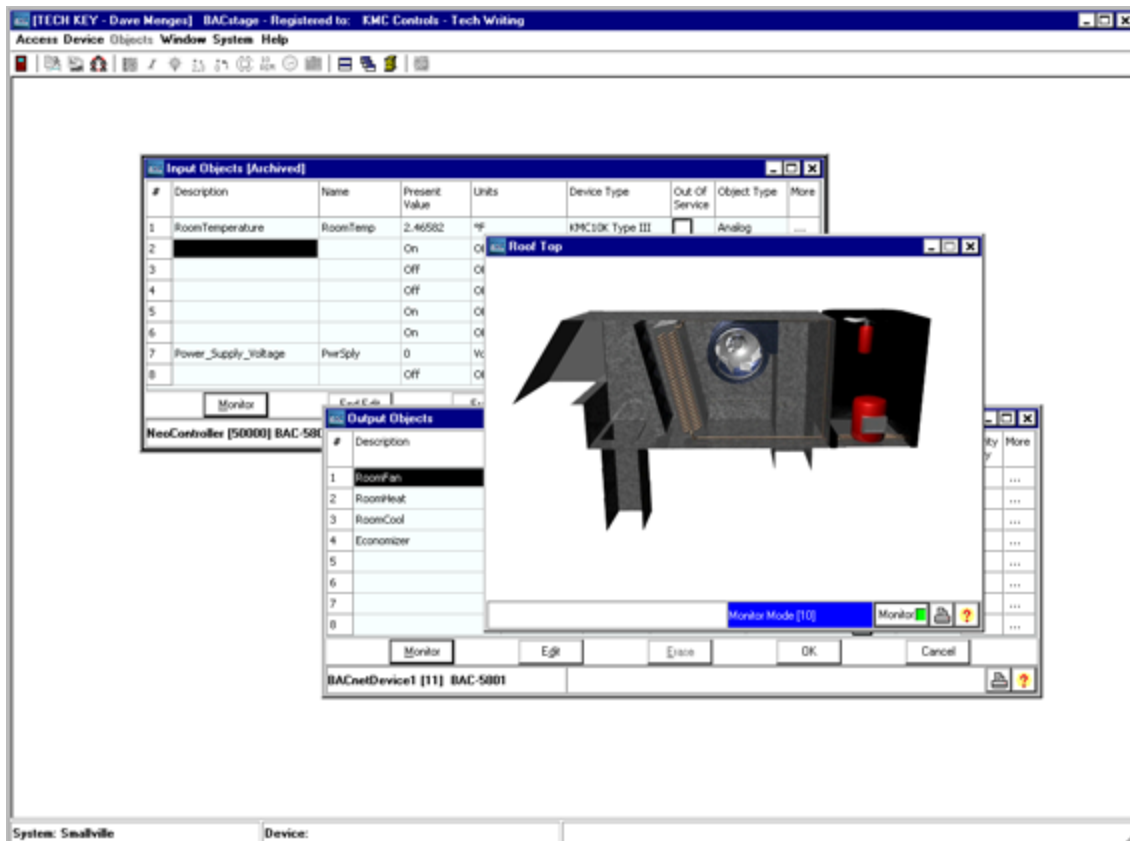


# BAC-5000

## BACstage Operator Workstation



Reference Guide and Operating Instructions

For version 2.4

©2012, KMC Controls, Inc.

WinControl, NetSensor, and the KMC logo are registered trademarks of KMC Controls, Inc.

TotalControl, BACstage, and FullBAC are trademarks of KMC Controls, Inc.

ActiveX, Silverlight, Microsoft Excel, Windows, and Windows Vista are registered trademarks of Microsoft, Inc.

All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means without the written permission of KMC Controls, Inc.

Printed in U.S.A.

Disclaimer

The material in this manual is for information purposes only. The contents and the product it describes are subject to change without notice. KMC Controls, Inc. makes no representations or warranties with respect to this manual. In no event shall KMC Controls, Inc. be liable for any damages, direct or incidental, arising out of or related to the use of this manual.

**KMC Controls, Inc.**

P.O. Box 497

19476 Industrial Drive

New Paris, IN 46553

U.S.A.

TEL: 1.574.831.5250

FAX: 1.574.831.5252

E-mail: [info@kmccontrols.com](mailto:info@kmccontrols.com)

Contents .....	3
Section 1: Using BACstage .....	7
Hardware and operating system requirements .....	7
What's new in this version .....	8
If you encounter difficulty .....	8
Configuring a controller for the network .....	9
Section 2: The Access menu .....	13
System List .....	14
Connection Parameters .....	15
Global Settings .....	18
Sign Off .....	19
Exit .....	20
Using the Simulator .....	20
Establishing point-to-point communications .....	20
Section 3: The Device Menu .....	23
Device List .....	24
Device Tables .....	25
Basic Tables .....	26
Configuring the NetSensor .....	27
Static Binding .....	34
Reinitialize Device .....	35
Disable/Enable device .....	36
Send Update Notification .....	36
Flush File Cache .....	37
Alarm/Event Summary .....	37
Backup Device .....	38
Restore Device .....	38
Backup Files .....	39
Restore Files .....	40
Auto Addressing .....	40
Section 4: The Objects menu .....	41
Device Objects .....	42
Input objects .....	45
Working with accumulator objects .....	51
Output objects .....	53
Analog value objects .....	59
Binary value objects .....	62
Multistate value objects .....	64
Basic Programs .....	67

Loop objects .....	69
Schedule object .....	72
Calendar object .....	76
Notification object .....	77
Event enrollment object .....	79
Working with alarms and events .....	83
Trend object .....	88
Viewing trend logs .....	90
About objects, properties and services .....	91
Priority arrays .....	92
<b>Section 5: The Window menu .....</b>	<b>95</b>
<b>Section 6: About Control Basic programs .....</b>	<b>97</b>
Using the Control Basic editor .....	97
About Control Basic scans .....	100
Writing Control Basic statements .....	101
Labels and line numbers .....	102
Programming with names .....	103
Programming format and notation .....	105
Real numbers .....	105
Hierarchy of operators .....	105
Using arithmetic operators .....	106
Using Boolean logic .....	107
Relational operators .....	107
Programming with variables .....	108
Transferring values between BACnet controllers .....	109
Programming with mnemonics .....	110
<b>Section 7: Keywords for Control Basic .....</b>	<b>113</b>
Using example programs from help .....	113
Syntax for commands and functions .....	113
<b>Section 8: The System Menu .....</b>	<b>151</b>
Password Manager .....	152
Settings .....	155
Group Displays .....	157
Creating a group display .....	158
Adding and modifying object properties .....	160
Adding and modifying links to group displays .....	163
Changing a present value from a group display .....	165
Objects List .....	166
Network Backup .....	167
Network Restore .....	168
Send Time .....	169
Alarms/Events .....	170

Custom Alarm Messages .....	172
BACnet Read/Write Property .....	173
Section 9: The Help menu .....	175
About BACstage .....	175
Contents .....	176
Appendix A: Supported engineering units .....	177
Appendix B: BACstage job file location .....	179
Appendix C: Reference to KMC BACnet controllers .....	181
General purpose controllers .....	181
Job specific controllers .....	182
Appendix D: Timekeeping .....	185
Appendix E: Next Generation Control Basic .....	189
Control Basic versions in controllers .....	190
Changes to IF THEN .....	190
Deprecated keywords .....	191
File compatibility .....	192
Keywords for new functions and statements .....	193
Labels in Next Generation Control Basic .....	194
Line numbers .....	195
Local variables .....	195
References to objects in remote devices .....	195
Using the conversion tool .....	196
Appendix F: BACdoor OEM Client .....	197
Opening BACdoor .....	197
Settings in BACdoor OEM client .....	199
Installing drivers for BACnet 8802-3 (Ehternet) .....	203
Appendix G: Glossary .....	207
Index .....	217



## Section 1: Using BACstage

This section includes topics to help you get started using BACstage.

---

Topics for using BACstage include:

- ◆ [Configuring a controller for the network on page 9](#)
- ◆ [BACdoor OEM Client on page 197](#)
- ◆ [Settings in BACdoor OEM client on page 199](#)
- ◆ [BACstage job file location on page 179](#)

### Hardware and operating system requirements

To run BACstage, you will need a computer that meets the following minimum requirements:

Table 1–1 Computer system requirements

Component	Windows 2000 Windows XP	Vista Business Vista Enterprise	Windows 8 Professional Windows 7 Professional Windows 7 Ultimate
Processor speed	300 MHz or faster	2 GHz or faster	2 GHz or faster
RAM memory	512 megabytes RAM or greater	3 GB or greater	2 GB or greater
Hard disk space	100 megabytes of hard drive space available after installation	100 megabytes of hard drive space available after installation	100 megabytes of hard drive space available after installation
Monitor	SVGA with minimum 800 x 600 resolution.	SVGA with minimum 800 x 600 resolution.	SVGA with minimum 800 x 600 resolution. DirectX 9 graphics processor
Network connection	Ethernet 10BaseT connection	Ethernet 10BaseT connection	Ethernet 10BaseT connection

**Computer system requirements (continued)**

MS/TP connection	Serial or USB port with KMD–5579 or third party EIA–485 converter.	Serial or USB port with KMD–5579 or third party EIA–485 converter.	Serial or USB port with KMD–5579 or third party EIA–485 converter.
License key	USB port dedicated to hardware key	USB port dedicated to hardware key	USB port dedicated to hardware key
Sound output and speakers	Required for audible alarm notification	Required for audible alarm notification	Required for audible alarm notification

In addition connecting the computer to the LAN with the Ethernet connection, you will need also one of the following:

- ◆ A KMD–5576 USB to RS–485 converter.
- ◆ A third party RS–232 to RS–485 converter.
- ◆ A third party USB to RS–485 converter.

**What’s new in this version**

For a list of new features and changes to the program, see the file [WhatsNew.pdf](#) included on the installation USB flash drive. The new features and BACstage history are available also on the KMC Controls partners web site.

**If you encounter difficulty**

If you experience difficulty with BACstage, KMC Controls provides the following assistance.

**Printed version of help** An Adobe Acrobat version of BACstage help is included on the BACstage installation USB flash drive. The document *Operating BACstage* is identical to the on-line help but, it is formatted to print as a reference manual.

**The KMC Controls web site** Navigate to the support section on the KMC Controls partner web site for the latest information for BACstage and other KMC Controls BACnet products.

[partners.kmcontrols.com](http://partners.kmcontrols.com)

**KMC technical support** Our distribution partners have unlimited and free access to our team of technical support representatives. We provide coast-to-coast, and toll-free, support from 8 AM to 5 PM.

Toll-Free Technical Support: 866.303.4562



## Configuring a controller for the network

Use BACstage to configure a controller with the following properties before connecting it to a network.

- ◆ Device instance
- ◆ MAC address
- ◆ Baud rate.



Connecting a controller to a network with a MAC address or device instance number that duplicates an existing MAC address or device instance numbers will result in poor or disrupted network traffic.

---

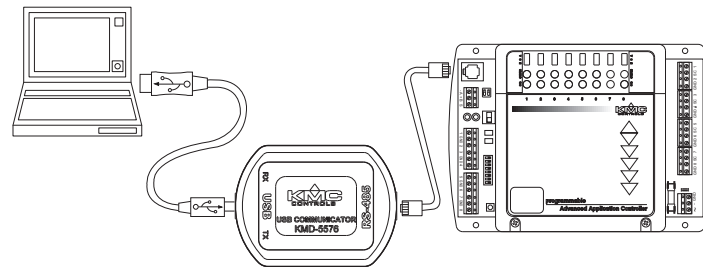
### Setting up for configuration

To configure a controller with BACstage, you will need a direct MS/TP connection between the computer and the controller you are configuring. You will also need to be familiar with System List, *Device List* and *Device Object* menus in BACstage.

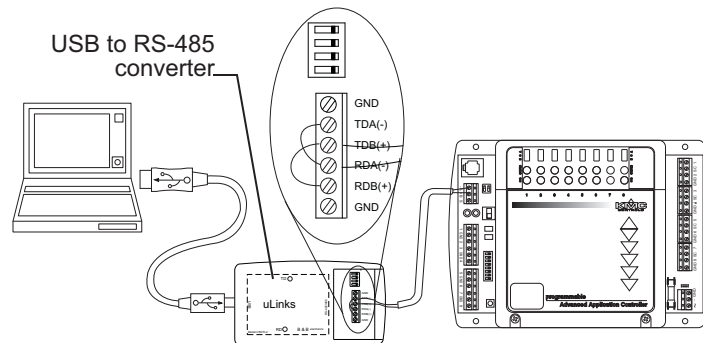
- ◆ [System List on page 14](#)
- ◆ [Device List on page 24](#)
- ◆ [Device Objects on page 42](#)

To connect to a BACnet network or controller, do the following:

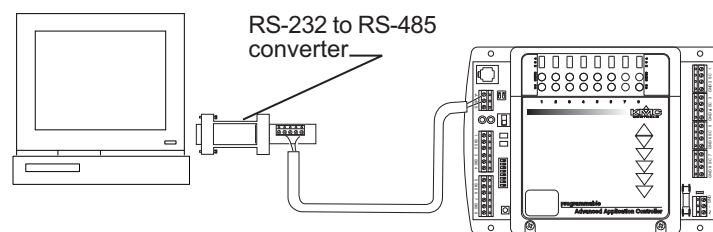
1. Connect the controller to a computer running BACstage. This is usually done with a direct MS/TP connection. See the illustration [Direct RS-232 and MS/TP network connections on page 10](#).
2. If BACstage is not running, start it.
3. Verify or set BACstage to a Device Instance and MAC address that will not be used by the controller connected for configuration. See [Settings in BACdoor OEM client on page 199](#).
4. Set the baud rate to the same rate as the controller connected for configuration. For new KMC Controls BACnet controllers this is 38,400.

**Illustration 1–1 Direct RS-232 and MS/TP network connections**

KMD-5576 connection



B&amp;B Electronics USB connection



B&amp;B Electronics RS-232 connection

**Configure the controller**

Use the following steps for each controller to configure. It may be useful to add a system name to the system list that is used only for configuring controllers.

1. Open the *System List* from the *Access* menu.
2. Choose the system to which the controller is connected. The *Device List* opens.
3. Choose the device for configuration from the device list.
4. Choose *Device Object* from the *Objects* menu.
5. Click *Edit* and enter values specified in the system plans for the properties listed in the table [Configuration properties on page 11](#).

**Table 1-2 Configuration properties**

<b>Property</b>	
MAC address	Required during configuration
Baud	Required during configuration
Device Instance	Required during configuration
Description	May be changed across a network.
Name	May be changed across a network.
Location	May be changed across a network.

1. Click *End Edit* and then *Close*.
2. Cycle the power for the controller or choose *Reinitialize Device* from the *Device* menu to make the changes effective.



## Section 2: The Access menu

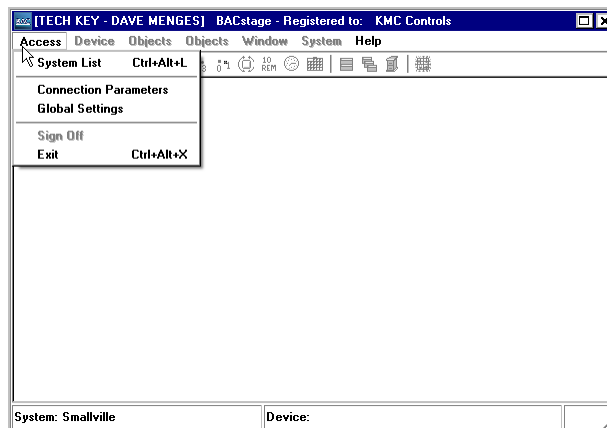
Use the Access Menu to establish a communication link between BACstage and a BACnet internetwork.

---

Select the following from the Access Menu.

- ◆ *System List* on page 14
- ◆ *Connection Parameters* on page 15
- ◆ *Global Settings* on page 18
- ◆ *Sign Off* on page 19
- ◆ *Exit* on page 20

### Illustration 2–1 Access Menu



## System List

Select *System List* to connect a computer running BACstage to an internetwork or controller. *System List* is an address book in which information is recorded about each individual job-site. BACstage uses the system list information two ways.

- ◆ Selecting a BACnet internetwork and the network media through which BACstage will connect.
- ◆ Automatically opening the correct information stored in the job folder.

### Related topics

- ◆ [BACdoor OEM Client on page 197](#)
- ◆ [Device List on page 24](#)
- ◆ [Using the Simulator on page 20](#)
- ◆ [Establishing point-to-point communications on page 20](#)

### Illustration 2–2 System List

#	System Name	LAN Type	Phone Number
1	School System	MS/TP	
2	Elementary School	MS/TP	
3	Middle School	BACnet IP	
4	Admin Building	Simulator	
5	Field House	TP (Dial Out)	
6		Simulator	
7		Simulator	
8		Simulator	
9		Simulator	
10		Simulator	
11		Simulator	
12		Simulator	
13		Simulator	
14		Simulator	
15		Simulator	
16		Simulator	

**#(System number)** Click the system number to choose the system. *System List* closes and the device list opens. See [Device List on page 24](#).

**System** Enter the system name. BACstage creates a job folder with the same name in which all information associated with this system name is stored. Up to 256 sites may be entered in the system list.

**LAN type** Choose the network protocol by which the computer running BACstage will connect to the internetwork. The LAN types are:

- ◆ MS/TP
- ◆ BACnet 8802.3 Ethernet
- ◆ BACnet IP
- ◆ Point-to-point—See *Establishing point-to-point communications* on page 20 for instructions for setting up a point-to-point connection.
- ◆ Simulator—See *Using the Simulator* on page 20 for additional information about the BACstage simulator.
- ◆ Point-to-point (Dial Out)—See *Establishing point-to-point communications* on page 20 for instructions for setting up a point-to-point connection.

**Phone Number** Enter the telephone number to use for contacting a remote BACnet internetwork. For a list of characters used for dialing, see the table *Dialing options*. The topic *Establishing point-to-point communications* on page 20 fully describes connecting to a remote internetwork with BACstage.

**Table 2-1 Dialing options**

Symbol	Action
Dash (-)	The dash (-) is optional. Use it to make the telephone phone number easier to understand.
Period (.)	Example: 555-1212.31 Place a period (.) in the dial string to pause dialing until you click the telephone on-hook switch to continue dialing. Use when dialing automatic answering systems which require entering an extension number.
Comma (,)	Example: ,9,555-1212 The comma (,) creates a pause in the dialing sequence. The pause time, typically 2 seconds for each comma, is set by the modem setup registers. A typical use for the comma is to provide a delay after dialing an access number for an outside telephone line.

## Connection Parameters

Use the settings in the Connection Parameters dialog to configure the network connection between BACstage and the BACnet internetwork.

See the topic *Settings in BACdoor OEM client* on page 199 for additional connection properties.

**Illustration 2–3 Connection Parameters dialog**

**Location of BACDOC.INI file** Displays the location for the file BACDOC . INI. The file stores the connection settings for BACstage.

**Our Peer Name** A 16-character name for the BACstage operator workstation and must be unique among all devices on the internetwork. The set of characters used in *Our Peer Name* is restricted to printable characters.

**Our Device Instance** A number that uniquely identifies BACstage as a BACnet device on the internetwork. The device instance number is assigned by the BACnet system designer. Valid instance number's range from 0 to 4,194,303. It is by reference to the device instance number that data is exchanged between BACnet devices.

Use one of the following methods to set the device instance:

- ◆ Click *Randomize*. BACstage chooses a random number between 0 and 4,194,303.
- ◆ Enter a number between 0 to 4,194,303 (inclusive).

