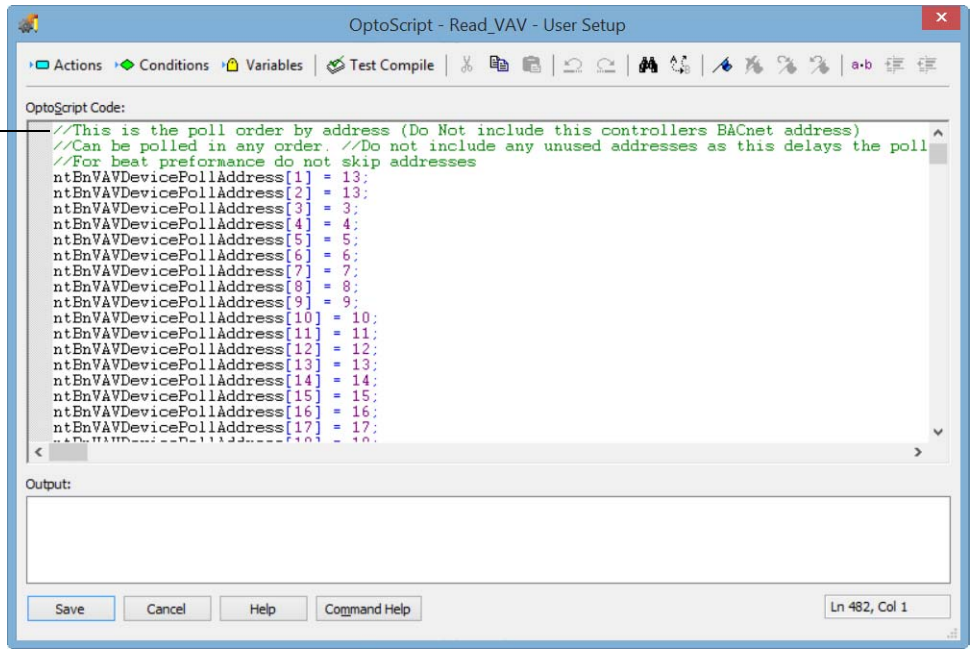
The user setup parameters are entered in Block 19 of the Read_VAV chart.

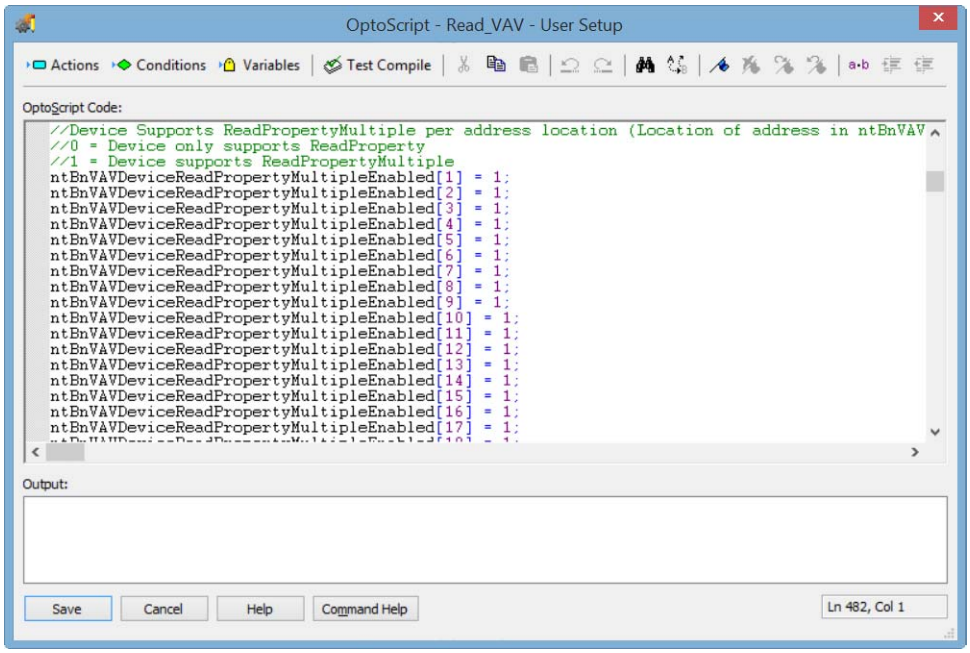
1. With your strategy open in PAC Control, open the Read_VAV chart.
2. Double-click on block 19, User Setup to open the Opto Script window.