**Vendor ID** Displays the BACnet vender name and identification number.

**Accept Private Transfers** Disables BACnet unconfirmed private transfers between BACstage and controllers on the internetwork to which BACstage is connected.



**Caution**

Clearing the Accept Private Transfers check box will disable functions such as the auto addressing and Flush File Cache. Do not clear this check box unless instructed by technical support at KMC Controls.



**WhoIs Interval** Sets the interval between automatic Who Is? broadcast messages.

### Point-To-Point settings

Selects and sets up the communication port for Point To Point (PTP) connections to a remote half-router.

**Port** Sets the serial port that BACstage will use for a PTP link to a half-router. Only valid ports are in the list.

**Baud** Sets the communication speed for a PTP link to a half-router. Valid rates are 9600, 19,200, 38,400 and 76,800 baud.

**Dialed** Select the Dialed check box to use a modem to establish a PTP link.

### MS/TP settings

Use these settings when connecting BACstage to an internetwork with an MS/TP connection.

**Port** Sets the serial port with which BACstage uses to connect to an MS/TP network. Only valid ports are in the list.

**Baud** Sets the communication speed between the MS/TP network and the computer on which BACstage is running. Valid rates are 9600, 19,200, 38,400 and 76,800 baud.

**MAC** This number assigns to BACstage a node number on the MS/TP network to which it is connected. The number must be unique on the local network but, may be duplicated on remote MS/TP networks. This is equivalent to *TS (MS/TP Node)* in the BACdoor configuration menu.

**Max Master** Indicates the highest Media Access Control (MAC) address assigned to any device on the MS/TP network to which the device is connected.

- ◆ Setting *Max Master* significantly higher than the highest numbered device may result in increased polling and slower response times.
- ◆ Setting *Max Master* lower than the highest numbered device will result in devices not appearing on the network.

**Max Info Frames** Sets the maximum number of packets that BACstage sends before passing the token.

### BACnet Internet protocol settings

Use the BACnet/IP Parameter configure BACstage for the IP network to which it is connected.

**Hex or Decimal** Select the format for entering the port number.

**Port** Sets the BACnet UDP port number which is supplied by the network system administrator.

- ◆ The port must match the port in use by BACnet devices on the network to which BACstage is connecting. Valid port numbers are 0xBAC0 in hexadecimal notation (47808 in decimal notation) to 0xBAC9 (47817).
- ◆ When registered as a foreign device, enter the port number of the BBMD. If port address translation (PAT) is used between the local router and the PAD or BBMD, contact the network system administrator for the correct public IP address.

**Subnet Mask** The IP subnet to which the computer running BACstage and BACdoor is connected.

**Register as Foreign Device with BBMD** Select this check box to connect to a BACnet Broadcast Management Device (BBMD). Foreign device registration to a BBMD is a technique for crossing an IP-only network router with BACnet broadcast messages.

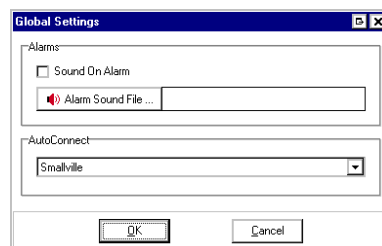
**BBMD IP Address** Sets the Bacnet Broadcast Management Device to which BACstage will connect as a foreign device. Additional properties are available in the topic [Settings in BACdoor OEM client on page 199](#).

**Registration Time-to-Live** (For Foreign Device connections only) Sets the interval at which BACstage sends a registration message to the BBMD with which it is registered.

## Global Settings

Use the Global Settings dialog to apply universal settings to BACstage operation.

### Illustration 2–4 Global settings



### Alarms

Set the alert sound for alarm or event notifications received by BACstage in the Global Settings dialog. To receive an alarm or event, the computer on which BACstage is running must be connected to the BACnet internetwork. A sound card is required.

**Sound On Alarm** When selected, the selected sound file will play through the computer computers upon receipt of an alarm or event. Choose from the following types of sound files:

- ◆ WAV
- ◆ AVI
- ◆ MP3

**Alarm Sound File** Click to select the sound file for the alarm sound.

Upon receipt of a notification of an alarm or event, BACstage will play the sound until it is acknowledged.

- ◆ To view, acknowledge or erase alarms, see [Alarms/Events on page 170](#).
- ◆ To view alarms in a device, see [Alarm/Event Summary on page 37](#).

### AutoConnect

When the BACstage is launched, the program automatically connects to the system in the **AutoConnect** text box and waits for a valid operator name and password.

See also [System List on page 14](#) and [Password Manager on page 152](#).

## Sign Off

Use *Sign Off* to change operators without disconnecting from the internetwork. Another operator can then sign on and operate or modify the system as permissions allow. BACstage remains connected to the internetwork with the System Sign-On dialog open. The modem does not hang-up.

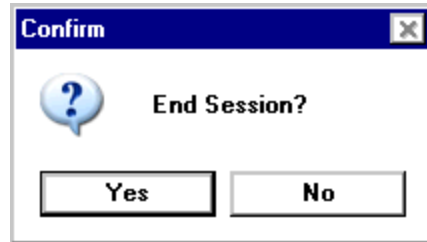
**Illustration 2-5 System Sign On dialog**

The screenshot shows a dialog box titled 'System Sign On'. It contains the following elements:

- User Name:** A dropdown menu with 'Dave' selected.
- Password:** A text field containing '#####'.
- System:** A text field.
- Buttons:** 'Sign In' and 'Cancel'.
- Logo:** The 'BACstage' logo is displayed in the bottom left corner.

**Exit**

Closes BACstage. Before BACstage closes, a confirmation dialog opens.

**Illustration 2–6 Exit confirmation dialog**

Click **Yes** to exit and **No** to return to the system.

**Using the Simulator**

BACstage supports off-line programming with a simulator mode. When using the simulator for off-line programming, the following constraints are in place.

- ◆ Control Basic programs may be compiled and saved but cannot run.
- ◆ Use the device list to build a simulated internetwork. See [Device List on page 24](#).
- ◆ Animated graphics reflect the state of the object property to which they are associated. For example, if an animated fan is associated with a binary output, the fan will rotate when the present value of the output is *On* and will not rotate when the present value is *Off*.

**Note:**

Only the master administrator may change a simulation connection to a live connection.

**Establishing point-to-point communications**

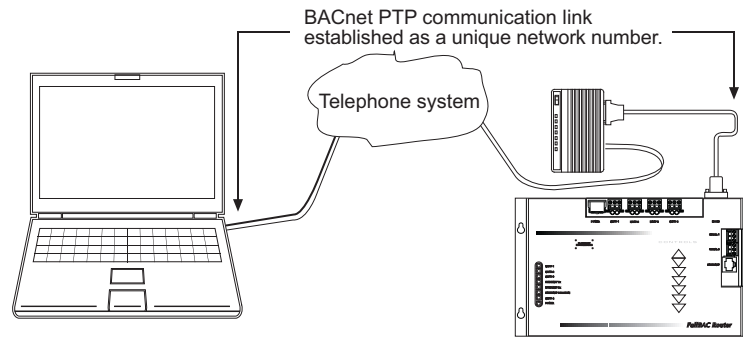
A BACnet Point-to-Point (PTP) link is established only between a pair of BACnet half-routers. PTP connections typically use a serial port connection and may be established between the following:

- ◆ A half-router on an internetwork to another half-router on the internetwork
- ◆ A half-router to a device designed for half-router connection.
- ◆ An operator workstation and a half-router on an internetwork.

BACstage uses PTP and a modem connection to connect to a remote internetwork.

- ◆ BACstage—through the BACdoor client—performs the half-router function on the computer.
- ◆ The BAC-5050 includes half-router functions for connection to an internetwork
- ◆ Use Serial Port 1 for a direct serial connection.
- ◆ Use the modem connector for a dial-up connection.

**Illustration 2-7 BACstage dial-up PTP connection**



### Configuring BACstage with BACdoor

See *Settings in BACdoor OEM client* for details on configuring the BACdoor OEM client. Use the following procedure to configure BACstage and BACdoor for PTP operation:

1. Assign to BACstage a unique device number for the internetwork to which it is connecting. The device instance is assigned in the *Our Device Instance* text box in BACdoor.
2. Assign to the point-to-point link a unique BACnet network number on the internet work. The network number is assigned in the *SNET* text box in BACdoor.
3. Select one of the following connection methods:
  - For direct serial connection between BACstage and a half-router, clear *Dialed* in BACdoor and enter a serial port configuration in the *Non-Dialed Com Port(reset)* text box.
  - For a modem connection, select *Dialed* in the *PTP Parameters* area of BACdoor.
4. If applicable, enter the telephone number of the remote modem in the *Phone Number* column of the system list.
5. Select *PTP (Dial Out)* as the *LAN Type*.
6. When configured, select a system by clicking the system number from the system list.

### Configuring a BAC-5050 router for PTP communications

The router must be enabled and configured for modem communications. For the BAC-5050 use *Router Configuration Tool* supplied with the router.

- ◆ All network rules must be observed. This includes avoiding duplicate network numbers and loops.
- ◆ The PTP network number is assigned in BACdoor. As with all network numbers, it must be unique on the internetwork.
- ◆ KMC Controls recommends using a KMD-5569 modem for the remote modem.

### For a direct serial connection between a computer and a BAC-5050

1. Connect a serial cable between the computer serial port and serial port 1 on the BAC-5050.
2. Open *Router Configuration Tool* to configure the router.
3. In the Router Configuration Tool PTP Setting tab, select PTP
4. In the *General Settings* tab, set *Serial 1 Baud Rate* to the baud rate of the link. This baud rate must match the baud set in *Non-Dialed Com Port (reset)* text box of BACdoor.
5. Close *Router Configuration Tool*.

### Configuring the BAC-5050 for a PTP modem connection

1. Connect a modem to the BAC-5050 modem port.
2. Open *Router Configuration Tool* to configure the router.
3. In the Router Configuration Tool PTP Setting tab, select Modem.
4. In the *General Settings* tab, set the baud rate of the link in Serial 1. This is the baud rate between the router and the modem and must be set higher than the highest expected modem-to-modem rate.
5. Close *Router Configuration Tool*.

## Section 3: The Device Menu

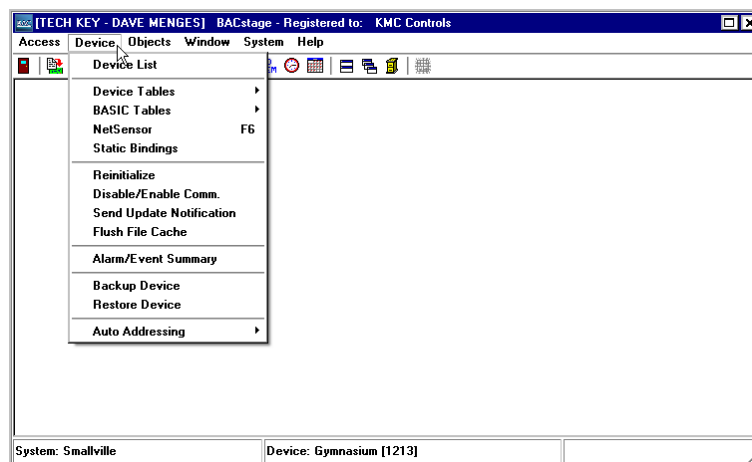
Through the Device Menu you can select and manage a device on a BACnet internetwork.

---

Use *Device Menu* for the following functions:

- ◆ *Device List* on page 24
- ◆ *Device Tables* on page 25
- ◆ *Basic Tables* on page 26
- ◆ *Configuring the NetSensor* on page 27
- ◆ *Static Binding* on page 34
- ◆ *Reinitialize Device* on page 35
- ◆ *Disable/Enable device* on page 36
- ◆ *Send Update Notification* on page 36
- ◆ *Flush File Cache* on page 37
- ◆ *Alarm/Event Summary* on page 37
- ◆ *Backup Device* on page 38
- ◆ *Restore Device* on page 38
- ◆ *Backup Files* on page 39
- ◆ *Restore Files* on page 40
- ◆ *Auto Addressing* on page 40

**Illustration 3–1** Device menu

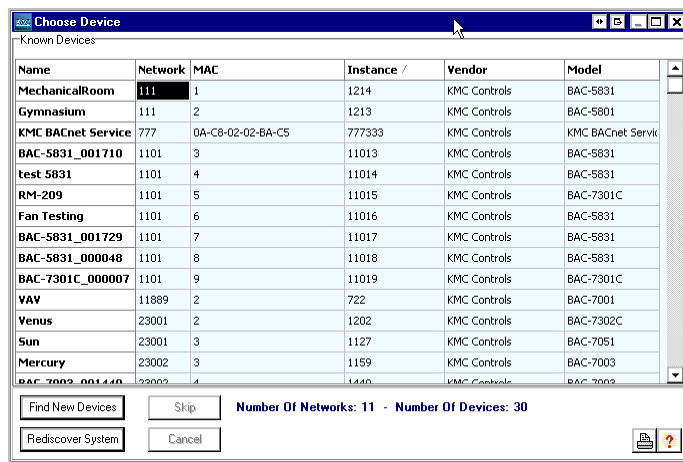


## Device List

Use the device list to choose a device from all of the devices on the internetwork to which you have connected using the *System List*. The information displayed in *Device List* is read-only. For KMC Control’s BACnet devices, property information is entered in the *Device Objects* dialog in the Objects menu.

To choose a device, click *Name*. If the device is available, the *Device List* dialog closes and the toolbar icons become active.

**Illustration 3–2 Device List dialog**



### Features of the device list

**Name** The name of the device.

**Network** The BACnet network number to which the device is attached.

**MAC** The MAC address of the device.

**Instance** The device instance number.

**Vendor** Manufacturer’s name.

**Model** Device model number.

**Find New Devices** Discovers new devices but does not delete any old devices from the device list.

**Rediscover System** Sends a BACnet *Who Is?* request to the network which results in a discovery of all of the on-line devices. If a device was once in the list but is not on-line, the device name will be empty.

**Number of Networks and Number of Devices** BACstage displays the total number of discovered networks and devices at the bottom of the device list dialog.



### Sorting the device list

To change the order in which the information in the device list is displayed, click the top of any column. BACstage will reorder the list based on the contents of the column. Clicking at the top of the same column will reverse the sort order.

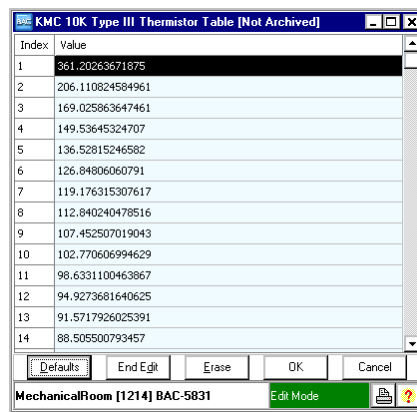
## Device Tables

A device table converts an input—such as a voltage from temperature transmitter—to an output such as temperature in degrees. The built-in tables in BACstage include values for the two thermistors in KMC Controls thermostats and the air flow sensor in the BAC–7000 series VAV controllers. Two additional tables are available for custom programming.

**Table 3–1 Device tables**

Device in table	Description
KMD 10K Type II	KMC Type II thermistors
KMD 10K Type III	KMC Type III thermistors
Airflow Sensor Table	BAC–7000 series VAV controllers
Device Table 4	Available for user input
Device Table 5	Available for user input

**Illustration 3–3 Device table**



To program a custom table, calculate a set of conversion factors based on 128 sample points spread over a 0-5 volt range. Use either the data from a device manufacturer’s specifications or data from measuring a device at a known voltage and recording its output. Enter the output value that corresponds to the input voltage in the *Index* column.

**Defaults** Restores the default values for the table. For the two thermistor devices and the air flow sensor, the default values are the manufacturer’s specifications for the devices. Use *Defaults* to load the default values into the table when using a table item in an object *Device Type*.

**Edit/End Edit** To make changes to the table properties, click *Edit*. BACstage indicates that edit mode is active by turning the status block green. Make changes and then click either *End Edit* or *Ok*.

**Erase** Deletes all entries in the table.

**Cancel** Closes the dialog box without making changes.

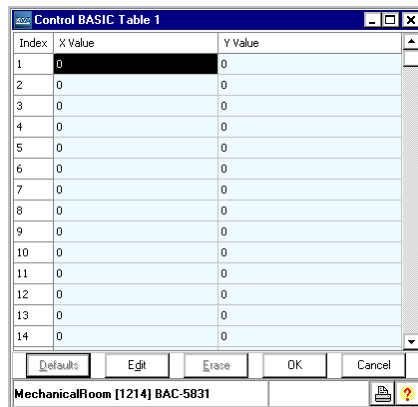
**OK** Sends the new settings to the controllers and closes the dialog box.

## Basic Tables

A Basic Table performs a look-up function based on the input of an object and 32 pairs of data. The table uses the input of the object to perform a linear interpolation between data pairs to calculate an output. The interpolated output is then used by an associated object as its present value. Tables are useful for any of the following reasons:

- ◆ a custom input range is required for a sensor that is not listed as a device type property for an input or value object.
- ◆ to create functions within Control Basic
- ◆ the value of an expression is nonlinear or requires a complicated calculation.

**Illustration 3–4 Basic Table**



Each of the three Basic Tables consists of up to 32 data pairs stored in two columns. Values in the X column correspond to the input value and must be entered in ascending order.

To associate a table with an object, choose the table number from the drop down list in the object’s device type property.

See the keyword [TBL on page 146](#) to access table data from a Control Basic program.

**Defaults** Restores the default values for the table.

**Edit/End Edit** To make changes to the table properties, click *Edit*. BACstage indicates that edit mode is active by turning the status block green. Make changes and then click either *End Edit* or *Ok*.

**Erase** Deletes all entries in the table.

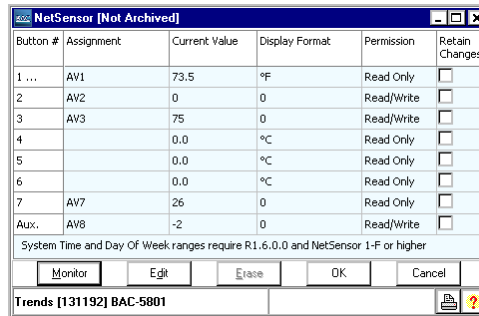
## Configuring the NetSensor

The NetSensor is a wall mounted display and sensor unit that connects directly to a KMC Controls BACnet controller. The unit consists of an LCD display, a thermistor, seven buttons, and an optional humidity sensor.

See the following for additional information on the NetSensor.

- ◆ [Programming the NetSensor on page 31](#)
- ◆ [NetSensor button assignments on page 30](#)
- ◆ [Adjusting calibration on page 29](#)
- ◆ [Setting display blanking on page 30](#)
- ◆ [NETSENSORSTATUS on page 133](#)
- ◆ In the installation guide shipped with the NetSensor.

**Illustration 3-5 NetSensor dialog**



**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the NetSensor dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays NetSensor values. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Clears the data for the selected button.

**Cancel** Closes the dialog box without making changes.

**OK** Sends the new settings to the controller and closes the dialog box.

**Assignment** Points to the object within the controller that stores the button value.

**Current Value** This numerical property indicates the current value—in engineering units—of the present value of the object entered in *Assignment*.

**Display Format** Use *Display Format* to select one of the units from the drop-down list.

- ◆ Units in the table *NetSensor analog display formats* are active when the object under *Assignment* is an analog input, output or value object.
- ◆ Units in the table *NetSensor binary display formats* are active when a binary input, output or value object is selected under *Assignment*.