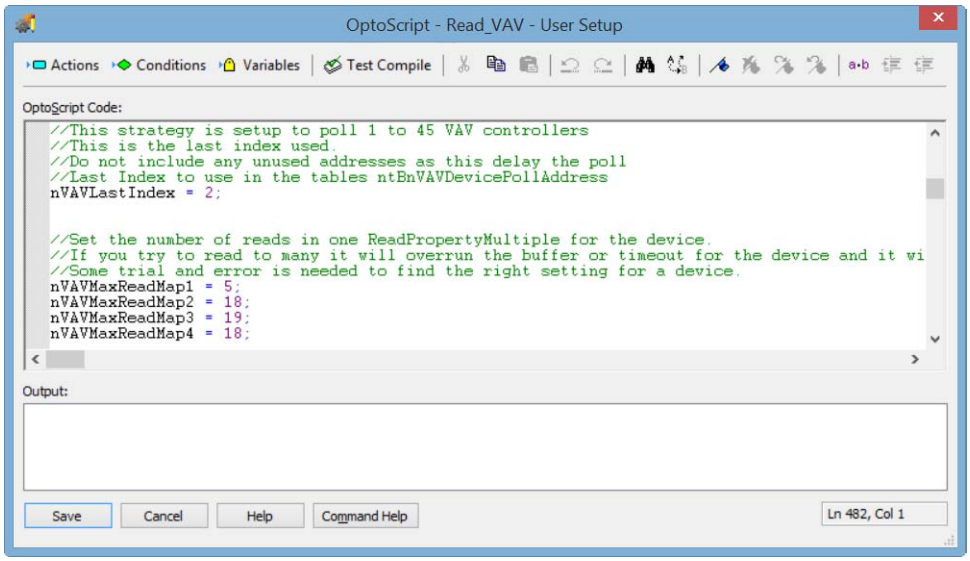
Block 19,
User Setup



Comments







At Startup

The Read_VAV chart is started by the BACnet_Protocol chart.

The chart waits until the Protocol charts is receiving tokens then executes a Who-Is broadcast. It waits 20 seconds for devices to respond then executes a Read Binding Name for each device that's responded to the Who_Is broadcast.

Complete Poll

After Read Binding Name the strategy executes a complete poll of each VAV controller. It also reads the Object Name for each point for each VAV controller. After the complete poll it sets the variable nVAV_Poll to false (0). After the Object Name reads, it sets the index for the address to false in the table ntVAVNameRead.

To execute a complete poll, set the variable nVAV_Poll to true (1).

To execute an Object Name read of a VAV controller, set the index for the address to 1 and set nVAV_Poll to true (1).

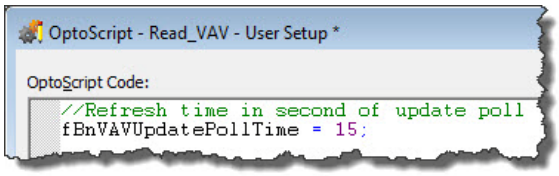
If the following variables are set to true (1, the default setting), the strategy moves the values from the read table to the write table for each address after the complete poll.

- nBnDeviceMap1WriteSync for map 1
- nBnDeviceMap2WriteSync for map 2
- nBnDeviceMap3WriteSync for map 3
- nBnDeviceMap4WriteSync for map 4

After the strategy executes, each of these variables is set to false.

Update Poll

The update poll refresh interval is set in the User Setup block of the Read_VAV chart.



The strategy executes an update poll at the interval that is set (the default is 15 seconds). The data is stored in a float table for each address. To disable update polling, set the variable nVAVUpdatePoll to false (0).

Write to VAV

Write is set up in the User Setup block 19 of the Read_VAV chart. For the location of this block, see "Entering User Setup Parameters" on page 40..

Write is a four-step process as described in the following sections:

"Step 1" on page 44

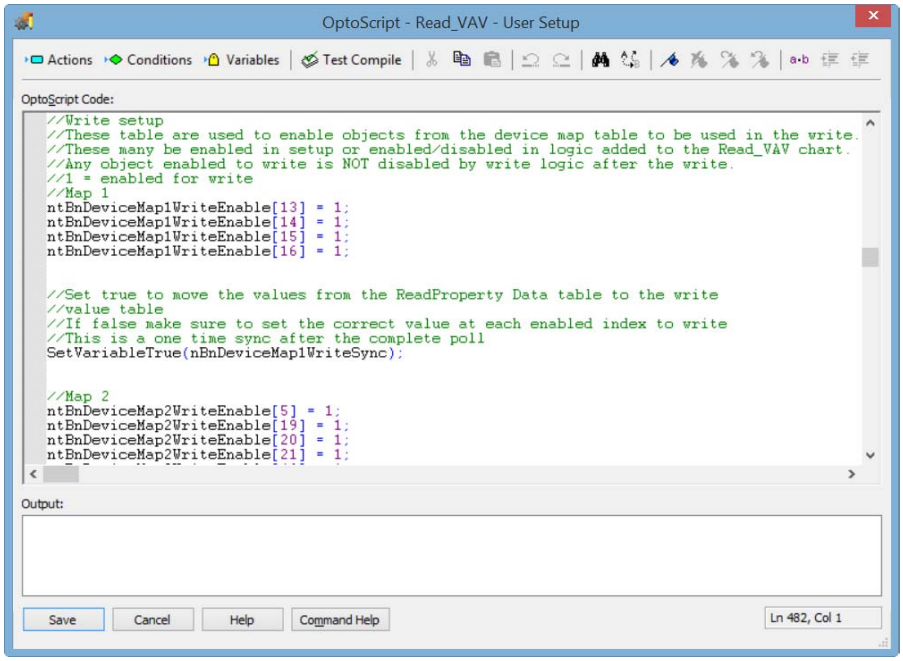
"Step 2" on page 44

"Step 3" on page 45

"Step 4" on page 45

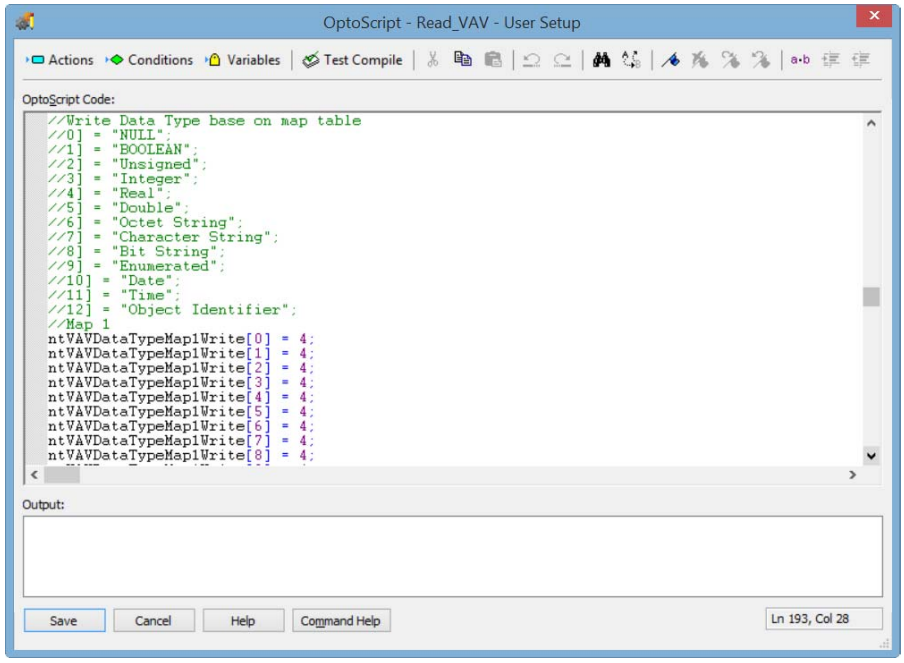
Step 1

There is a write enable table for each device map. When Write is activated it will write to each object that is enabled for each enabled address.



Step 2

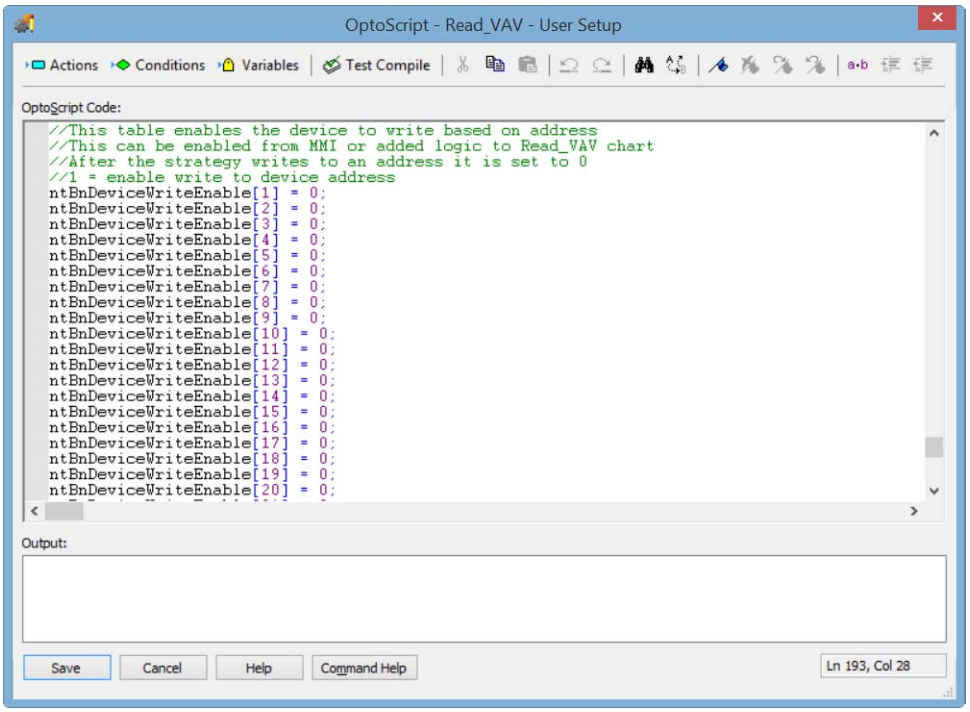
For each device map set the data type for each object to write.



Step 3

Enable the address or addresses of the devices to write.

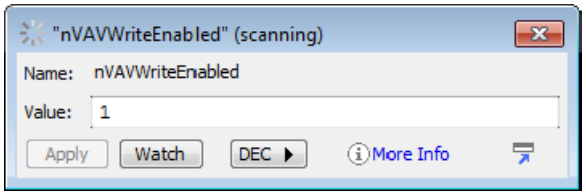
The table index is set to 0 after writing to that address by the strategy.



Step 4

To write to the VAV controllers, set the variable nVAVWriteEnabled to true (1).

After the write is complete, the strategy sets this variable to false.