**Table 3–2 NetSensor analog display formats**

Unit	Action and display
°C	Displays temperature in degrees Celsius. Available only on Button 1. If °C is selected, Button 1 is assigned to the internal temperature sensor and must be associated with a value object.
°F	Displays temperature in degrees Fahrenheit. Available only on Button 1. If °F is selected, Button 1 is assigned to the internal temperature sensor and must be associated with a value object.
0	Sets the precision of the display to nearest whole number.
0.0	Sets the precision of the display to one place to the right of the decimal point.
0.00	Sets the precision of display to two places to the right of the decimal point.
Time	Sets the NetSensor to display a time format.
Off/Low/High	The NetSensor cycles through each word as arrow buttons are pressed and released. The analog value object cycles from 0 to 2.
Off/On1/2/3	The NetSensor display cycles through each word as arrow buttons are pressed and released. The analog value object cycles from 0 to 3.
System Time	Use to set time in a stand-alone controller when an operator workstation or other time master device is not available.
Day Of Week	Use to set the day of week in a stand-alone controller when an operator workstation or other time master device is not available.

**Table 3-3 NetSensor binary display formats**

Label	Action and display
On/Off	The NetSensor toggles between words as arrow buttons are pressed and released. The binary object toggles between 0 and 1.
Low/High	The NetSensor toggles between words as arrow buttons are pressed and released. The binary object toggles between 0 and 1.
Cool/Heat	The NetSensor toggles between words as arrow buttons are pressed and released. The binary object toggles between 0 and 1.

**Permission** The permission property sets the button to be either a display-only button or a button that an operator can use to change a value in the building automation system.

- ◆ *Read* indicates an operator may only view the data displayed on the NetSensor.
- ◆ *Write* indicates an operator may use the arrow buttons to change a value.

**Retain Changes** When selected, the present value of the object associated with a button is written to flash memory and retrieved after a cold start.

**Tip:**

*Retain Changes* is operational only for controllers with a firmware release of 1.4 or later.

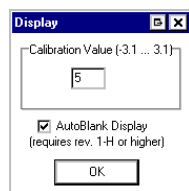
**Adjusting calibration**

A calibration constant can be added to the Button 1 value in the Display dialog.

To add a calibration constant, click the ellipsis (...) next to 1 in the Button column. The Display dialog opens.

- ◆ For a low temperature reading enter a positive correction value.
- ◆ For a high temperature reading enter a negative correction value.
- ◆ The maximum calibration is 3.2 degrees Fahrenheit above or below the displayed value.

**Illustration 3-6 NetSensor calibration dialog**



### Setting display blanking

To set automatic display blanking, click the ellipsis (...) next to 1 in the Button column. The Display dialog opens.

When the AutoBlank Display check box is selected, the NetSensor display will go dark approximately 15 seconds after the last button was pushed.

**Note:** Automatic display blanking is not available on NetSensors manufactured before June 2010.

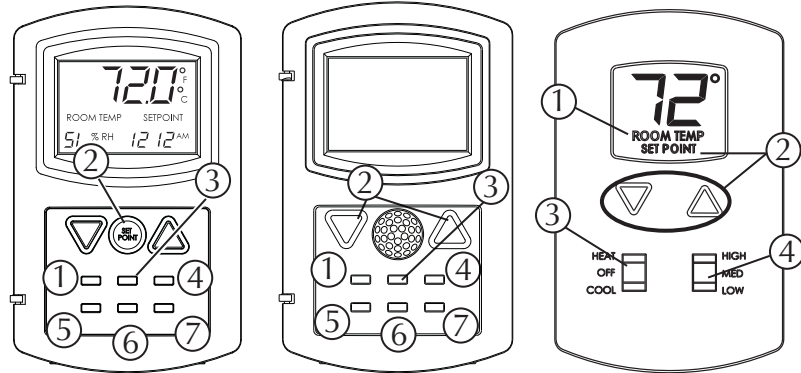
### NetSensor button assignments

The functions of the NetSensor buttons are listed in the following table.

Table 3–4 Netsensor buttons

Button	Function
Up arrow	Increases displayed analog values and toggles binary values
Down arrow	Decreases displayed analog values and toggles binary values
Button 1	Typically assigned to the temperature sensor internal to the NetSensor.
Button 2	Labeled as <i>Setpoint</i> but can be defined as an input, output or value object within the controller.
Buttons 3–6	Can be defined as an input, output or value object within the controller.
Button 7	On applicable models, associated with the optional humidity sensor.
Aux	Press buttons 5 and 7 together and then press the up arrow or down arrow button to change <i>Aux</i> from <i>Off</i> to <i>On</i> .

**Illustration 3-7 NetSensor buttons**



**Programming the NetSensor**

The following examples are methods by which the NetSensor can be configured to display room temperature, humidity, a setpoint and time. See [Configuring the NetSensor on page 27](#) for details about the NetSensor dialog box and button assignments.



**Caution**

When displaying the value from the internal temperature sensor, only associate Button 1 with an analog value object. Associating Button 1 with an input or output object will result in improper operation.

**Button 1 and the internal temperature sensor** The internal temperature sensor inside of the NetSensor are usually associated with analog value object AV1.

1. In *Assignment*, select an analog value object to associate with the temperature button.
2. Set *Display Format* to °F or °C.
3. Set *Permission* to *Read*. Operators can then view the room temperature by touching Button 1 but cannot change the value.

**Button 1-controller points** To manage or display an object value from Button 1, configure as follows:

1. In *Assignment*, select an input, output or value object to associate with the temperature button.
2. Set *Display Format* to 0, 1 or 2 *Decimal*.
3. Set *Permission* to *Read* or *Read/Write*.
  - If set to *Read*, operators can view the value associated with the Button but cannot change it.
  - Is set to *Read/Write* operators can change the value of the object by first touching Button 1 and then pressing the up or down arrow buttons.

**Setpoint** Button 2 is usually associated with analog value object AV2.

1. In *Assignment*, select an analog value object to associate with the setpoint button.
2. Set *Display Format* to 0, 1 or 2 *Decimal*.
3. Set *Permission* to *Read/Write*. Operators can view and change the setpoint by first touching button 2 and then pressing an up or down arrow button.
4. Write a Control Basic statement to start equipment based on the conditions of the setpoint.

```
10 IF AV2 < AV1 THEN START BO6
```

**Humidity** (Humidity equipped models only) Button 7—the humidity sensor— is usually associated with analog value object AV7.

1. In *Assignment*, select an analog value object to associate with the humidity button.
2. Set *Display Format* to 0 *Decimal*.
3. Set *Permission* to *Read/Write*. Operators can view the room humidity by touching button 7 but, cannot change the value.

**Time** Typically button 5 is assigned to display system time and is associated with analog value object AV5.

1. Set *Display Format* to *Time* which will automatically format the display with a colon (:).
2. In *Assignment*, select an analog value object to associate with the time button.
3. Add a Control Basic line as follows:

```
10 AV5 = TIME
```

**Day of Week** Use to enter a day of the week from a NetSensor.



1. Set *Display Format* to *Day Of Week*.
2. *Assignment* is left empty. It cannot be configured.
3. Add a Control Basic line as follows:

```
10 AV5 = DOW
```

**Note:**

In the analog variables dialog box and in Control BASIC, the day of week value is represented by 0 thru 6 for Sunday (0) thru Saturday (6). In the NetSensor dialog box and on the NetSensor display the day of week value is represented by 1 thru 7 for Sunday (1) thru Saturday (7).

**System Time** The *System Time* feature is designed for stand-alone controllers. To change time from a NetSensor:

1. In *Assignment*, select an analog value object to associate with the system time button.
2. Designate a button for system time and set *Display Format* to *System Time*.
3. Operators can view and change the time in only the connected controller by first touching the designated system time button and then pressing an up or down arrow button.

**Motion sensing** (Motion sensing models only) Detects movement in the room and changes the value of the Auxillary function.

- ◆ Under *Assignment*, select an analog value object to associate with the motion sensor. Typically this is analog value object AV8
- ◆ Set *Display Format* to 0.
- ◆ Set *Permission* to *Read/Write*.
- ◆ Use Control Basic to test the state of the object assigned to Auxillary.
  - A value of -1 indicates motion
  - A value of -2 indicates no motion

A value of 0 or 1 indicates the auxiliary function is active.

```
10 IF+ AV8 = -1 THEN START BV9 , STOP A
20 IF AV8 = -2 THEN START A
30 IF TIMEON( A ) > 0:20:00 THEN STOP BV9
```

**Verifying a functioning NetSensor** To check if a functioning NetSensor is present, use the Control Basic keyword [NETSENSORSTATUS](#) on page 133.

```
10 IF NOT NETSENSORSTATUS THEN STOP BV1
```

## Static Binding

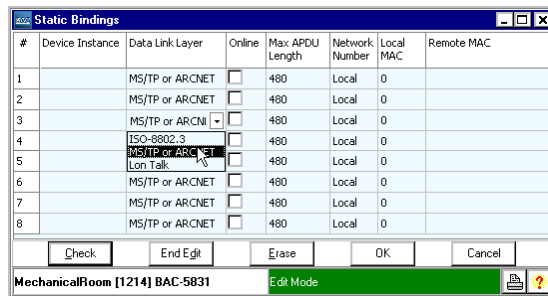
BACnet devices are referenced by their device object instance through a binding process. Normally device address binding is accomplished via the BACnet *Who-Is* and *I-Am* services. It is possible that a BACnet device may need to communicate with another device, such as a slave device, that cannot send an *I-Am* response to a *Who-Is* service request. In this case, use static binding to bind a device instance to a physical address.

Although device static binding is intended to resolve addresses for MS/TP slave devices, it can be used to define any device that cannot identify itself with the *I-Am* service. An example of this would be a computer that is not a BACnet device residing on an Ethernet network that can receive event and alarm notifications.

The information required to statically bind a device instance to a physical address includes

- ◆ The instance number of the device,
- ◆ The number of the BACnet network on which the device resides
- ◆ The MAC address of the device.
- ◆ If a device is not on the local network, then routing information is also needed.

**Illustration 3–8 Static binding dialog box**



**Device Instance** Enter the device instance of the remote device. The device instance can range from 0 to 4,194,302; device instance 4,194,303 is reserved.

**Data Link Layer** Enter the network protocol.

**Online** Select the Online check box to make the device active on the internetwork.

**Max APDU Length** Enter the maximum of size of an *APDU*.

**Network number** Enter the number of the local network of the device being configured.

**Local MAC and Remote Mac** Entering the MAC addresses depends upon where the device is located:

- ◆ If the device to which you are binding *is* on the local network, enter the MAC address of the device in Local Mac. The Remote Mac entry remains empty.
- ◆ If the device to which you are binding *is not* on the local network, enter the MAC address of the device in the Remote MAC. Enter the MAC address of the router in Local MAC.

The following apply to static device binding in KMC BACnet controllers:

- ◆ KMC controllers support a maximum of eight device static bindings.
- ◆ The static device binding for a particular device instance must be defined on every controller that needs to communicate with that device instance.
- ◆ The network topology must be known, particularly, the BACnet network number each device resides on.
- ◆ Static bindings can be overridden. If a device instance is configured to be statically bound and I-Am request is received that matches that device instance, then the addressing information in the I-Am request will replace the static binding information. Care must be taken to make sure that static bindings do not conflict with devices that support the DM-DDB-B BIBB.

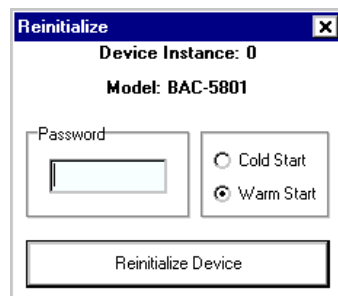
See also the following information:

- ◆ [I-Am service](#) on page 210
- ◆ [Who-Is service](#) on page 215
- ◆ The topic [master and slave devices](#) on page 211

## Reinitialize Device

Use *Reinitialize Device* to force either a *Warm Start* or *Cold Start* in the connected controller. Changes made to the controller are not effective until a warm start, update notification, cold start or power up cycle is performed.

**Illustration 3-9 Reinitialize Device dialog**



**Warm Start** Restarts the process in the controller. All Control Basic programs suspend operation and present values are held at their condition prior to the warm start.

**Cold Start** Restarts the processor in the controller and sets it to its power-up state. All outputs and values are set to default levels until Control Basic programs return the outputs to operational levels.

**Password** Enter the password *snowman*.

### Disable/Enable device

Sends the BACnet *silence* command to the connected device. In the silence mode the device will process and respond to all *APDU*'s as required but will not initiate messages except an *I Am* in response to a Who Is APDU.

**Illustration 3–10 Disable/Enable Device dialog**



### Send Update Notification

Sends an update notification to the connected device.

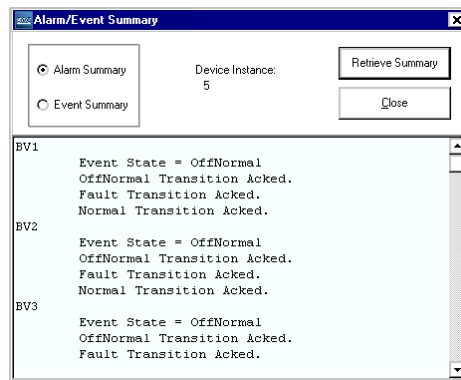
## Flush File Cache

Stores data from writable properties to in FLASH memory. This is normally done in a KMC Controls BACnet controller every five minutes or as part of a warm start or cold start. Choosing *Flush File Cache* forces the data to be stored immediately.

## Alarm/Event Summary

Use *Alarm/Event Summary* to review the most recent alarm and event actions in a device. To review the alarms and events received by BACstage, see [Alarms/Events on page 170](#).

**Illustration 3-11 Alarm/Event Summary dialog**



**Alarm Summary** Displays the recent alarms and information about them in the selected device. After choosing *Alarm Summary*, click *Retrieve Summary*.

**Event Summary** Displays the recent events and information about them in the selected device. After choosing *Event Summary*, click *Retrieve Summary*.

**Device Instance** Enter the instance number of the device to examine.

**Retrieve Summary/Next** Opens the a selected alarm and event LOG file. Continue to click *Next* to scroll through the list. LOG files are stored in the Alarms folder inside of the job folder. See the topic [BACstage job file location on page 179](#) for details about the job folder.

**Close** Exit from the dialog.

## Backup Device

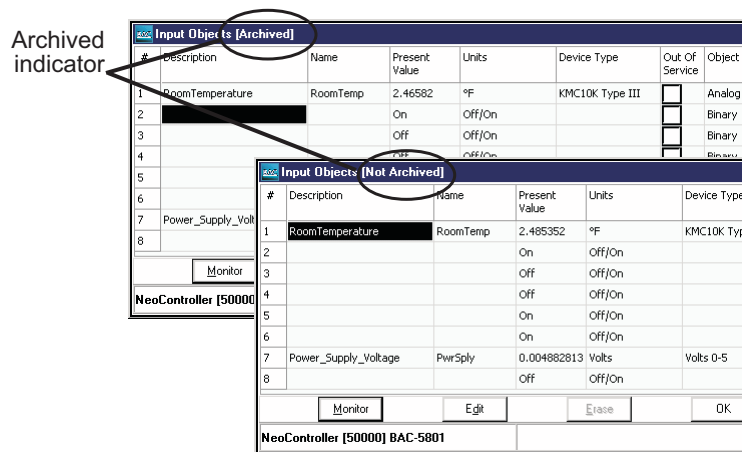
Use *Backup Device* to save to disk the configuration of the connected controller. BACstage stores object, device properties and programming information in a BAC file. The BACstage default storage location is the BAC folder inside of the job folder. See [BACstage job file location on page 179](#). The BAC file is human-readable with a text editing program.

### Related topics

- ◆ [Restore Device on page 38](#)
- ◆ [Backup Files on page 39](#)
- ◆ [Restore Files on page 40](#)
- ◆ [Network Backup on page 167](#)
- ◆ [Network Restore on page 168](#)

BACstage indicates that an object has been changed and not back up by displaying Not Archived in the title bar. The illustration [Achieved and Not Achieved objects on page 38](#) shows examples of both indicators.

**Illustration 3–12 Achieved and Not Achieved objects**

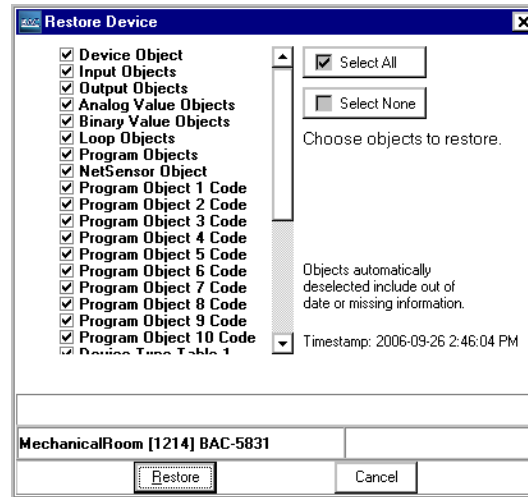


## Restore Device

Use *Restore Panel* to retrieve object, device properties and programs from a BAC file from disk and send them to the connected controller. BACstage looks first in the BAC folder in the job folder for a BAC file. See [BACstage job file location on page 179](#).

Use the *Selective Restore* dialog to choose properties from the backup file to send to the panel.

Illustration 3-13 Selective Restore dialog



**Note:** If *Device Object* is selected from the *Selective Restore* dialog, only the name, description and location properties are retrieved from the BAC file and sent to the panel.

#### Related topics

- ◆ [Backup Device on page 38](#)
- ◆ [Backup Files on page 39](#)
- ◆ [Restore Files on page 40](#)
- ◆ [Network Backup on page 167](#)
- ◆ [Network Restore on page 168](#)

## Backup Files

Use Backup Files to save to disk the configuration of the connected controller. The Backup Files command saves the configuration in an .NGB file which is not compatible with .BAC files. Using Backup Files requires the password Snowman.

#### Related topics

- ◆ [Restore Files on page 40](#)
- ◆ [Backup Device on page 38](#)
- ◆ [Restore Device on page 38](#)
- ◆ [Network Backup on page 167](#)
- ◆ [Network Restore on page 168](#)

## Restore Files

Use Restore Files to open an .NGB file and then restore the configuration of the controller from which it was saved. Using Restore Files requires the password Snowman.

### *Related topics*

- ◆ [Backup Files on page 39](#)
- ◆ [Backup Device on page 38](#)
- ◆ [Restore Device on page 38](#)
- ◆ [Network Backup on page 167](#)
- ◆ [Network Restore on page 168](#)

## Auto Addressing

Use the commands in Auto Addressing to control and manage MS/TP automatic MAC addressing in KMC BACnet controllers.

**Request Status** Displays the status of automatic addressing in the controller.

**Lock Addresses** (Anchor controller only) Locks the temporary MAC addresses in nomad controllers and makes the MAC addresses permanent.

**Restart Anchor** (Anchor controller only) Restarts the auto addressing process without sending a warm start or cold start command to the anchor controller.