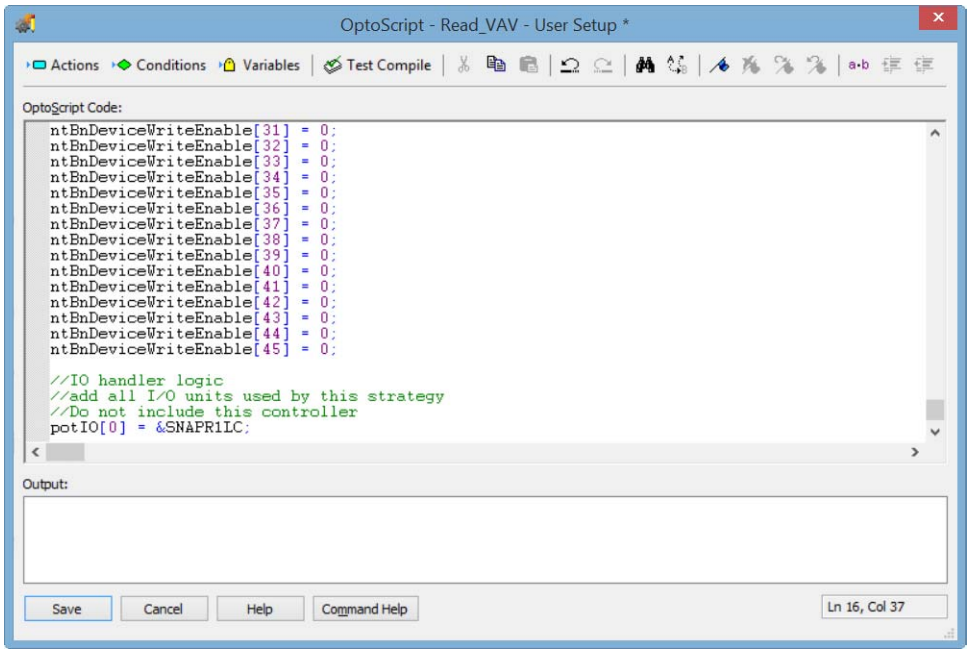


IO Handler

I/O management and error logging is handled by the IO Handler block 420 on the Read_VAV chart.

Each I/O unit configured is entered in the pointer table (potIO). Do not include the controller running this strategy.

Block 420 will check the status of each I/O unit and if needed it will enable communication to the I/O unit. It will also remove errors from the controller's message queue and log the last 100 in the string table stErrorQueue.



4: PAC Display Test Mode

Test Mode logic and a Test Mode PAC Display project are included to help test and troubleshoot BACnet devices on the network.

To run the Test Mode project:

1. Make sure the BACnetInt.idb strategy is running on your control engine.
2. Open the Test Mode project, ProjectTestMode.UUI, in PAC Display Configurator.
3. Select File > Save Project and Load Runtime.

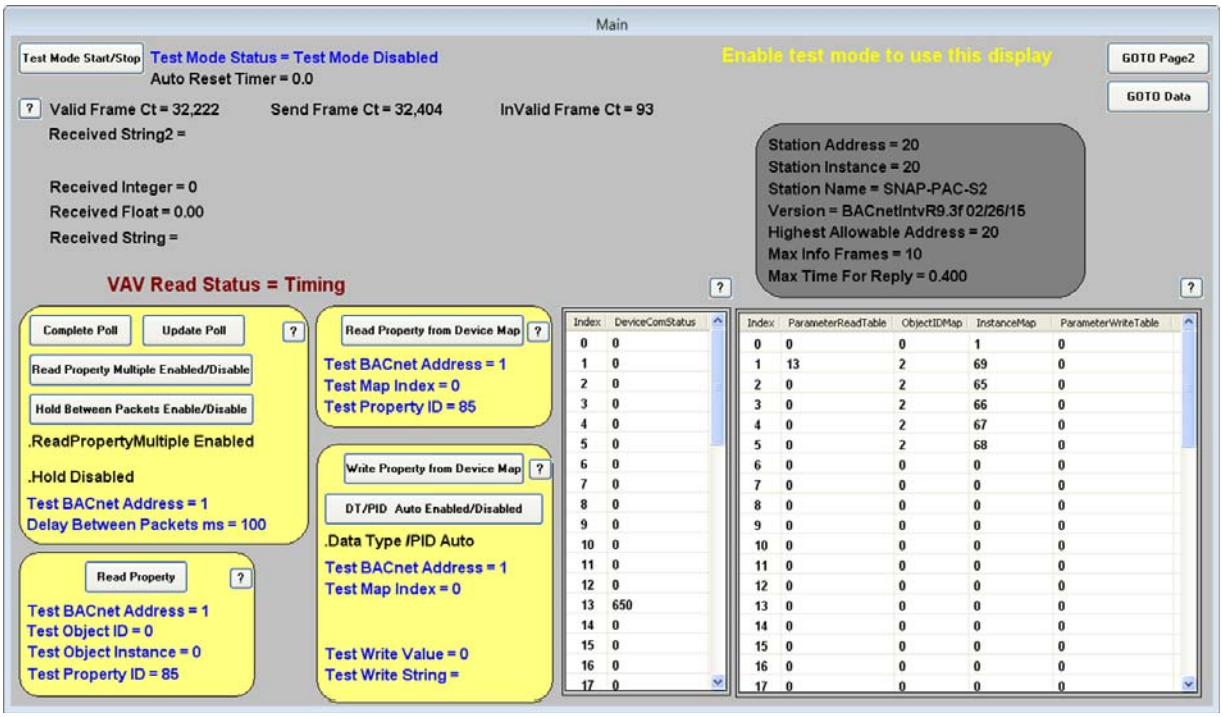
   The project's Main window opens.

Polling is disabled while in test mode. A timer will disable test mode after 600 seconds of inactivity if the operator forgets to disable test mode.

Test mode is unavailable while the strategy goes through its startup and complete poll procedure.

Main Window with Test Mode Disabled

When you launch the PAC Display test project, test mode will be disabled. Some data is available when test mode is disabled. These are variables used by the BACnet_Protocol chart and the Read_VAV chart while polling.



Valid Frame Ct—Frames received by controller. This count should be increasing several per second. If the count is not increasing there is a BACnet network problem.

Send Frame Ct—Number of frames sent from controller.

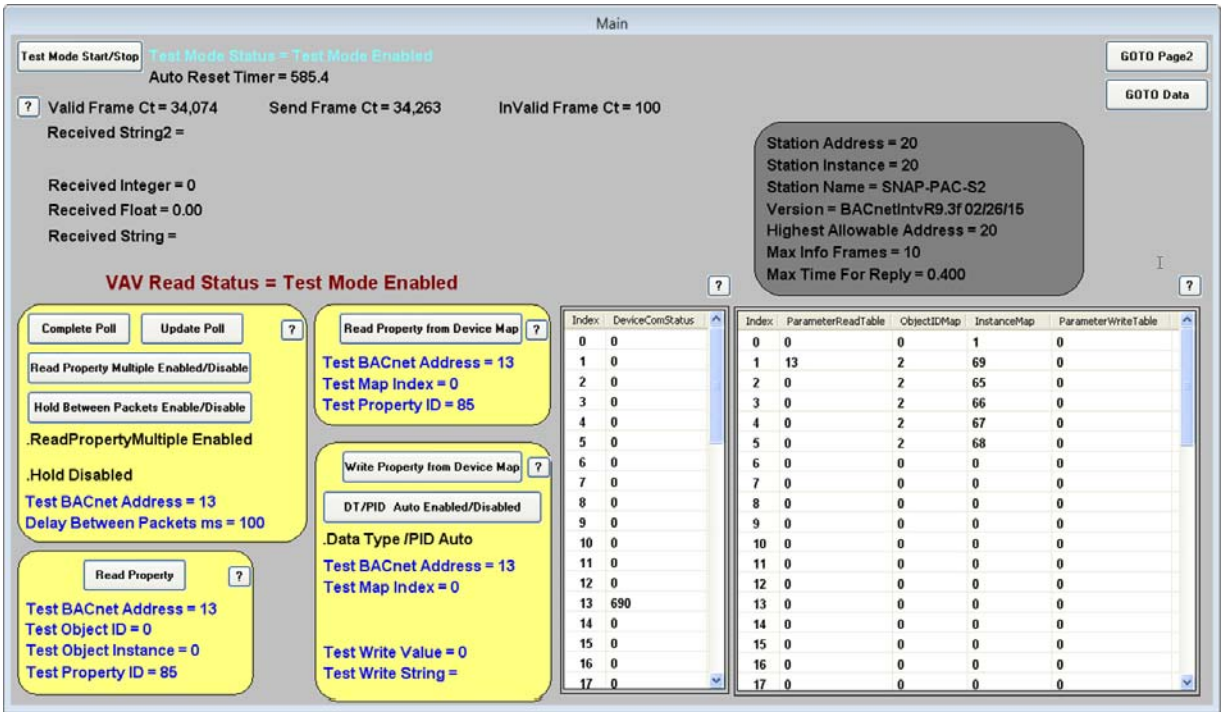
Invalid Frame Ct—Received frames that were bad.

DeviceComStatus Table—Consecutive received packets with no error per address. It will reset to 0 if a bad packet or if there is a timeout.

The columns under ParameterReadTable, ObjectIDMap, InstanceMap, and ParameterWriteTable are used by the subroutines the Read_VAV chart use. Click on [?] icon above the tables for more details.

Main Window with Test Mode Enabled

To start test mode, click Test Mode Start/Stop.



Auto Reset Timer—Time remaining before test mode is disable by strategy.

Received String2—Lists error messages from the BACnet device or Timeout if the BACnet device does not respond. It may be blank for some commands.

Received Integer, Received Float, Received String—Data received from the Read Property command.

Update Poll button—Use with Read Property or Read Property Multiple with or without Hold Between Packets. The Hold Between Packets button can help find objects in your device map that do not exist in the BACnet device.

Complete Poll button—Use with Read Property or Read Property Multiple with or without Hold Between Packets. Hold Between Packets can help find objects in you device map that do not exist in the BACnet device