## Section 4: The Objects menu

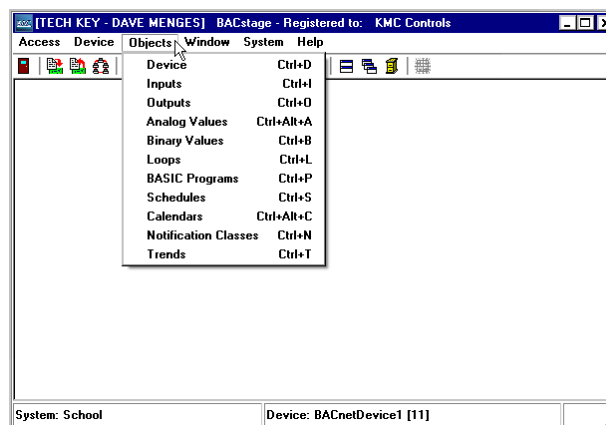
Use the Object Menu to program and configure properties with objects.

---

Select from the following list to program an object.

- ◆ *Device Objects* on page 42
- ◆ *Input objects* on page 45
- ◆ *Output objects* on page 53
- ◆ *Analog value objects* on page 59
- ◆ *Binary value objects* on page 62
- ◆ *Multistate value objects* on page 64
- ◆ *Loop objects* on page 69
- ◆ *Basic Programs* on page 67
- ◆ *Schedule object* on page 72
- ◆ *Calendar object* on page 76
- ◆ *Notification object* on page 77
- ◆ *Event enrollment object* on page 79
- ◆ *Trend object* on page 88

**Illustration 4–1** Object menu



## Device Objects

Use the Device Objects dialog box to configure the device object properties in a KMC BACnet controller. Assign a MAC address and Device Instance and set Baud before placing the device on a network. See [Configuring a controller for the network on page 9](#).



**Caution**

Do not connect any device to a network until it is configured with a MAC address, device instance and baud rate. Connecting a device to a network before it is configured will cause unpredictable operation and a disruption of network traffic. See application note AN-0404A, *Planning BACnet Networks*, for details on numbering devices.

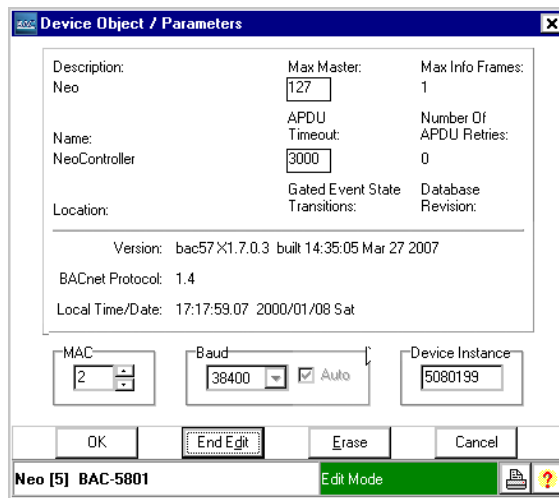
**End Edit** Clicking **End Edit** returns to the device object properties, click *Edit*. BACstage indicates that edit mode is active by turning the status block green. Make changes and then click either *End Edit* or *Ok*.

**Erase** Erases the description, name and location text boxes and sets *MAC*, *Device Instance*, *Baud*, *APDU Timeout* and *Max Master* to defaults.

**Cancel** Closes the dialog box without making changes.

**OK** Sends the new settings to the controllers and closes the dialog box.

**Illustration 4–2 Device Object dialog box**



**Description** A 32-character label of the device.

**Name** A 16-character label of the device. This property must be unique among all devices on the internetwork. The set of characters used in *Name* is

restricted to printable characters. The default entry for *Name* is the model number of the controller followed by the serial number.

**Location** An optional 32-character description of the object's physical location.

**MAC** The MAC address is a number assigned to the device that indicates its node number on the MS/TP network to which it is connected.

- ◆ The address must be unique on the local network but, may be duplicated on other MS/TP networks.
- ◆ MAC addresses for MS/TP networks start at 0 and are assigned sequentially.
- ◆ Valid addresses are 0-127 for master devices. However, BACstage reserves MAC address 0 for BACstage, a service tool or a router. MS/TP MAC addresses 1-127 are the only addresses that can be assigned to controllers with BACstage.

See also [MAC address on page 211](#).

**Device Instance** A number that uniquely identifies the device on the internetwork. The device instance number is determined by the BACnet system designer. Valid instance number's range from 0 to 4,194,303. It is by reference to the device instance number that data is exchanged between BACnet devices.

**Baud** Set *Baud* to match the speed of the MS/TP network to which the device is connected. All devices on the same network must be set to the same speed. Valid settings are 9600, 19,200, 38,400 and 76,800 baud. The default rate is 38400.

**Auto** When selected, the controller will automatically set the MS/TP baud to match the network baud.

**Max Master** Indicates the highest MAC address the device will attempt to locate while polling for a master device on the local network.

- ◆ Setting *Max Master* to allow an additional five controllers beyond the number of controllers connected to the local network will not significantly decrease response time.
- ◆ Setting *Max Master* significantly higher than the highest numbered device could result in increased polling and slower response times.
- ◆ In BACstage, *Max Master* cannot be set lower than the *MAC* address of the controller.



Setting *Max Master* lower than the highest addressed controller will result in controllers that are not polled and data from those controllers that is not shared.

---

**APDU Timeout** Indicates the period—in milliseconds— between retransmissions of an APDU requiring an acknowledgement for which no acknowledgment has been received. The default value is 3000 milliseconds.

**Max Info Frames** Sets the maximum number of packets that are sent before passing the token. *Max Info Frames* is not editable and is set to 1 by BACstage.

**Number of APDU Retries** Indicates the maximum number of retries that an APDU shall be retransmitted. It is fixed at 0.

**Database Revision** An number under control of the device's firmware that displays the revision of the device's internal database. The revision number is incremented when an object is created, an object is deleted, the name of an object changes, an object identifier number changes or a restore is performed.

**Local Time and Date** Displays the system time as kept in the connected device.

**Version** Displays the firmware version number stored in the device. Check the [KMC Controls website](#) for the most current version of firmware. When calling for technical support, have the firmware release number available.

**Gated Event State Transitions** When selected, prevents the in-alarm bit in the status flags property (a property not visible in BACstage) from indicating an alarm condition when *Event Enable* within an object is set to *Disabled*. This prevents Acuity and third-party software from detecting an alarm condition when the *Event Enable* property is set to *Disabled*.

When the *Gated Event State Transactions* check box is clear, the in-alarm bit indicates an alarm when the present value of an object meets alarm conditions regardless of the *Event Enable* property value.

The *Event Enable* property is set for alarm or event conditions within input, output, value, loop and trend objects. See also [Working with alarms and events on page 83](#) for details about setting up alarms and events.

**Backup Failure Timeout** This is an optional property that indicates the time—in seconds—that the device being backed up or restored must wait before unilaterally ending the backup or restore procedure.

**APDU Segment Timeout** The APDU Segment Timeout property indicates the amount of time—in milliseconds—between retransmission of an APDU segment. The default value for this property is 2000 milliseconds. To maintain reliable communication, set the values of the Segment Timeout properties of all device objects of all intercommunicating devices to the same value.

**UTC Offset** The UTC Offset property indicates the time offset—in minutes—between local standard time and Universal Time Coordinated. The value of the property ranges from -780 to +780 seconds. The time zones to the west of the zero degree meridian are positive values; those to the east are negative values. The value of the UTC Offset property is subtracted from the UTC

received in a UTC Time Synchronization service request to calculate the correct local standard time.

**System Status** This property reflects the current physical and logical status of the BACnet device. System Status can have any of the following properties:

- ◆ OPERATIONAL
- ◆ DOWNLOAD\_IN\_PROGRESS
- ◆ OPERATIONAL\_READ\_ONLY
- ◆ NON\_OPERATIONAL
- ◆ DOWNLOAD\_REQUIRED

**MAX APDU Length** This property is the maximum number of octets that may be contained in a single, indivisible application layer protocol data unit. The value of this property is greater than or equal to 50.

**DST Status** The **Daylight Saving Status** property indicates *True* when daylight saving time is in effect and *False* when it is not in effect at the device's location.

**Last Restore Time** This is an optional property that holds the time at which the device's firmware image was last restored. This property is supported if the device supports the BACnet backup and restore procedures.

**Segmentation Supported** BACnet segmentation indicates whether the device supports segmentation of messages and, if so, whether it supports segmented transmission, segmented reception, or both:

- ◆ SEGMENTED\_BOTH
- ◆ SEGMENTED\_RECEIVE
- ◆ SEGMENTED\_TRANSMIT
- ◆ NO\_SEGMENTATION

**Max Segments** This property indicates the maximum number of APDU segments accepted by the device.

## Input objects

Use the input object dialog to select and configure each of the controllers inputs with one of the three input object types. Select the input object type from the following list:

- ◆ An analog input object
- ◆ A binary input object
- ◆ An accumulator object for pulse inputs.

For additional information, see [About objects, properties and services](#) on page 91.

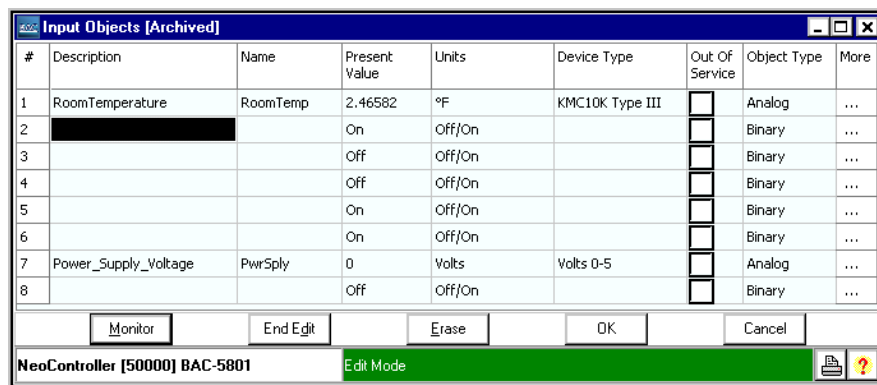
**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the input dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays input signals as they are received. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the input object to the original KMC settings.

**OK** Sends the new settings to the controllers and closes the dialog box.

**Cancel** Closes the dialog box without making changes.

**Illustration 4–3 Input object dialog**



**# (Input number)** The input object number. Input objects are numbered sequentially within the KMC Controls BACnet device and directly correspond to the controller’s input terminal.

**Description** A 32-character label of the object. The set of characters entered for *Description* must be printable characters.

**Name** A 16-character label of the object. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must be printable characters.

**Present Value** This numerical property indicates the current value of the input being measured. To manually change the present value property, first check *Out-Of-Service* and then change *Present Value*.

BACstage can add a calibration constant to the present value of an analog input. To calibrate the input, in edit mode do the following:

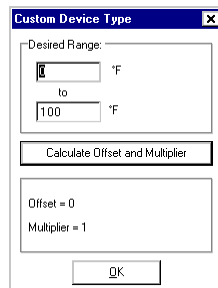
1. Right-click on the present value of the input to calibrate.
2. Select the shortcut menu *Calibrate Present Value*.
3. Add a calibration constant.
  - For a low present value reading enter a positive correction value.
  - For a high present value reading enter a negative correction value.

**Units** Select a unit of measure to associate with the input signal. BACstage supports units of measure for both analog and binary inputs. For binary inputs, the first unit in the pair of units is the Normal Inactive state of the input. See [Supported engineering units on page 177](#) for a list of the available units. See also [Polarity on page 48](#) for the relationship between input *Units* and *Polarity* property.

**Device Type** This property is a description of the physical device connected to the input. Choose from an available type in the drop-down list. See [Device Tables on page 25](#) when using a KMC thermistor or air flow sensor.

**Custom Device** An analog custom device automatically scales an input voltage for display as a unit of measure. For example, if the output of a temperature transmitter represents 0 volts at 0 degrees and 5 volts represents 100 degrees, BACstage will display a 2.5 volt input as 50 degrees.

#### Illustration 4-4 Custom device dialog



To define a custom device, do the following:

1. Choose a label from the *Units* list that represents the input of the custom device.
2. Choose *Custom Device* from the *Device Type* list.
3. Enter the display value the corresponds to 0 volts in the top box.
4. Enter the display value that corresponds to 5 volts in the bottom box.
5. Click *Calculate Offset and Multiplier*. BACstage displays new values in the bottom of the dialog.
6. Close the dialog.

**Out Of Service** *Out Of service* indicates that the physical input is internally disconnected from the input object. BACstage sets this property to *True* when checked and *False* when unchecked. When *Out Of Service* is checked, and sent to the controller, *Present Value* does not respond to changes at the physical input of the device.

**Object Type** Sets the input as either an analog, binary or accumulator object.

- ◆ Analog—Devices with modulating inputs that operate from a varying voltage (0-5 volts)
- ◆ Binary—Devices which require one of only two states (*On* or *Off*)
- ◆ Accumulator—Use with power or other meters with pulse inputs.

**Delay** *Delay* defines a minimum period for a set of conditions to exist before a *TO-OFFNORMAL* or *TO-NORMAL* event occurs. Use *Delay* with *High Limit*, *Low Limit* and *Deadband* to define conditions that indicate *Present Value* is out of an expected, predefined operating range. *Delay* is expressed in seconds.

**Event Enable** Use *Event Enable* to enable notifications for *TO-OFFNORMAL*, *TO-NORMAL* and *TO-FAULT*.

**Polarity** (Binary input only) The polarity property sets the relationship between the physical state of the input and the logical state represented by *Present Value*. BACstage displays *Present value* as one of the pairs of units in the table, [Binary unit pairs on page 178](#).

**Table 4–1 Input object polarity relationships**

Polarity	Voltage at input	Unit displayed	Example	Dry contact with pull-up
Normal	0	Normal Inactive	Off, Stop	Closed
Normal	5	Normal Active	On, Start	Open
Reverse	0	Normal Active	On, Start	Closed
Reverse	5	Normal Inactive	Off, Stop	Open



**Scale** (Accumulator object only) This property is the conversion factor which must be applied to the pulse count in *Present Value* to provide a value in the units column.

**Max. Value** (Accumulator object only) Indicates the maximum value of *Present Value*.

**Notification Class** Specifies the notification class object to be used when handling and generating event notifications for this object. See [Notification object on page 77](#) for details about the notification class object.

**Notify Type** This property specifies whether the notifications generated by the object are *Events* or *Alarms*. BACstage displays a red message at the bottom of the work window when it receives an alarm; events are placed in the alarm/event list without displaying a message.

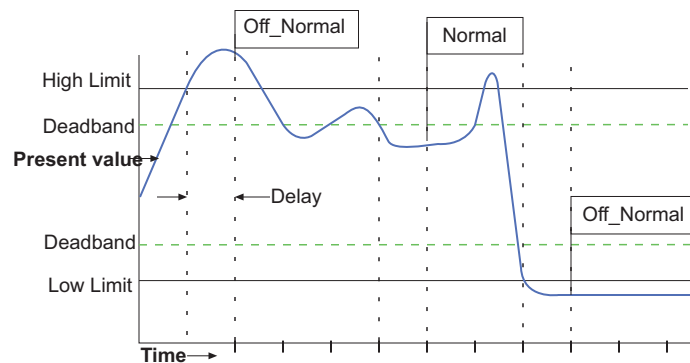
**High Limit** (Analog and accumulator inputs only) This property is used with intrinsic reporting to define an upper limit for a normal operating range of *Present Value* for analog objects and *pulse rate* for accumulator objects. Use with *Limit Enable*, *Deadband* and *Delay* to define conditions that indicate *Present Value* (pulse rate for accumulator objects) is out of a normal operating range.

Conditions for generating an *TO-OFFNORMAL* event when *Present Value* exceeds a predefined upper limit:

- ◆ *Present Value* must exceed *High Limit* for the period specified by *Delay*, and
- ◆ *High* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *OFFNORMAL*.

Conditions for generating a *TO-NORMAL* event when *Present Value* returns to a normal value:

- ◆ *Present Value* must fall below *High Limit* minus *Deadband* for the period specified by *Delay*, and
- ◆ *High* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**Illustration 4–5 Example of Off\_Normal and Normal events**

**Low Limit** (Analog and accumulator inputs only) This property is used with intrinsic reporting to define a lower limit for a normal operating range of *Present Value* for analog objects and *pulse rate* for accumulator objects. Use with *Limit Enable*, *Deadband* and *Delay* to define events that indicate *Present Value* (pulse rate for accumulator objects) is out of a normal operating range.

Conditions for generating an *TO-OFFNORMAL* event when *Present Value* exceeds a predefined lower limit:

- ◆ *Present Value* drops below *Low\_Limit* for the period specified by *Delay*, and
- ◆ *Low* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *OFFNORMAL*.

Conditions for generating a *TO-NORMAL* event when *Present Value* returns to a normal value:

- ◆ *Present Value* must rise above *Low Limit* plus *Deadband* for the period specified by *Delay*, and
- ◆ *Low* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**Monitoring Interval** (Accumulator only) This property specifies the monitoring period—in seconds—for determining the value of pulse rate.

**Deadband** (Analog input only) This property specifies a range between the high limit and low limit properties in which *Present Value* must remain before the device generates a *TO-NORMAL* event.

Conditions for generating a *TO-NORMAL* event are:

- ◆ *Present Value* must fall below *High Limit* minus *Deadband*, and
- ◆ *Present Value* must exceed *Low Limit* plus *Deadband*, and
- ◆ *Present\_Value* must remain within this range for the period specified by *Delay*, and
- ◆ *High*, *Low* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**Limit Enable** (Analog input only) This property separately enables and disables reporting of high limit and low limit *OFFNORMAL* events and their return to normal.

**Filter Weight** Sets the number of samples that are averaged together to calculate the displayed value. A sample is taken on each scan.

**Table 4-2 Filter weight values**

Filter Weight	Scans to average
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

**COV Increment** This property specifies the minimum change of **Present Value** that will send a COV notification to subscriber notification clients.

## Working with accumulator objects

Use the BACnet accumulator object to monitor devices that have pulse outputs. Example devices included:

- ◆ Electric power meters
- ◆ Water meters
- ◆ Natural gas meters.

### Configuring an input as an accumulator object

To connect a device with pulse outputs to a KMC controller you will need the following information. See [Input objects on page 45](#) for details on configuring the accumulator object.

- ◆ Device details supplied by the manufacturer.
- ◆ The manual supplied with the KMC BACnet controller for input connection details.
- ◆ Application note AN0504L, Connecting inputs and outputs to KMC controllers, describes methods for working with pulse inputs.

Any of the controller inputs can be configured as an accumulator object. To maintain the highest performance in the controller, BACstage limits the number of accumulators to three in a single controller.

1. Connect the pulse input to a controller input.
2. Open the *Input* dialog and click *Edit*.
3. Select the input to which the pulse output device is connected.
4. Choose *Accumulator* from *Object Type*. *Device Type* automatically changes to *Pulse*.
5. Click *More* and enter values for *Scale*, *Max. Value* and *Monitoring Interval*.
6. If required, enter values for the alarm and event notification properties.
7. Click *Ok* when complete.
8. Click *End Edit* or *Monitor*.

### Resetting the accumulator present value

The accumulator present value can be cleared or changed only when the object is set to *Out Of Service*.

1. Open the *Input* dialog and click *Edit*.
2. Select the input to which the pulse output device is connected.
3. Check *Out Of Service*.
4. Check *End Edit* or *Monitor* to end the edit session and send an update notification.
5. Click *Edit*.
6. Enter a new present value.
7. Uncheck *Out Of Service*.
8. Click *End Edit* or *Monitor*.

### Programming the accumulator

The accumulator properties available to Control Basic are present value and pulse rate. The following lines of Control Basic assign an accumulator present value and pulse rate to analog value objects.

```
80 AV11 = ACC5-PR
90 AV12 = ACC5
```

### Calculating consumption and demand

To calculate the actual demand and consumption represented by pulse inputs, use the scale factor appropriate for the input device. In the following example, one output pulse from the monitored power meter equals 0.108 kilowatt/hours.

```
110 AV13 = ACC5-PR * 0.108
```

```
120 AV14 = ACC5 * 0.108
```

Line 110 reads the current demand from the accumulator pulse rate, scales it and stores the result in analog value AV13.

Line 120 reads the total consumption from the accumulator present value, scales it and stores the result in analog value AV14.

### Logging demand and consumption

To log demand and consumption in proper units of measure:

1. Configure two analog value objects for the correct units of measure.
2. Use the examples in [Calculating consumption and demand](#) to convert present value and pulse rate to consumption and demand.
3. Configure trend log objects to collect trend data from the analog value objects.

### Accumulator alarms and events

Accumulator alarms and events are configured the same as other alarms and events except that *high-limit* and *low limit* are triggered by the pulse rate and not by the present value property. To configure an alarm or event to trigger the actual unit of measure for consumption and demand:

1. Configure two analog value objects for the correct units of measure.
2. Use the examples in [Calculating consumption and demand](#) to convert present value and pulse rate to consumption and demand.
3. Configure the analog value objects for the alarms or events.

For additional information, see the topic [Working with alarms and events on page 83](#).

## Output objects

Use the output object dialog select and configure each of the controllers outputs as either an analog or binary output object.

For additional informational, see [About objects, properties and services on page 91](#) and [Priority arrays on page 92](#).

**Edit and MonitorDes** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the output dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays output signals as *Present Value* changes. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the output object to the original KMC settings.

**OK** Sends the new settings to the controller and closes the dialog box.

**Cancel** Closes the dialog box without making changes.

#### Illustration 4–6 Output object dialog

#	Description	Name	Present Value	Units	Device Type	Out Of Service	Object Type	Priority Array	More
1	RoomFan	fAN	On	Off/On		<input type="checkbox"/>	Binary	...	...
2	RoomHeat	RmHeat	Off	Off/On		<input type="checkbox"/>	Binary	...	...
3	RoomCool	RmCool	On	Off/On		<input type="checkbox"/>	Binary	...	...
4	Economizer	Econ	50	%	0-100%	<input type="checkbox"/>	Analog	...	...
5			---	Volts		<input type="checkbox"/>	Analog	...	...
6			---	Volts		<input type="checkbox"/>	Analog	...	...
7			---	Volts		<input type="checkbox"/>	Analog	...	...
8			---	Volts		<input type="checkbox"/>	Analog	...	...

Buttons: Monitor, Edit, Erase, OK, Cancel

BACnetDevice1 [11] BAC-5801

**# (Output number)** The output object number. Output objects are numbered sequentially within the KMC BACnet device and directly correspond to the controller's output terminal.

**Description** A 32-character label of the object. The set of characters entered for *Description* must be printable characters.

**Name** A 16-character label of the object. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must be printable characters.

**Present Value** For analog objects, this is a numerical property that indicates the current value—in engineering units—of the output terminal of the device.

For binary objects, *Present Value* reflects the logical state which is either *Inactive* or *Active*. The relationship between *Present Value* and the physical state of the output is determined by the polarity property. The possible states are summarized in the table [Output object polarity relationships on page 58](#).

To manually change the present value property, enter the new value and then press the enter key or click another property. A dialog opens in which the write priority level is selected. See [Priority arrays on page 92](#).

**Units** Select a unit of measure to associate with the output signal. The available units are listed in the section [Supported engineering units on page 177](#). BACstage supports several units of measure for both analog and binary outputs. For binary outputs, the first unit in the pair of units is the Normal Inactive state of the output. See also [Polarity on page 58](#) for the relationship between *Units* and *Polarity* property.

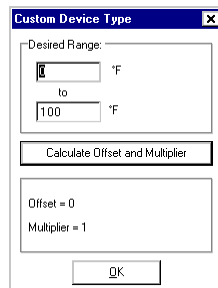
**Device Type** This property is a description of the physical device connected to the output. Choose from an available type in the drop-down list.

**Custom Device** An analog custom device automatically scales the output voltage for display as an output unit of measure. For example, if the input of a variable frequency drive requires 0 volts for minimum speed and 10 volts represents full speed, entering 50% for present value will set the output to 5 volts.

To define a custom device:

1. Choose a label from the *Units* list that represents the input of the custom device.
2. Choose *Custom Device* from the *Device Type* list.
3. Enter the display value the corresponds to 0 volts in the top box.
4. Enter the display value that corresponds to 10 volts in the bottom box.
5. Click *Calculate Offset and Multiplier*. BACstage displays new values in the bottom of the dialog.
6. Close the dialog.

#### Illustration 4-7 Custom device type dialog



**Out Of Service** *Out Of service* indicates that the physical output is internally disconnected from the output object. This property will be *True* when checked or *False* when unchecked. When *Out Of Service* is checked, *Present Value* does not change the value at the output terminal of the controller.

**Object type** Sets the output object as either an analog or binary object.

**Table 4–3 Output object types**

<b>Object types</b>	<b>Description</b>
Analog	Devices with modulating outputs that operate from a varying voltage (0-5 volts)
Binary	Devices which require one of only two states ( <i>On or Off</i> )

**Relinq. Default** Sets the status or value that will take effect when all levels of the priority array are *NULL*. See [Priority arrays on page 92](#).

**Priority Array** Displays the priority level for the object. See [Priority arrays on page 92](#).

**Delay** *Delay* defines a minimum period for a set of conditions to exist before a *TO-OFFNORMAL* or *TO-NORMAL* event occurs. Use *Delay* with *High Limit*, *Low Limit* and *Deadband* to define conditions that indicate *Present Value* is out of an expected, predefined operating range. *Delay* is expressed in seconds.

**Event Enable** Use *Event Enable* to enable notifications for *TO-OFFNORMAL*, *TO-NORMAL* and *TO-FAULT*.

**Notification Class** Specifies the notification class to be used when handling and generating event notifications for an output object. See [Notification object on page 77](#) for details about the notification class object.

**Notify Type** This property specifies whether the notifications generated by the output object are *Events* or *Alarms*. BACstage displays a red message at the bottom of the work window when it receives an alarm; events are placed in the alarm/event list without displaying a message.

**0%–100% Voltage** Sets the voltages which correspond to 0% and 100% output.

**High Limit** (Analog objects only) This property is used with intrinsic reporting to define an upper limit for a normal operating range of *Present Value*. Use with *Limit Enable*, *Deadband* and *Delay* to define conditions that indicate *Present Value* is out of a normal operating range. See the illustration. See the illustration [Example of Off\\_Normal and Normal events on page 50](#).

Conditions for generating an *TO-OFFNORMAL* event when *Present Value* exceeds a predefined upper limit:



- ◆ *Present Value* must exceed *High Limit* for the period specified by *Delay*, and
- ◆ *High* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *OFFNORMAL*.

Conditions for generating a *TO-NORMAL* event when *Present Value* returns to a normal value:

- ◆ *Present Value* must fall below *High Limit* minus *Deadband* for the period specified by *Delay*, and
- ◆ *High* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**Low Limit** (Analog objects only) This property is used with intrinsic reporting to define a lower limit for a normal operating range of *Present Value*. Use with *Limit Enable*, *Deadband* and *Delay* to define events that indicate *Present Value* is out of a normal operating range. See the illustration [Example of Off\\_Normal and Normal events on page 50](#).

Conditions for generating an *TO-OFFNORMAL* event when *Present Value* exceeds a predefined lower limit:

- ◆ *Present Value* drops below *Low\_Limit* for the period specified by *Delay*, and
- ◆ *Low* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *OFFNORMAL*.

Conditions for generating a *TO-NORMAL* event when *Present Value* returns to a normal value:

- ◆ *Present Value* must exceed *Low Limit* plus *Deadband* for the period specified by *Delay*, and
- ◆ *Low* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**Deadband** (Analog objects only) This property specifies a range between the high limit and low limit properties in which *Present Value* must remain before the device generates a *TO-NORMAL* event.

Conditions for generating a *TO-NORMAL* event are:

- ◆ *Present Value* must fall below *High Limit* minus *Deadband*, and
- ◆ *Present Value* must exceed *Low Limit* plus *Deadband*, and
- ◆ *Present\_Value* must remain within this range for the period specified by *Delay*, and
- ◆ *High*, *Low* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**Limit Enable** (Analog objects only) This property separately enables and disables reporting of high limit and low limit *OFFNORMAL* events and their return to normal.

**Polarity** (Binary objects only) For binary outputs this property sets the relationship between the physical state of the output and the logical state represented by *Present Value*.

**Table 4–4 Output object polarity relationships**

Polarity	Voltage at output	Unit displayed	Example
Normal	0	Normal Inactive	Off, Stop
Normal	10	Normal Active	On, Start
Reverse	0	Normal Active	On, Start
Reverse	10	Normal Inactive	Off, Stop

**Feedback Value** (Binary objects only) This property is used with intrinsic reporting to indicate the value from which *Present Value* must differ to generate an event.

Conditions for generating a *TO-OFFNORMAL* alarm event:

- ◆ *Present\_Value* must be different from *Feedback Value* for the minimum period specified by *Delay* and
- ◆ The selection in *Event Enable* must include *OFFNORMAL*.

Conditions for generating a *TO-NORMAL* alarm event:

- ◆ *Present\_Value* must remain equal to *Feedback Value* for the minimum period specified by *Delay* and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**COV Increment** This property specifies the minimum change of **Present Value** that will send a COV notification to subscriber notification clients.

## Analog value objects

Use the analog value object dialog to configure the value object in the BACnet device.

*Related topics*

[Binary value objects on page 62](#)

[Multistate value objects on page 64](#)

[About objects, properties and services on page 91](#)

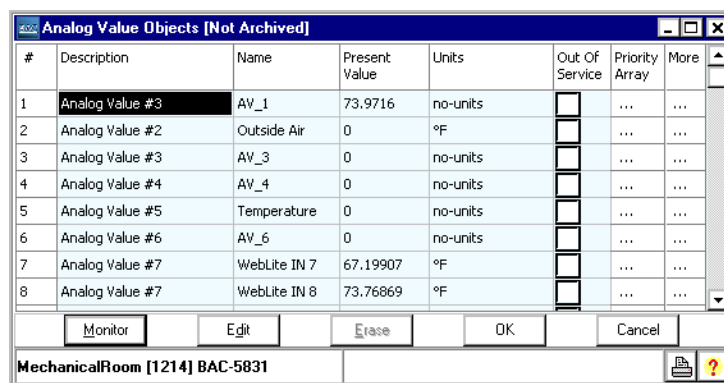
**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the value object dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays input signals as they are received. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the value object to the original KMC settings.

**Cancel** Closes the dialog box without making changes.

**OK** Sends the new settings to the controllers and closes the dialog box.

**Illustration 4-8 Analog Value object dialog**



**# (Object number)** The value object number. Value objects are numbered sequentially within the KMC BACnet device.

**Description** A 32-character label of the object. The set of characters entered for *Description* must be printable characters.

**Name** A 16-character label of the object. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must

be printable characters.

**Present Value** This numerical property indicates the current value—in engineering units—of the value object. To manually change the present value property, enter the new value and then press the enter key or click another property. A dialog opens in which the write priority level is selected.

**Units** Use *Units* to select the unit of measure for the value object. The available units are listed in the section [Supported engineering units on page 177](#).

**Out Of Service** When *Out Of service* is checked, the present value cannot be updated by programs running on the controller. This property will be *True* when checked or *False* when unchecked. Changes can still be made to the object's priority array.

**Priority Array** Displays the priority level for the object. See [Priority arrays on page 92](#).

**Relinq. Default** Sets the value for the object that will take effect when all levels of the priority array are *NULL*. See [Priority arrays on page 92](#).

**Delay** *Delay* defines a minimum period for a set of conditions to exist before a *TO-OFFNORMAL* or *TO-NORMAL* event occurs. Use *Delay* with *High Limit*, *Low Limit* and *Deadband* to define conditions that indicate *Present Value* is out of an expected, predefined operating range. *Delay* is expressed in seconds.

**Event Enable** Use *Event Enable* to enable notifications for *TO-OFFNORMAL*, *TO-NORMAL* and *TO-FAULT*.

**Notification Class** Specifies the notification class to be used when handling and generating event notifications for the analog value object. See [Notification object on page 77](#) for details about the notification class object.

**Notify Type** This property specifies whether the notifications generated by the object are *Events* or *Alarms*. BACstage displays a red message at the bottom of the work window when it receives an alarm; events are placed in the alarm/event list without displaying a message.

**High Limit** This property is used with intrinsic reporting to define an upper limit for a normal operating range of *Present Value*. Use with *Limit Enable*, *Deadband* and *Delay* to define conditions that indicate *Present Value* is out of a normal operating range.

Conditions for generating an *TO-OFFNORMAL* event when *Present Value* exceeds a predefined upper limit:

- ◆ *Present Value* must exceed *High Limit* for the period specified by *Delay*, and
- ◆ *High* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *OFFNORMAL*.

Conditions for generating a *TO-NORMAL* event when *Present Value* returns to a normal value:

- ◆ *Present Value* must fall below *High Limit* minus *Deadband* for the period specified by *Delay*, and
- ◆ *High* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**Low Limit** This property is used with intrinsic reporting to define a lower limit for a normal operating range of *Present Value*. Use with *Limit Enable*, *Deadband* and *Delay* to define events that indicate *Present Value* is out of a normal operating range.

Conditions for generating an *TO-OFFNORMAL* event when *Present Value* exceeds a predefined lower limit:

- ◆ *Present Value* drops below *Low\_Limit* for the period specified by *Delay*, and
- ◆ *Low* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *OFFNORMAL*.

Conditions for generating a *TO-NORMAL* event when *Present Value* returns to a normal value:

- ◆ *Present Value* must exceed *Low Limit* plus *Deadband* for the period specified by *Delay*, and
- ◆ *Low* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**Deadband** This property specifies a range between the high limit and low limit properties in which *Present Value* must remain before the device generates a *TO-NORMAL* event.

Conditions for generating a *TO-NORMAL* event are:

- ◆ *Present Value* must fall below *High Limit* minus *Deadband*, and
- ◆ *Present Value* must exceed *Low Limit* plus *Deadband*, and
- ◆ *Present\_Value* must remain within this range for the period specified by *Delay*, and
- ◆ *High*, *Low* or *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.

**Limit Enable** This property separately enables and disables reporting of high limit and low limit *OFFNORMAL* events and their return to normal.

**COV Increment** This property specifies the minimum change of **Present Value** that will send a COV notification to subscriber notification clients.

## Binary value objects

Use the binary value object dialog to configure the value object in the BACnet device.

### *Related topics*

- ◆ [Analog value objects on page 59](#)
- ◆ [Multistate value objects on page 64](#)
- ◆ [About objects, properties and services on page 91](#)

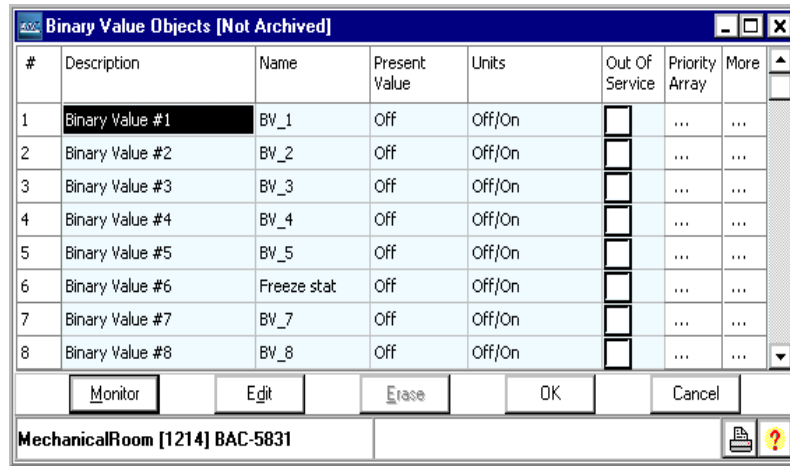
**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the value object dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays input signals as they are received. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the value object to the original KMC settings.

**Cancel** Closes the dialog box without making changes.

**OK** Sends the new settings to the controllers and closes the dialog box.

Illustration 4-9 Binary Value object dialog



**# (Object number)** The value object number. Value objects are numbered sequentially within the KMC BACnet device.

**Description** A 32-character label of the object. The set of characters entered for *Description* must be printable characters.

**Name** A 16-character label of the object. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must be printable characters.

**Present Value** This numerical property indicates the current state—in engineering units—of the value object. To manually change the present value property, enter the new value and then press the enter key or click another property. A dialog opens in which the write priority level is selected.

**Units** Use *Units* to select the unit of measure for the value object. The available units are listed in the section [Supported engineering units on page 177](#).

**Out Of Service** When *Out Of service* is checked, the present value cannot be updated by programs running on the controller. This property will be *True* when checked or *False* when unchecked. Changes can still be made to the object’s priority array.

**Priority Array** Displays the priority level for the object. See [Priority arrays on page 92](#).

**Relinq. Default** Sets the value for the object that will take effect when all levels of the priority array are *NULL*. See [Priority arrays on page 92](#).

**Delay** *Delay* defines a minimum period for a set of conditions to exist before a *TO-OFFNORMAL* or *TO-NORMAL* event occurs. Use *Delay* with *Alarm* to

define conditions that indicate *Present Value* is out of an expected operating state. *Delay* is expressed in seconds.

**Event Enable** Use *Event Enable* to enable notifications for *TO-OFFNORMAL*, *TO-NORMAL* and *TO-FAULT*.

**Notification Class** Specifies the notification class to be used when handling and generating event notifications for a binary value object. See [Notification object on page 77](#) for details about the notification class object.

**Notify Type** This property specifies whether the notifications generated by the object are *Events* or *Alarms*. BACstage displays a red message at the bottom of the work window when it receives an alarm; events are placed in the alarm/event list without displaying a message.

**Alarm** This property is used with intrinsic reporting to define a change of *Present Value* that will generate an alarm event.

Conditions for generating a *TO-OFFNORMAL* alarm event:

- ◆ *Present\_Value* must maintain the value specified by *Alarm Value* for the minimum period specified by *Delay* and
- ◆ The selection in *Event Enable* must include *OFFNORMAL*.

Conditions for generating a *TO-NORMAL* alarm event:

- ◆ *Present\_Value* must remain unequal to the value specified by *Alarm Value* for the minimum period specified by *Delay* and
- ◆ The selection in *Event Enable* must include *NORMAL*.

## Multistate value objects

A multistate value object is a standard BACnet object whose properties represent the state of a multistate value. The actual functions associated with a specific state are not defined by the BACnet standard. For example, a particular state of an object may represent the active or inactive condition of several physical inputs and outputs or the value of an analog input or output.

*Related topics*

- ◆ [Analog value objects on page 59](#)
- ◆ [Binary value objects on page 62](#)

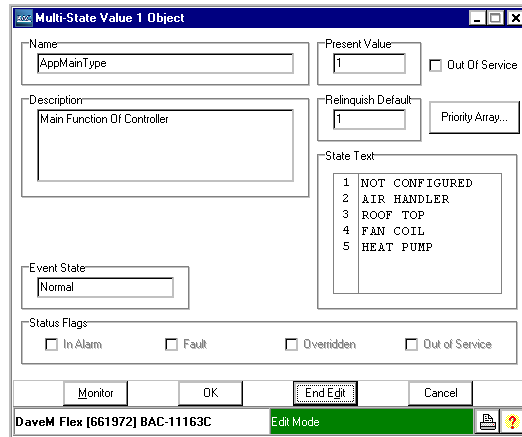
**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the value object dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays each property. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.