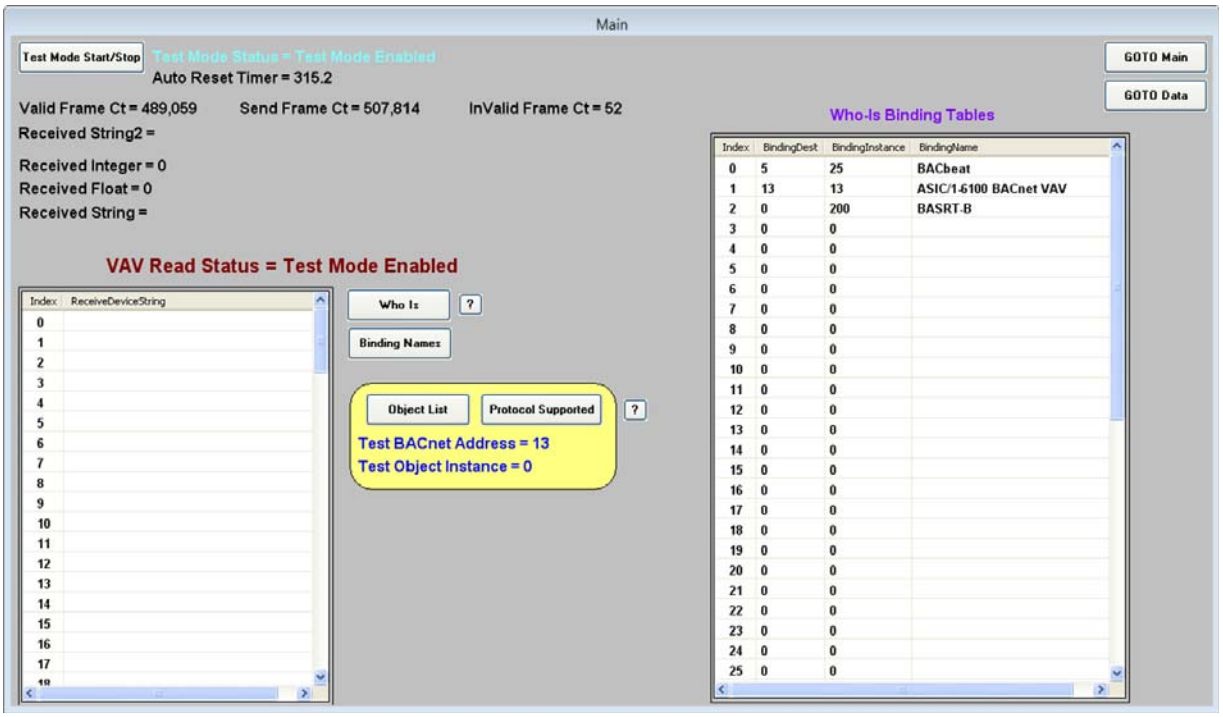
Read Property from Device Map button—Read an object from your device map.

Read Property button—Read any object in a BACnet device.

Write Property from Device Map button—Write a value to an object in your device map. With DT/PID in auto the strategy will get the data type from the data type table in the setup block for each object. In Manual Mode you can enter the data type and property ID manually.

Main Window with Test Mode Enabled, Page 2

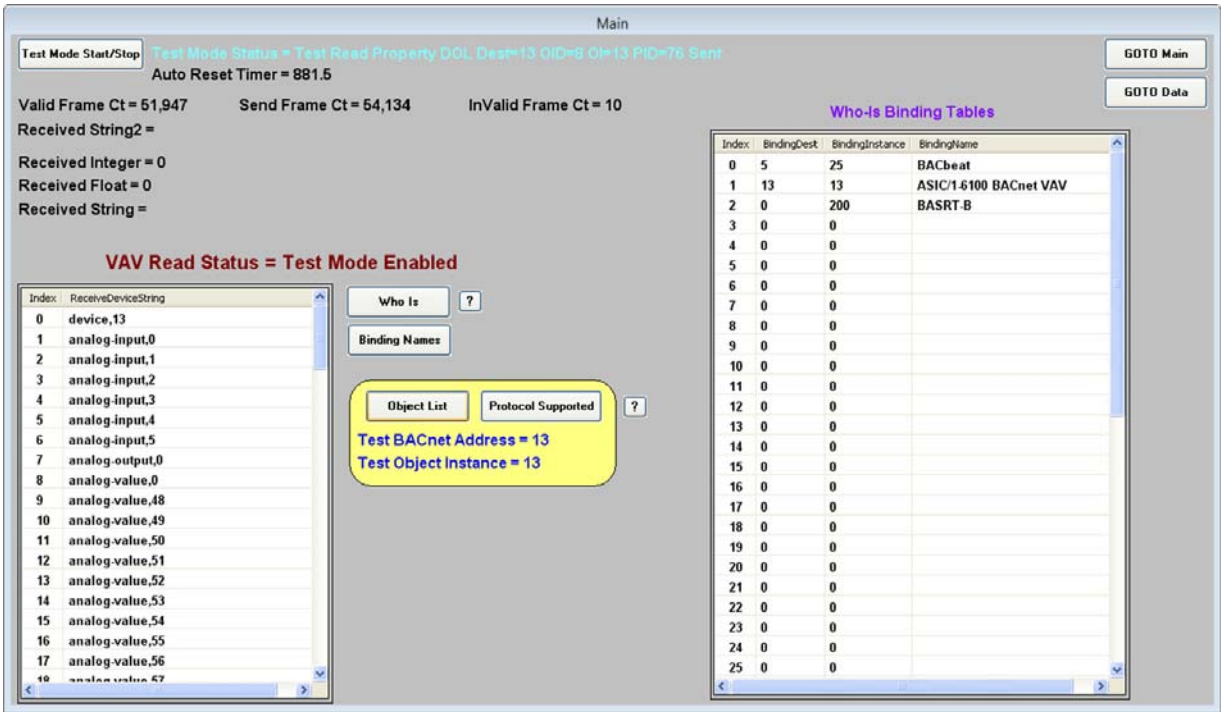
To see page 2, click the GOTO Page2 button.