**Cancel** Closes the dialog box without making changes.

**OK** Sends the new settings to the controllers and closes the dialog box.

#### Illustration 4-10 Multistate value object



**Name** A 16-character label of the object that must be unique within the BACnet device that maintains it. The set of characters entered for Name must be printable characters.

**Description** A 32-character label of the object. The set of characters entered for Description must be printable characters.

**Present Value** This numerical property indicates the numerical current state of the value object.

**Relinquish Default** Sets and displays the default priority level for the object. See [Priority arrays on page 92](#).

**Priority Array** Click to open. See [Priority arrays on page 92](#).

**Out Of Service** When Out Of service is selected, the present value cannot be updated by programs running on the controller. This property will be *True* (1) when selected or *False* (0) when cleared. Changes can still be made to the object's priority array.

**State Text** Each entry in the State Text list corresponds to a value of Present Value. For example, if Present Value equals 1 then the value of State Text is the first entry in the list.

- ◆ To change the entries in State Text, point to the text and type the changes.
- ◆ To increase the number of entries in the list, place the pointer at the end of the entry and then press ENTER.
- ◆ To decrease the number of entries in the list, place the pointer at the beginning of the entry and then press DELETE.

**Event State** Use **Event State** to determine that this value object has an active event state associated with it.

- ◆ If the object supports intrinsic reporting, then **Event State** indicates the state of the object.
- ◆ If the object does not support intrinsic reporting, then the value of this property is **Normal**.
- ◆ If **Reliability** is present and does not have a value of **No Fault Detected**, then the value of **Event State** is **Fault**. Changes in **Event\_State** to the value **Fault** are considered to be fault events.

**Status Flags** The four BACnet status flags are an indication of the general condition of the multistate value object.

**Table 4–5 Multistate value object status flags**

Flag	Description
IN ALARM	<i>False</i> (0) if the event state property is <i>Normal</i> , other wise <i>True</i> (1)
FAULT	<i>True</i> (1) if Reliability is present and the value for <i>Reliability</i> is not <i>No Fault Detected</i> , otherwise <i>False</i> (0)
OVERRIDDEN	<i>True</i> (1) if the point has been overridden by some mechanism local to the BACnet device. Otherwise, the flag is <i>False</i> (0). When this flag is <i>True</i> , <i>Present Value</i> cannot be changed through BACnet services.
OUT OF SERVICE	<i>True</i> (1) if <i>Out Of Service</i> is selected. Otherwise <i>False</i> (0).

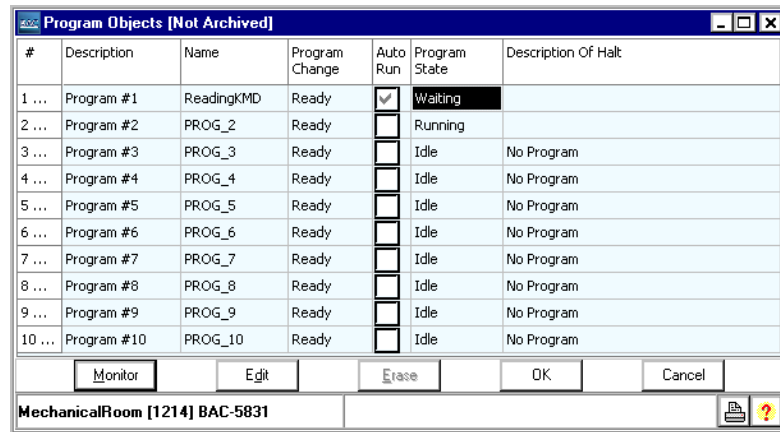
**COV Increment** This property specifies the minimum change of **Present Value** that will send a COV notification to subscriber notification clients.

## Basic Programs

Control Basic program objects are the method by which automation is added to KMC BACnet controllers.

- ◆ For details on writing Control Basic programs, see *About Control Basic programs* on page 97.
- ◆ To open the Control Basic editor see the topic *Using the Control Basic editor* on page 97

**Illustration 4-11 Program Objects dialog**



**Edit and Monitor** When BACstage is in edit mode, the properties in the Control Basic dialog are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the program object dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays input signals as they are received. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the Control Basic dialog to the original KMC settings.

**Cancel** Closes the dialog box without making changes.

**OK** Sends the new settings to the controllers and closes the dialog box.

**# (Input number)** The program number. To choose a program, click *Edit* or *End Monitor* and then the program number.

**Description** A 32-character label of the program. The set of characters entered for *Description* must be printable characters.

**Name** A 16-character label of the program. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must be printable characters.

**Change** Use Change to change the operational state of a program. The process may change its own state as a consequence of execution as well.

**Table 4–6 Program object program changes**

<b>Change</b>	<b>Description of change</b>
READY	The program is ready for a change request. This is the normal state of the object.
LOAD	Requests that the program be loaded, if it not already loaded
RUN	Request that the program begin executing, if not already running
HALT	Request that the program halt execution.
RESTART	Request that the process restart at its initialization point
UNLOAD	Request that execution halts and the program unloads.

**Auto Run** When selected, Control Basic in the program object will automatically start after either a warms start or cold start.

**Program State** This property reflects the current state of the Control Basic program.

**Table 4–7 Control Basic program states**

<b>State</b>	<b>Description of state</b>
IDLE	The program is not executing
LOADING	The program is being loaded.
RUNNING	The program is currently executing.
WAITING	The program is waiting for some external event.
HALTED	The program is halted because of some error condition.
UNLOADING	The program has been requested to terminate.

**Reason For Halt** If the Control Basic program is stopped for any reason, Reason For Halt displays an explanation of the halt.

**Table 4–8 Control Basic reason for halt**

<b>State</b>	<b>Description of state</b>
NORMAL	The Control Basic program has not stopped because of any error condition.

**Control Basic reason for halt (continued)**

State	Description of state
LOAD_FAILED	The Control Basic program could not complete loading.
INTERNAL	The Control Basic program halted because of some internal mechanism.
PROGRAM	The Control Basic program was halted by a program change request.
OTHER	The Control Basic program is halted for some other reason.

**Loop objects**

Use the Loop Objects dialog to manage the PID controller loops in the connected controller. A PID controller is a mathematical function which calculates the analog output required to maintain a process at or near a setpoint. The output of the loop object directly controls the present value of either an analog output or analog value object.

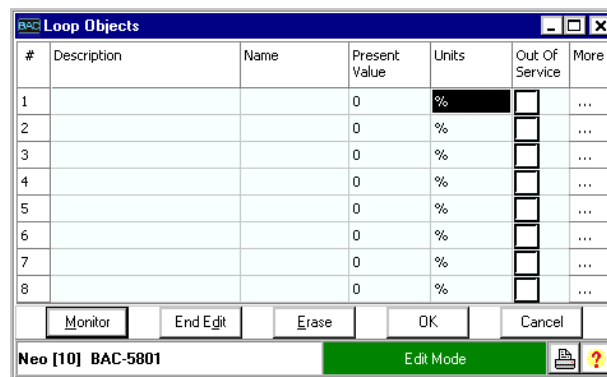
**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstag has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the loop object dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays output signals as *Present Value* changes. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the loop object to the original KMC settings.

**OK** Sends the new settings to the controller and closes the dialog box.

**Cancel** Closes the dialog box without making changes.

**Illustration 4-12 Loop object dialog**



**# (Loop number)** The loop object number. Loop objects are numbered sequentially within the KMC BACnet device.

**Description** A 32-character label of the object. The set of characters entered for *Description* must be printable characters.

**Name** A 16-character label of the object. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must be printable characters.

**Present Value** A numerical property that indicates the current value—in engineering units—of the output of the loop object.

**Units** Use *Units* to select the unit of measure for the output signal. BACstage supports several units of measure for loop objects. The available units are listed in the section [Supported engineering units on page 177](#).

**Out Of Service** When *Out Of Service* is checked, the output of the loop is not updating the loop object selected by *Manipulated*. Check *Out Of Service* to manually change *Present Value*.

**Manipulated** *Manipulated* selects either an analog output or value object to receive the output of the loop. By selecting an analog output object as the loop output, *Manipulated* can be used to control the position of a device connected to the corresponding output terminal of the controller. By selecting an analog value object for *Manipulated* and polling the object's present value in Control Basic, other objects—such as one or more binary outputs—can be controlled by a loop object.

**Controlled and Units** An analog input or value object whose present value is a condition under control. A typical application is to select an analog input object that represents a space temperature that is to be maintained at a setpoint.

**Setpoint Reference** Use *Setpoint* to select from the drop down list an analog input or value object whose present value represents the setpoint of the condition under control. Select *None* to enter a fixed setpoint in *Setpoint*.

**Setpoint** To enter a fixed setpoint for a loop object, first set *Setpoint Reference* to *None* and then enter a value.

**Action** The action of the loop. Action can be either direct acting or reverse acting.

- ◆ Direct acting loop objects *increase* the value of *Manipulated* as the value of *Controlled* rises above the value of *Setpoint*.
- ◆ Reverse acting loop objects *decrease* the value of *Manipulated* as the value of *Controlled* rises above the value of *Setpoint*.

**Proportional and Units** *Proportional* is the value of the proportional gain parameter used by the loop algorithm. It represents the amount of sensed

change—expressed in engineering units—that will cause the output to move from 0 to 100%.

**Integral and Units** Integral is the value of the integral parameter—expressed in hours or minutes— used by the loop algorithm. Integral adds a correction factor to the control loop based on how long the condition has been outside the setpoint. It specifies the number of times the magnitude of the error is added or subtracted to the output signal, over time, to eliminate the offset.

**Derivative and Units** *Derivative*—specified in minutes—slows the rate of change of the error. Use *Derivative* to reduce overshoot. If the error is changing at 1.0 per second (60/min) and the rate was .25 minutes then the derivative component would equal  $60 / \text{Min} \times .25 \text{ Min} = 15\%$ . This 15% would be added in over the 1 minute in a direction to reduce the rate of changing regardless of whether the input is above or below the setpoint.



Use *Derivative* only in systems without time lags. The input must start responding immediately to an output change. If there is a time delay the control loop will be unstable and will perform better without rate correction.

---

**Bias** the output value at setpoint. The bias is the value the controller will reach at equilibrium when derivative is not used.

**Priority** Sets the level in the priority array for the output of the loop. The default priority level is 8. See [Priority arrays on page 92](#).

**Delay** *Delay* defines a minimum period for a set of conditions to exist before a *TO-OFFNORMAL* or *TO-NORMAL* event occurs. Use *Delay* with *Error* to define conditions that indicate *Error* is out of an expected, predefined operating range. *Delay* is expressed in seconds.

**Event Enable** Use *Event Enable* to enable notifications for *TO-OFFNORMAL*, *TO-NORMAL* and *TO-FAULT*.

**Notification Class** Specifies the notification class to be used when handling and generating event notifications for a loop object. See [Notification object on page 77](#) for details about the notification class object.

**Notify Type** This property specifies whether the notifications generated by the loop object are *Events* or *Alarms*. BACstage displays a red message at the bottom of the work window when it receives an alarm; events are placed in the alarm/event list without displaying a message.

**Error Limit** This property sets the absolute magnitude that the difference between *Setpoint* and *Controlled* must exceed before a *TO-OFFNORMAL* event is generated.

**Min. Output** This property sets the minimum allowable value of the loop’s present value property. It is normally used to prevent the loop algorithm from controlling beyond the range of the controlled device.

**Max. Output** This property sets the maximum allowable value of the loop’s present value property. It is normally used to prevent the loop algorithm from controlling beyond the range of the controlled device.

## Schedule object

Use the schedule object dialog to enter and manage a periodic schedule that may recur during a range of dates. Schedules are divided into days, of which there are two types:

- ◆ Normal days are defined by the weekly schedule.
- ◆ Exception days are defined by exception schedules.

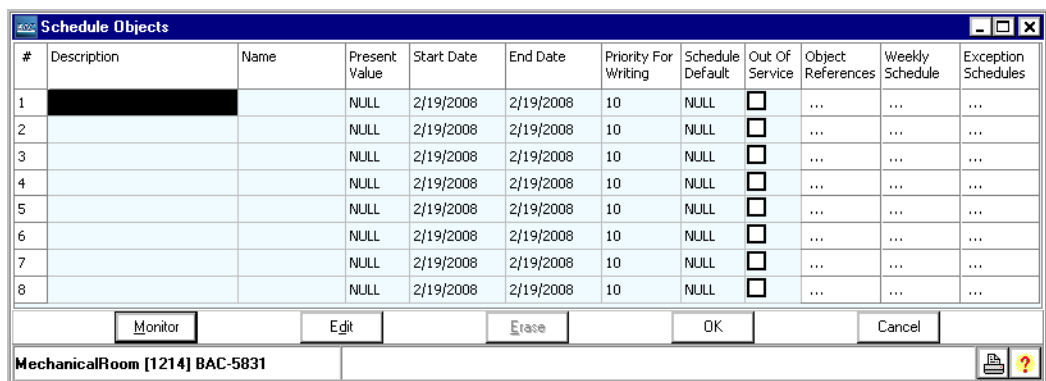
**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the schedule dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays schedule data as it is received from the controller. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the schedule object to the original KMC settings.

**OK** Sends the new settings to the controller and closes the dialog box.

**Cancel** Closes the dialog box without making changes.

**Illustration 4–13 Schedule object dialog**



**# (Schedule number)** The schedule object number. Schedule objects are numbered sequentially within the KMC BACnet device.

**Description** A 32-character label of the object. The set of characters entered for *Description* must be printable characters.



**Name** A 16-character label of the object. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must be printable characters.

**Present Value** This property indicates the value most recently written to a referenced object property from either a *Weekly Schedule* or one of the *Exception Schedules*. The value may be any valid analog or binary BACnet value.

**Start Date/End Date** Set the active period of the schedule with *Start Date* and *End Date*. Create seasonal schedules by defining several schedules with non-overlapping *Start* and *End Date* periods to control the same property references.

**Priority for Writing** Sets the priority for writing of referenced objects. See *Priority arrays on page 92*.

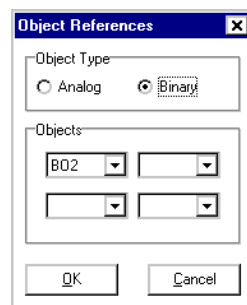
**Schedule Default** Each day in a weekly schedule covers a 24-hour period. *Schedule Default* defines the state of the schedule between 12:00 A.M. (midnight) and the first time slot in the weekly schedule. The present value of the schedule remains at the value of the last time slot until 12:00 A.M.

**Out of Service** When *Out Of Service* is checked, and sent to the controller, *Present Value* does not respond when a weekly schedule changes state.

**Object References** Enter up to four output or value objects that the schedule will control. When the current time and date are within the date range of the schedule *and* the day and time of the weekly schedule, the value associated with the day of the week and time in *Weekly Schedule* is assigned to *Present Value* in the referenced object.

- ◆ Reference objects must be within the same device as the schedule object.
- ◆ Referenced objects can be either analog or binary objects but not mixed within the same schedule.

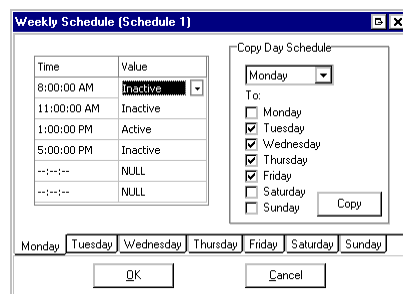
**Illustration 4-14 Object Reference dialog**



**Weekly Schedule** Click the ellipsis under the *Weekly Schedule* to open the *Weekly Schedule* dialog. Use *Weekly Schedule* to enter a list of values that— at specific times during each day of the week— are sent to *Present Value* in the referenced objects. See *Object References* for details about referenced objects.

- ◆ Times entered must be sequential starting with the earliest time of day.
- ◆ *Schedule Default* defines the state of the schedule's present value for the period between 12:00 midnight and the first time slot.
- ◆ Time/value pairs with *Null* as the value will be ignored.
- ◆ The present value of the schedule will remain at the value of the last time entered until 12:00 midnight.

**Illustration 4–15 Weekly Schedule dialog**



**Tip:**

BACstage displays six time/value pairs when connected to controllers with firmware 1.4. For firmware 1.3 and all earlier releases, BACstage will display only four time/value pairs.

*To duplicate a schedule*

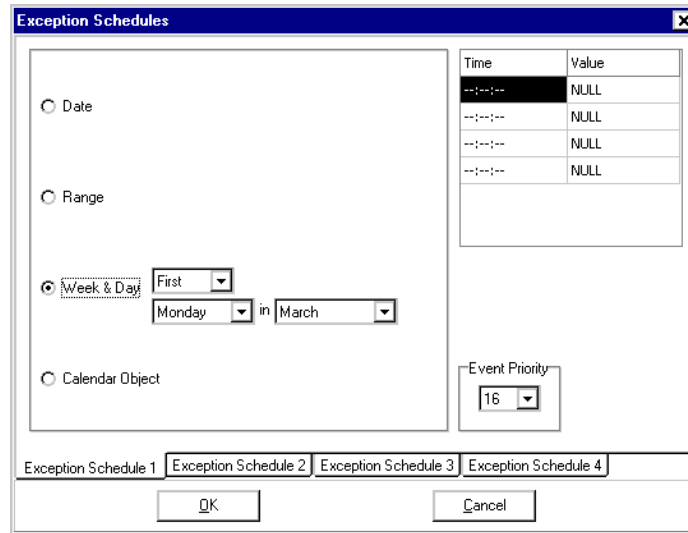
Once the time/value pairs are entered for any day of the week, that daily schedule can be duplicated in the other days of the schedule

To duplicate a daily schedule, do the following:

1. Enter the times and values for a day.
2. Under **Copy Day Schedule**, select the day with the original schedule from the text box.
3. Under **To:** select the check boxes for days to which you will copy the original schedule.
4. Click **Copy**.

**Exception Schedules** Use exception schedules to override the values in the weekly schedule. The *Exception Schedule* dialog includes the following choices for setting dates.

**Illustration 4-16 Exception Schedule dialog**



- ◆ **Date**—A single date on which the values and times listed in the exception schedule will override the values of the weekly schedule. Use the check boxes to select the same day each month.
- ◆ **Range**—A range of dates on which the values and times listed in the exception schedule will override the values of the weekly schedule. If *End Date* is empty, than all dates beginning with *Start Date* are considered valid dates in the range of dates. If *Start Date* is empty then all dates from the current system date up to *End Date* are considered to be valid dates in the range of dates.
- ◆ **Week & Day**—A day of the week and month on which the values and times listed in the exception schedule will override the values of the weekly schedule. If no month is selected, then the value entered in the exception schedule will override the weekly schedule on that day of the month for every month of the year.
- ◆ **Calendar Object**—If the date in the selected calendar object is *True* then the exception schedule will override the weekly schedule. See [Calendar object on page 76](#).
- ◆ **Event Priority**—Sets the order of precedence for conflicting exception schedules. For example if *Exception Schedule 2* has an event priority 8 and *Exception Schedule 4* has an event priority of 10 then *Exception Schedule 2* will override *Exception Schedule 4* when there is a conflict in the values for the reference object.

## Calendar object

Use the calendar object to enter and manage a list of special dates. These special dates may be holidays, special events or other days that require special attention on a calendar. See [Schedule object on page 72](#) for more information about setting up schedules.

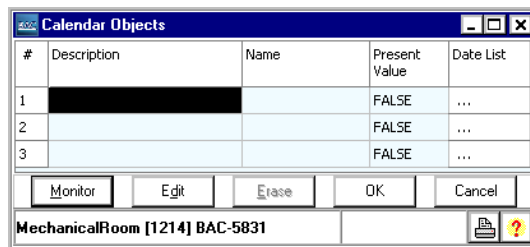
**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the schedule object dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays calendar information updates as they are received from the controller. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the calendar object to the original KMC settings.

**OK** Sends the new settings to the controller and closes the dialog box.

**Cancel** Closes the dialog box without making changes.

### Illustration 4–17 Calendar object dialog



**# (Calendar number)** The calendar object number. Calendar objects are numbered sequentially within the KMC BACnet device.

**Description** A 32-character label of the object. The set of characters entered for *Description* must be printable characters.

**Name** A 16-character label of the object. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must be printable characters.

**Present Value** This property indicates the current value of the calendar object. If the current system date and time is in the date list, *Present Value* is *True*. If the current date and time do not have a match in the date list, *Present Value* is *False*.

**Date List** The calendar object date list can include up to four entries. Each entry can be one of the following:

- ◆ **Date**—A single date. If the date matches the current system time and date *Present Value* of the calendar object is *True*.
- ◆ **Range**—If the current system date falls with the range of dates specified by *Range* then *Present Value* of the calendar object is *True*. If *End Date* is empty, than all dates beginning with *Start Date* are considered valid dates in the range of dates. If *Start Date* is empty than all dates from the current system date up to *End Date* are considered to be valid dates in the range of dates.
- ◆ **Week & Day**—If the current system time and date match the day of the week and month, then *Present Value* is *True*. If no month is selected, than *Present Value* is *True* on that day of the week for every month of the year.

## Notification object

Use the notification class object to manage the distribution and processing of alarms and events originating within a device. The notification object:

- ◆ Maintains a list of destination devices which are usually a BACnet operator workstation
- ◆ Sets the prioritization of TO-OffNORMAL and TO-NORMAL events by the destination device
- ◆ Designates if the event notification requires an acknowledgement
- ◆ Designates the process a recipient device should perform upon the receipt of an event.

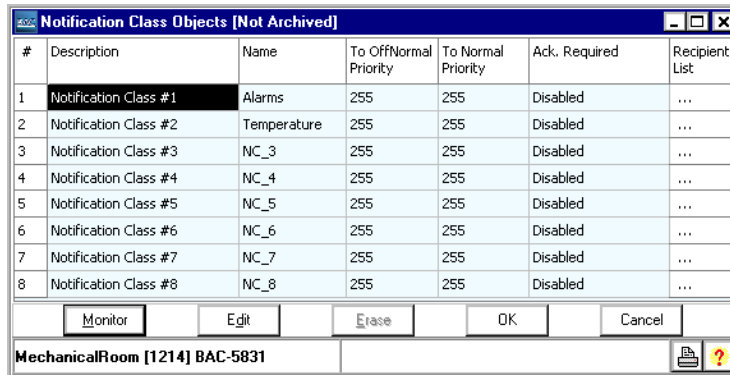
**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the notification class dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays notification class data as it is received from the controller. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the notification class object to the original KMC settings.

**OK** Sends the new settings to the controllers and closes the dialog box.

**Cancel** Closes the dialog box without making changes.

**Illustration 4–18 Notification class object dialog**



**# (Object number)** The notification class object number. Input objects are numbered sequentially within the KMC BACnet device and directly correspond to the controller’s input terminal.

**Description** A 32-character label of the object. The set of characters entered for *Description* must be printable characters.

**Name** A 16-character label of the object. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must be printable characters.

**To OffNormal Priority** Choose the priority for *OffNormal* event notification. The highest priority is 0; the lowest is 255.

**Table 4–9 Alarm and event priority**

Alarm and event priority	Network priority
00-63	Life safety message
64-127	Critical equipment message
128-191	Urgent message
192-255	Normal message

**To Normal Priority** Choose the priority for *ToNormal* event notification. The highest priority is 0; the lowest is 255.

**Ack. Required** Select the condition or conditions for which acknowledgement is required.

**Recipient List** The notification object recipient list may include up to four devices—each with a specific day and time—may be designated as recipients of the event.

- ◆ Valid Days—The days of the week on which this destination may be used between *From Time* and *To Time*.
- ◆ From Time/To Time—The window of time (inclusive) during which the destination is viable on the days of the week checked in Valid Days.
- ◆ Process Identifier—The handle of a process within the recipient device that is to receive the event notification.
- ◆ Transitions—A set of flags that indicate To OffNormal, To-Fault or To Normal for which the recipient is suitable.
- ◆ Device Instance—The destination of the device to receive notification.
- ◆ Issue Confirmed Notifications—Check when confirmed notifications are to be sent. Leave unchecked if confirmed notifications are not required.

## Event enrollment object

An event enrollment object is a standard BACnet object that monitors a property in another BACnet object for alarm or event conditions. When the condition is detected, a notification is sent to a notification class object for further handling.

**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the event enrollment object dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays the value of each property in the object. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

**Erase** Choosing *Erase* returns the properties in the value object to the original KMC settings.

**Cancel** Closes the dialog box without making changes.

**OK** Sends the new settings to the controllers and closes the dialog box.

Illustration 4–19 Event enrolment object

**Name** A 16-character label of the object and must be unique within the BACnet device that maintains it. The set of characters entered for **Name** must be printable characters.

**Description** A 32-character label of the object. The set of characters entered for **Description** must be printable characters.

**Object Property Reference** These parameters designate the referenced property. Enter the device instance number, the object number and the property to monitor.

- ◆ Object ID–The object that contains the property the even enrolment object is monitoring.
- ◆ Property–The property within the designated object. Typically the property is **Present Value**.

**Event Type** The Event Type property specifies which of the standard algorithms should be applied when monitoring the referenced object as entered under **Object Property Reference**. The selection of the Event Type changes the display of the parameter values needed for each algorithm. Each of the parameters are described in the following topics.

The selection of **Event Type** changes the display of the parameter values needed for each algorithm. Each of the parameters are described in the following topics.

**Bitmask** Applies when Event Type is set to *CHANGE OF BITSTRING*. The selected bits represent a bitmask that indicates which bits in the referenced property are to be monitored by the algorithm.



- ◆ A selected check box next to a bit indicates that the bit in the referenced property is to be monitored by the algorithm.
- ◆ A cleared check box next to a bit indicates that the bit in the referenced property is not significant for the purpose of detecting *Change Of Bitstring*.

**List of Bitstrings** This list defines the set of states for which the referenced property is *Off Normal*. Only the bits selected in **Bitmask** are significant. If the value of the referenced property changes to one of the values in the **List of Bitstring**, then the Event State property of the Event Enrollment object changes to *To Off Normal* and appropriate notifications are sent to the Notification Class object.

**List Of Values** Applies when **Event Type** is set to *CHANGE OF STATE*. If the value of the referenced property changes to one of the values in the List Of Values, then the value of Event State changes to *To Off Normal* and notifications are sent to the Notification Class object.

**Referenced Property Increment** This parameter represents the increment by which the referenced property must change to initiate an event.

**Time Delay** This parameter represents the time—in seconds—that the conditions monitored by the event algorithm must persist before an event notification is issued.

**Feedback Property Reference** This parameter applies when **Event Type** is set to *COMMAND FAILURE*. It identifies the object and property that provides the feedback to ensure that the commanded property has changed value. This property may reference only object properties that have enumerated values or are of type *BOOLEAN*.

**Setpoint Reference** This parameter applies when **Event Type** is set to *FLOATING LIMIT*. It indicates the setpoint reference for the reference property interval.

**High Limit** This parameter applies when **Event Type** is set to *OUT OF RANGE*. It defines the upper limit for a normal operating range of the monitored property in the referenced object.

**Low Limit** This parameter applies when **Event Type** is set to *OUT OF RANGE*. It defines the lower limit for a normal operating range of the monitored property in the referenced object.

**Deadband** This parameter applies when **Event Type** is set to *FLOATING LIMIT* and *OUT OF RANGE*. It specifies a range between the high limit and low limit properties in which the monitored property in the referenced object must remain before the object generates a notification.

**High Diff Limit** This parameter applies when **Event Type** is set to *FLOATING LIMIT*. When added to Setpoint Reference it defines an upper

limit for a normal operating range of the monitored property in the referenced object.

**Low Diff Limit** This parameter applies when **Event Type** is set to *OUT OF RANGE*. When added to **Setpoint Reference** it defines an lower limit for a normal operating range of the monitored property in the referenced object.

**Notification Threshold** This parameter applies when **Event Type** is set to *BUFFER READY*. It specifies the value of Records Since Notification at which notification occurs.

**List Of Life Safety Alarm Values** This parameter applies when **Event Type** is set to *CHANGE OF LIFE SAFETY*. If the value of the referenced property changes to one of the values in the list of Life Safety Alarm Values, then the value of **Event State** changes to *To Off Normal* and appropriate notifications are sent to the Notification Class object.

**Alarm Values** This parameter applies when **Event Type** is set to *CHANGE OF LIFE SAFETY*. It is a list of states that apply to the *CHANGE OF LIFE SAFETY* algorithm. If the value of the referenced property changes to one of the values in the **Alarm Values**, then the value of **Event State** changes to *To Off Normal* and appropriate notifications are sent to the Notification Class object.

**Mode Property Reference** This parameter applies when **Event Type** is set to *CHANGE OF LIFE SAFETY*. It identifies the object and property that provides the operating mode of the referenced object providing life safety functionality (normally the Mode property). This parameter may reference only object properties for BACnet Life Safety.

**Table 4–10 Event Types, Event States, and Event Parameters**

<b>Event Type</b>	<b>Event State</b>	<b>Event Parameters</b>
CHANGE OF BITSTRING	NORMAL	Time Delay Bitmask
	OFFNORMAL	List Of Bitstring Values
CHANGE OF STATE	NORMAL	Time Delay
	OFFNORMAL	List Of Values
CHANGE OF VALUE	NORMAL	Time Delay
	OFFNORMAL	Bitmask Referenced Property Increment
COMMAND FAILURE	NORMAL	Time Delay
	OFFNORMAL	Feedback Property Reference

**Event Types, Event States, and Event Parameters (continued)**

<b>Event Type</b>	<b>Event State</b>	<b>Event Parameters</b>
FLOATING LIMIT	NORMAL HIGH LIMIT LOW LIMIT	Time Delay Setpoint Reference Low Diff Limit High Diff Limit Deadband
OUT OF RANGE	NORMAL HIGH LIMIT LOW LIMIT	Time Delay Low Limit High Limit Deadband
BUFFER READY	NORMAL	Notification Threshold
CHANGE OF LIFE SAFETY	NORMAL OFFNORMAL LIFE SAFETY ALARM	Time Delay List Of Alarm Values List Of Life Safety Alarm Values Mode Property Reference

**Notify Type** This property specifies whether the notifications generated by the object are **Events** or **Alarms**. Alarms and events notifications are handled differently by the device—usually a workstation—that receives the notification.

**Notification Class** Specifies the notification class object to use when handling and generating event notifications for this object. See [Notification object on page 77](#) for details about the notification class object.

**Event Enable** Use **Event Enable** to enable notifications for **To Off Normal**, **To Normal** and **To Fault**.

**Acked Transitions** This property controls three flags that separately indicate the receipt of acknowledgements for **To Off Normal**, **To Fault**, and **To Normal** events.

**Event State** Use **Event State** to determine that this object has an active event state associated with it. The possible values for Event State are listed in [Event Types, Event States, and Event Parameters on page 82](#).

**Event Time Stamps** Holds the times of the last event notifications of **To Off Normal**, **To Fault** and **To Normal** events.

## Working with alarms and events

BACstage and the KMC BACnet controllers support intrinsic event and alarm reporting. This method of generating event and event notification is

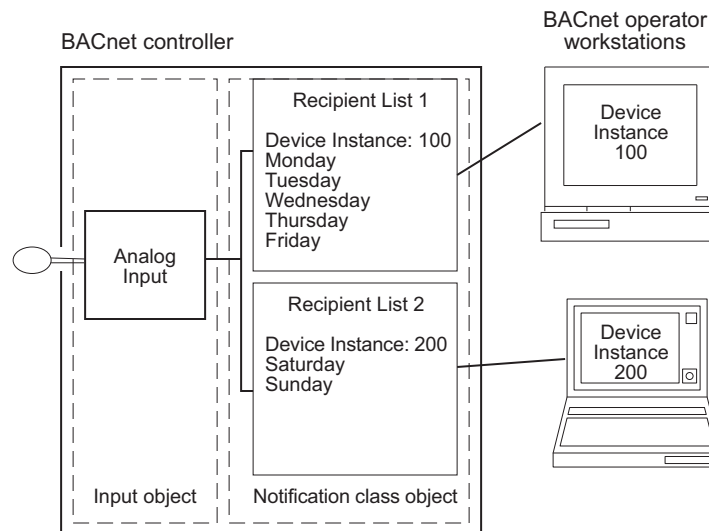
configured using only standard BACnet properties within the objects that support events and alarms.

### An example notification

In the example shown in the diagram *The intrinsic alarm and event notification flow on page 84*, a thermistor is connected to an analog input object. The object is configured to generate events when the temperature input moves outside of a predefined temperature range. When the temperature makes the transition outside of the normal range, the following actions take place:

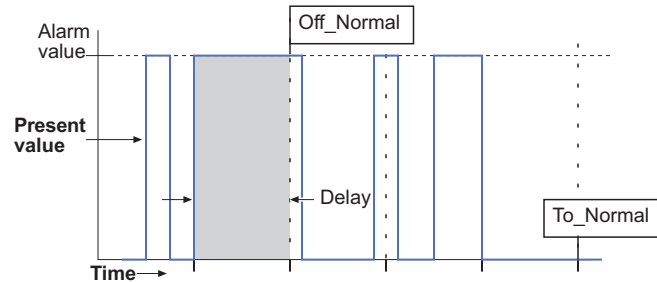
- ◆ The input object sends the event to the designated notification class object for further handling.
- ◆ On weekdays, Recipient List 1 directs the event to the computer with workstation configured as device instance #100.
- ◆ On weekends, Recipient List 2 sends the event to the computer with workstation configured as device instance #200.
- ◆ The operator workstation processes the event. The exact actions that take place at the operator workstation depend upon the capabilities of the workstation and the type and priority of the notification.

**Illustration 4–20 The intrinsic alarm and event notification flow**



### Notifications in binary objects

The following illustration is an example of a binary object configured to generate events when present value falls outside of predefined limits.

**Illustration 4-21 Example of Off\_Normal and Normal events**

Conditions for generating a *To Off Normal* event in a binary object.

- ◆ *Present Value* must maintain the value specified by *Alarm Value* for the period specified by *Time Delay*, and
- ◆ The selection in *Event Enable* must include *To Off Normal*.

Conditions for generating a *To Normal* event in binary objects:

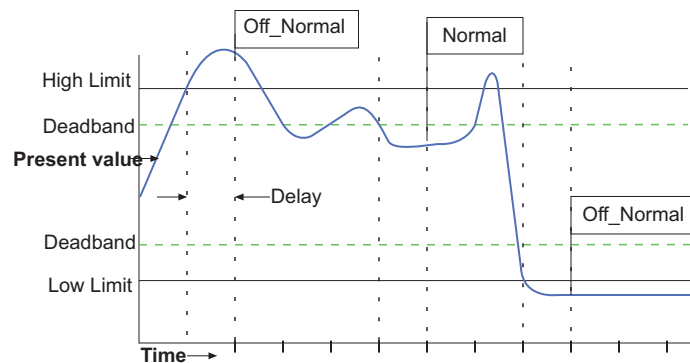
- ◆ *Present Value* must remain unequal to the value specified by *Alarm Value* for the period specified by *Time Delay*, and
- ◆ The selection in *Event Enable* must include *To Normal*.

### Notifications in analog objects

Each BACnet object that supports notifications has within it properties that define conditions under which an event is initiated.

- ◆ When the monitored value makes a transition from normal conditions to a value outside of normal conditions, the object sends a *To Off Normal* event to a notification class object.
- ◆ *To Normal* events are generated when present value makes the transition from outside the defined range to normal conditions.

The following illustration is an example of an analog object configured to generate events when present value falls outside of predefined limits.

**Illustration 4–22 Example of Off\_Normal and Normal events in analog objects**

Conditions for generating a *To Off Normal* event when *Present Value* exceeds normal limits:

- ◆ *Present Value* must be either greater than *High Limit* or less than *Low Limit* for the period specified by *Time Delay*, and
- ◆ *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *Off Normal*.

Conditions for generating a *To Normal* event when *Present Value* returns to a normal value:

- ◆ *Present Value* must return to a level that is less than *High Limit* minus *Deadband* and greater than *Low Limit* plus *Deadband* for the period specified by *Delay*, and
- ◆ *Low/High* must be selected in *Limit Enable*, and
- ◆ The selection in *Event Enable* must include *NORMAL*.
- ◆ *Delay* and *Deadband* provide a buffer that ignores minor fluctuations in *Present Value*.

### About the notification class object

BACnet notification class objects route events from objects-in-alarm to destination devices. A notification class object:

- ◆ Designates the priority for the handling device to process the alarm.
- ◆ Includes recipient lists that designate—by device instance—the devices that will handle the event. Each recipient list includes also a day of the week and a time span. By configuring multiple recipient lists for different days of the week or periods of time, events can be sent to the workstations or devices where appropriate action can be taken by operators.
- ◆ Designates that an acknowledgement is required from an operator at a BACnet Operator Workstation such as BACstage.

### Configuring notification class objects

At least one notification class object must be configured within the same device as the object-in-alarm. See [Notification object on page 77](#) for complete details about the notification class object.

1. Select a priority for both *To OffNormal* and *To Normal* transitions.
2. Choose operator acknowledgment from *Ack.Required*.
3. Open the recipient list and set the following:
  - Select the day of the week and time for notification.
  - Enter the device instance number of the BACnet operator workstation that will receive the event. For BACstage, this is *Our Device Instance* in BACdoor.
  - In *Transitions*, select the state that triggers the event. This must match the selection under *Event Enable* in the properties of the object-in-alarm.
  - Check *Confirmed Notifications*. For BACstage, this must be checked.
4. Close the recipient lists and end the edit.

### Configure the object for events and alarms

Events and alarms are initiated in an object by setting the following parameters:

1. Select the notification class object that will receive the event.
2. Select *Event Enable* transition state for notification. This must match *Transitions* in the recipient list.
3. Select *Notification Type*. The manner in which BACstage displays events and alarms is described in the topic [Alarms/Events on page 170](#).
4. Configure the conditions that will trigger the event or alarm:
  - For binary objects select *Active* in the alarm property.
  - For analog objects select the required *Limit Enable* setting and then set the values of the limit.
5. End the edit.

### Viewing events and alarms in BACstage

Within BACstage are two methods to view events and alarms.

- ◆ See [Alarms/Events on page 170](#) to acknowledge alarms and view a complete list of events and alarms received by BACstage.
- ◆ See [Alarm/Event Summary on page 37](#) to view alarms and events in an individual device.

## Trend object

Setup and managed trend logs with the trend object dialog. Use trend logs to monitor the present value of one or more objects within a device. Each trend log periodically saves the data record along with a timestamp and relevant status information at the time the controller saved the record to a trend log.

See also [Viewing trend logs on page 90](#) for details about the trend log display.

### Tip:

For KMC BACnet controllers with firmware 1.4 or later, the trend object will hold 256 samples. Earlier versions of firmware hold 128 samples.