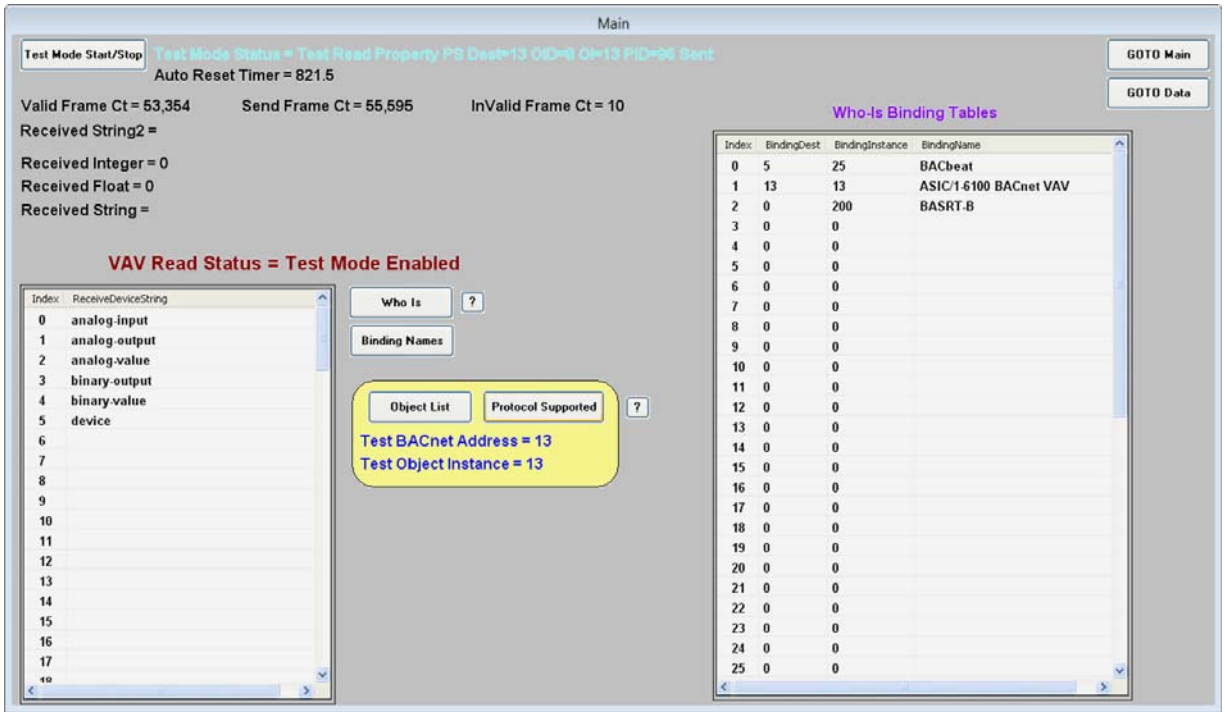
Who-Is button—Broadcast network Who-is. All devices that respond are listed in the binding tables (BindingDest and BindingInstance).

Binding Names button—Get device name for each device that responded to Who-Is broadcast (BindingName).

Object List button—If your device supports this command it will send a list of all its objects and object instances. They will be listed in the ReceiveDeviceString table. The illustration below shows the results of selecting the Object List button in the ReceiveDeviceString table.



Protocol Supported button—If your device supports this command it will send a list of all the object IDs it supports. They will be listed in the ReceiveDeviceString table. The illustration below shows the results of selecting the Protocol Supported button in the table ReceiveDeviceString.



Who_Is Binding Tables

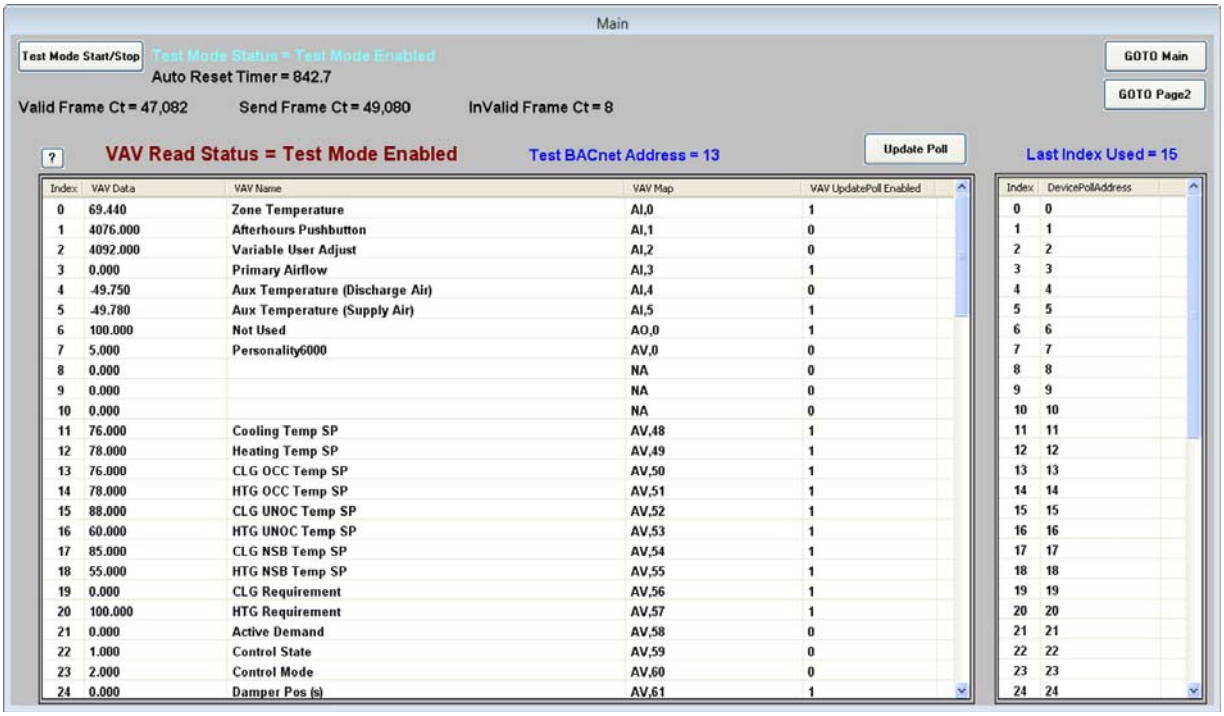
BindingDest—List of BACnet addresses of devices that responded to Who_Is broadcast.

BindingInstance—Instance number for each device that responded to Who_Is broadcast. When using the Object List or Protocol Supported command this is the test object instance you enter.

BindingName—List of device names for each address after using the Binding Names command.

Main Window with Test Mode Enabled, Data Page

To see page 2, click the GOTO Data button.



Index	VAV Data	VAV Name	VAV Map	VAV UpdatePoll Enabled
0	69.440	Zone Temperature	AI,0	1
1	4076.000	Afterhours Pushbutton	AI,1	0
2	4092.000	Variable User Adjust	AI,2	0
3	0.000	Primary Airflow	AI,3	1
4	-49.750	Aux Temperature (Discharge Air)	AI,4	0
5	-49.780	Aux Temperature (Supply Air)	AI,5	1
6	100.000	Not Used	AO,0	1
7	5.000	Personality6000	AV,0	0
8	0.000		NA	0
9	0.000		NA	0
10	0.000		NA	0
11	76.000	Cooling Temp SP	AV,48	1
12	78.000	Heating Temp SP	AV,49	1
13	76.000	CLG OCC Temp SP	AV,50	1
14	78.000	HTG OCC Temp SP	AV,51	1
15	88.000	CLG UNOC Temp SP	AV,52	1
16	60.000	HTG UNOC Temp SP	AV,53	1
17	85.000	CLG NSB Temp SP	AV,54	1
18	55.000	HTG NSB Temp SP	AV,55	1
19	0.000	CLG Requirement	AV,56	1
20	100.000	HTG Requirement	AV,57	1
21	0.000	Active Demand	AV,58	0
22	1.000	Control State	AV,59	0
23	2.000	Control Mode	AV,60	0
24	0.000	Damper Pos (s)	AV,61	1

Index	DevicePollAddress
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24

Test BACnet Address—Select the address of the device to view its data.

Update Poll button—Execute a refresh poll.

Last Index Used—Last index to use in the DevicePollAddress table. Index 0 not used

Data Tables

VAV Data—Current data in the strategy read table for the selected address.

VAV Name—Names for each object (part of complete poll) for selected address.

VAV Map—Object map for the selected address.

VAV UpdatePollEnabled—Only objects with 1 (enabled) are included in the refresh poll. This data can be edited.

DevicePollAddress—Polling order by address table. It will poll addresses to the Last Index Used setting. Index 0 not used. This data can be edited.

A: BACnet PIC Statement

The BACnet Protocol Implementation Conformance (PIC) Statement is as follows:

Date: 5/26/2010

Vendor Name: Opto 22

Product Name: BACnet MS/TP Integration Kit

Application Software Version: 8.2e

BACnet Protocol revision: 9

Product Description

This will allow the PAC controller to act as a BACnet client/server using MS/TP. Connection to the BACnet MS/TP network can be made using an S-series controller's port configured as RS-485.

BACnet Standardized Device Profile (Annex L)

BACnet Application Specific Controller (B-ASC)

BACnet Interoperability Building Blocks (BIBBs) Supported (Annex K)

DS-RP-A (readProperty), DS-RP-B (readProperty), DS-RPM-A (readPropertyMultiple), DS-RPM-B (readPropertyMultiple), DS-WP-A (WriteProperty), DS-WP-B (WriteProperty), DS-WPM-B (WritePropertyMultiple),

DS-COV-A (COV), DS-COVU-A (COV Unsolicited)

DM-TM-B (TextMessage), DM-TS-B (TimeSynchronization), DM-UTC-B (UTCTimeSynchronization),

DM-RD-B (ReinitializeDevice), DM-DDB-A (DynamicDeviceBinding), DM-DDB-B (DynamicDeviceBinding),

DM-DOB-B (DynamicObjectBinding)

Segmentation Capability

None

Standard Object Type Supported

Device Object, Analog Input, analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Multi-state input, Multi-state output, Multi-state value

Data Link Layer Options

MS/TP Master (Clause 9), Baud rate(s): 9.6K, 19.2K, 38.4K, 76.8K, 115.2K

Device Binding Methods

Send who-Is, receive I-Am (DM-DDB-A)

Receive who-Is, send I-Am (DM-DDB-B)

Receive who-Has, send I-Have (DM-DOB-B)

Network Options

None

Character Sets Supported

ANSI X3.4