**Edit and Monitor** When BACstage is in edit mode, the values and settings are editable but, BACstage has not yet sent them to the controller. When either *End Edit* or *Ok* are clicked, all of the values displayed in the trend object dialog are sent to the controller. Choosing *Monitor* changes the mode to monitor and displays trend data as it is received. BACstage indicates the edit mode is active by turning the status block green; monitor mode is indicated by blue.

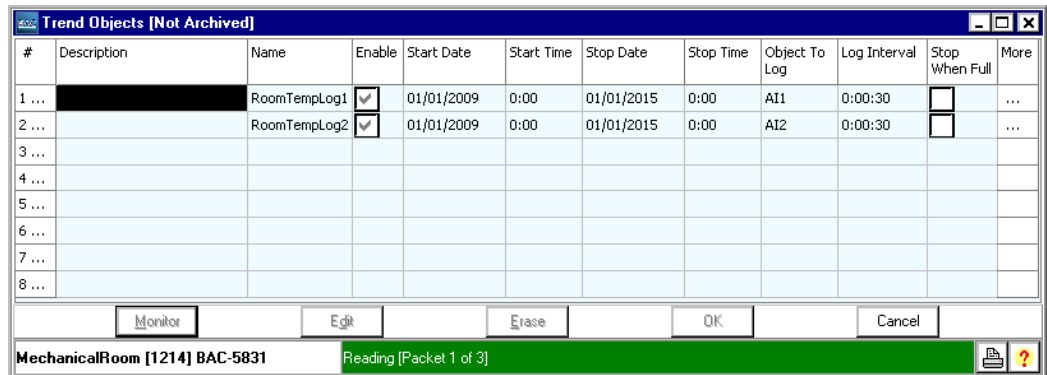
**Erase** Choosing *Erase* returns the properties in the trend object dialog to the original KMC settings.

**OK** Sends the new settings to the controller and closes the dialog box.

**Cancel** Closes the dialog box without making changes.



Illustration 4-23 Trend object dialog



**# (Trend number)** The trend object number. Trend objects are numbered sequentially within the KMC BACnet device. To view a trend chart, click the ellipsis (...) next to the trend number.

**Description** A 32-character label of the object. The set of characters entered for *Description* must be printable characters.

**Name** A 16-character label of the object. *Name* must be unique within the BACnet device that maintains it. The set of characters entered for *Name* must be printable characters.

**Enable** If unchecked, the trend object does not log data. If *Enable* is checked and the current time and date are within the range of time and dates specified by *Start Time/Date* and *Stop Time/Date* the trend object logs data.

**Start and Stop Date and Time** Set the period for logging data with *Start* and *Stop Time* and *Date*.

**Object to Log** Choose the object within the BACnet device whose present value is to be the source of the data for the trend log. The object may be an input, output, value or accumulator object.

**Log Interval** This property, specifies the periodic interval—in hundredths of seconds—for which the data from *Object to Log* is to be logged.

**Stop When Full** When checked, once the buffer is full the trend object stops adding new records to the trend log buffer. If unchecked, the oldest data record is replaced with a new record.

**Notification Threshold** When the number of records reaches the value of *Notification Threshold*, a notification is sent to the notification class object specified for this trend.

**Event Enable** Use *Event Enable* to enable notifications for *TO-OFFNORMAL* and *TO-NORMAL*.

**Notification Class** Specifies the notification class object within the device that will process notifications from the trend object. See [Notification object on page 77](#) for instructions on configuring the notification class object.

**Notify Type** Specifies whether the notification generated by the trend object is an alarm or event.

**COV Resubscription Interval** If the trend log is acquiring data from a remote device by COV subscription and COV subscription is in effect, this property specifies the number of seconds between COV resubscriptions.

If COV subscriptions are in effect, the first COV subscription is issued when the trend log object begins operation or when **Log Enable** becomes *True*. If present, the value of this property must be non-zero. If this property is not present, then COV subscription cannot be attempted.

**Client COV Increment** If the trend log object is acquiring COV data, this property specifies the increment to be used in determining that a change of value has occurred.

## Viewing trend logs

To view either the current or a historical trend chart, open the trend object dialog from the *Object Menu* and then click the ellipsis (...) next to the trend number. To alter the view of the chart, right-click and use the trend object pop-up menu.

**Illustration 4–24 Viewing a trend chart**

Click the ellipsis (...) to view a trend chart.

#	Description	Name	Enable	Start Date	Start Time	Stop Date	Stop
1 ...		RoomTempLog1	<input checked="" type="checkbox"/>	01/01/2009	0:00	01/01/2015	0:00
2 ...		RoomTempLog2	<input checked="" type="checkbox"/>	01/01/2009	0:00	01/01/2015	0:00
3 ...							
4 ...							
5 ...							
6 ...							
7 ...							
8 ...							

Buttons: Monitor, Edit, Erase, OK

MechanicalRoom [1214] BAC-5831 Reading [Packet 1 of 3]

**Refresh** Loads the latest trend data from the controller and redraws the screen.

**3D** Toggles between two-dimensional and three-dimensional views of the displayed data.

**Show Values** Displays the recorded values on the chart.

**Show Text** Displays the recorded numeric values and the time the controller recorded the values. A status condition may take the place of a numeric value in the *Value/Status* column.

## About objects, properties and services

**Zoom and Undo Zoom** To zoom in on an area of the chart, left click and drag from left to right over the area of the chart you wish to examine. Choose *Undo Zoom* to return to the full view of the chart.

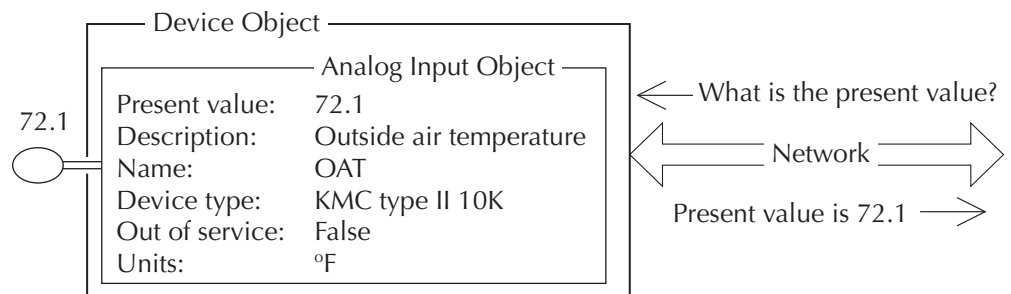
**Clear** Clears the trend chart from the memory of the controller.

Objects are the means by which a BACnet device represents information that can be observed or changed. An object may represent a physical point such as an input or output or a logical grouping of data such as a PID loop, schedule or variable. Objects are a method of organizing and accessing data in a way that corresponds to real-world inputs and values.

### An example object

The example object in the illustration [Object model on page 91](#) is a typical analog input object found in a BACnet controller. The physical input is a thermistor connected to the input of the controller. In this example, the present value property corresponds to the actual room temperature. Other properties label the object with a name, give a description to the input, assign a unit of measure, designate the type of input connected to the object, and if the object is in-service.

**Illustration 4-25 Object model**



The BACnet standard strictly defines available objects, their properties, and the acceptable values for each property. Because each type of object has the same set of required properties, and the properties follow the same rules about what values they can be, the data that the property represents is generally accessible to any BACnet process that requests it.

### Services

BACnet devices use services to acquire information from another device, command another device to perform certain actions, or announces to one or more devices that some event has taken place. Examples of services include scheduled commands and alarms between BACnet devices. Some services read or write properties of objects in the receiving device. Other services

convey notification of alarms or other special events, still others read and write files. The services provided by a BACnet device are described by the device's PIC. statement.

In the object model shown in the illustration [Object model on page 91](#) the read property service is shown as the question “What is the present value?” The analog input object responds with “Present value is 72.1”.

## Priority arrays

BACnet devices use the priority array property to control *Present Value* in certain objects. For KMC BACnet devices this property is part of both analog and binary output and value objects. The priority array property maintains order when several commands are simultaneously issued to change a present value property. For example, an operator may enter a command to stop a fan when a schedule is commanding it to run. By programming the command from the operator at a higher priority, the priority array property permits the operator command to take precedence over the schedule.

Priority array properties have 16 levels associated with them. Priority 1 is the highest; priority 16 is the lowest. When a command is issued for a present value property of an output or value object, rather than directly affecting the present value, the object stores the value in its priority array property at the appropriate priority level. The command with the highest priority sets the present value of the object.

### An example priority array

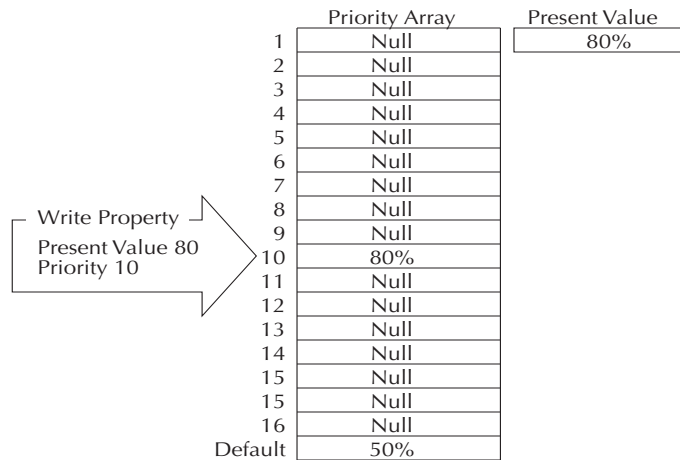
Initially, all levels of a priority array are filled with a *Null* value. The default value is entered in *Relinquish Default* in the object menu. In this example *Relinquish Default* equals 50%.

#### Illustration 4–26 Initial state

	Priority Array	Present Value
1	Null	50%
2	Null	
3	Null	
4	Null	
5	Null	
6	Null	
7	Null	
8	Null	
9	Null	
10	Null	
11	Null	
12	Null	
13	Null	
14	Null	
15	Null	
16	Null	
Default	50%	

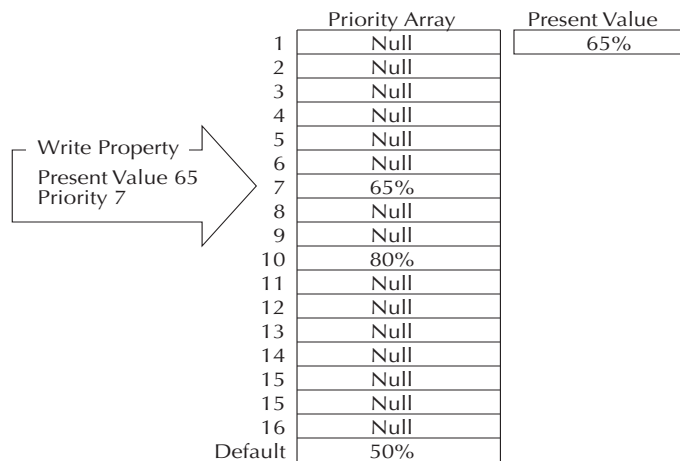
Then, a write property command with a value of 80% at a priority level of 10 is sent to the object. Because this new priority is a higher priority than the default level, the present value changes to 80%. Note that the array default value remains in the array.

**Illustration 4-27 First write property command**



Next, a write property command with a value of 65% at a priority level 7 is sent to the object. Since this new command has a higher priority than the previous level 10 priority, the present value becomes 65%. As before, the previous values remain in the priority array.

**Illustration 4-28 Second write property command**



At this point, if a write property command is sent to the object with a *NULL* value at priority 7, this relinquishes the priority 7 control at that priority. The

output reverts to the next highest priority, which in this example is the 80% value at level 10.

The same principles for controlling analog objects hold true for binary objects, the only difference is that the values for binary objects are *Inactive (0)* or *Active (1)* and are referred to as numerical values.

To manipulate the write priority of an object with Control Basic, see the Control Basic keyword [RLQ on page 139](#).

### Standard BACnet priority levels

Some priorities are designated by the BACnet standard. For example, Priority 1 is reserved for use by life/safety systems and Priority 8 is reserved for manual operator commands. The intent of standardizing the meanings of various priority levels encourages a consistent application of those priorities by various vendors across a multitude of facility types where the objectives of the programming cannot be predicted in advance.

**Table 4–11 BACnet standard priorities**

Priority Level	BACnet Standard Priority
P1	Manual-Life Safety
P2	Automatic-Live Safety
P3	
P4	
P5	Critical Equipment Control
P6	Reserved for minimum On/Off time
P7	
P8	Manual Operator
P9–P16	

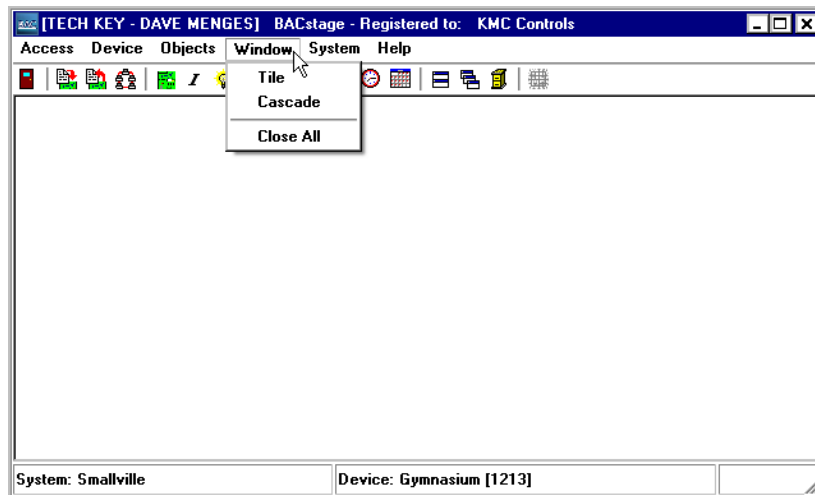
Even with these designations, the standard meanings are subject to interpretation and it is often a local decision as to how priority levels are applied. The assignment of specific meanings to the priorities is site-dependent and represents the objectives of the building's owner and management. However, to maintain interoperability, system programmers should apply priorities consistently across all controllers within a facility.

## Section 5: The Window menu

The Window menu sets the order in which BACstage displays pop-up menus.

---

Illustration 5–1 Window menu



**Cascade** Arrange all open windows to be visible in a stack.

**Tile** Arranges and resizes all open windows to fill the available space.

**Close All** Closes all open BACstage windows.





## Section 6: **About Control Basic programs**

Control Basic is the process that creates the automation in KMC controllers. Topics in this section cover the rules for writing Control Basic programs.

---

Every KMC controller includes space for Control Basic programs. Within each controller a program continuously evaluates input data from the building automation system. Then, based upon the instructions in the program, the controller takes action to keep one or more pieces of equipment operating within required parameters.

The instructions within the program object are written in Control Basic, a programming language that is similar to BASIC (Beginner's All-purpose Symbolic Instruction Code). In addition to standard BASIC programming functions, it includes specialized functions specific for the building automation controls industry.

Each of the following topics cover a key aspect of Control Basic.

- ◆ *Using the Control Basic editor on page 97*
- ◆ *About Control Basic scans on page 100*
- ◆ *Programming format and notation on page 105*
- ◆ *Labels and line numbers on page 102*
- ◆ *Real numbers on page 105*
- ◆ *Hierarchy of operators on page 105*
- ◆ *Relational operators on page 107*
- ◆ *Using arithmetic operators on page 106*
- ◆ *Using Boolean logic on page 107*
- ◆ *Programming with variables on page 108*
- ◆ *Transferring values between BACnet controllers on page 109*
- ◆ *Programming with mnemonics on page 110*

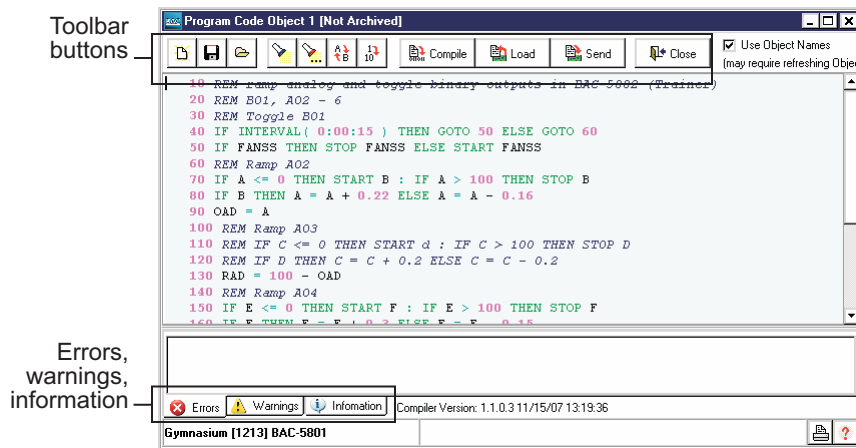
### **Using the Control Basic editor**

When you enter Control Basic Editor window, BACstage opens an empty window with a flashing cursor. To enter a line, type a line number followed by a space and then a statement.

To open the Control Basic editor, do the following:

1. From the **Objects** menu choose **Basic Programs**.
2. In the **Program Objects** dialog click the number of the program to edit under the # symbol.
3. Enter the program and then do either of the following:
  - Click **Compile** to test the program for proper syntax.
  - Click **Send** to save the program in the controller.

**Illustration 6–1 Control Basic editor window**



**Note:** Program lines are not checked for errors until you choose either *Send* or *Compile*. If errors are found in the program, the program cannot be sent to the controller until the error is corrected.












#### Related topics

- ◆ [Writing Control Basic statements on page 101](#)
- ◆ [Transferring values between BACnet controllers on page 109](#)
- ◆ [Using arithmetic operators on page 106](#)
- ◆ [Relational operators on page 107](#)
- ◆ [Using Boolean logic on page 107](#)
- ◆ [Programming with variables on page 108](#)
- ◆ [Programming with mnemonics on page 110](#)
- ◆ [Real numbers on page 105](#)
- ◆ [Programming format and notation on page 105](#)
- ◆ [Keywords for Control Basic on page 113](#)

### Toolbar buttons

Entering a Control Basic program is similar to using a text editor. Toolbar buttons assist with file opening and closing and cutting, copying and pasting text.

**Table 6-1 Code Editor toolbar button**

Action	Toolbar	Description
Clear		Clears the Control Basic program area.
Save to File		Saves the Control Basic file to disk. Standard Control Basic is saved with a .BAS extension. Next Generation Basic is save with an .NG extension.
Open		Opens a Control Basic file stored on disk.
Find		Use to find a word or phrase.
Find next		Finds the next word or phrase entered in Find.
Replace		Searches for a word or phrase and replaces it with another word or phrase.
Renumber		(Standard Control Basic Only) Renumbers the program starting with the first line and incrementing by 10.
Compile		Tests the program for proper syntax but does not send it to the controller.
Load		Retrieves and displays the program from the current controller.
Send		Tests the program for proper syntax and then, if it is correct, sends it to the controller. After the program is sent to the controller, it is stored in the controller's non-volatile memory. A message is displayed with the size of the program and to which program area it was sent.
Close		Stops the controller from running the program.

### Using the editor shortcut keys

Use the shortcuts in the table [Shortcut keys on page 100](#) when entering Control Basic programs. **Cut**, **Copy**, **Paste**, and **Select All** are accessible by right-clicking in the program listing.

**Table 6–2** Shortcut keys

Shortcut	Action
Cut (ctrl-x)	Permanently removes the selected text.
Copy (ctrl-c)	Copies the selected text to the clipboard for pasting in another location.
Paste (ctrl-v)	Moves text from the clipboard to the selected location.
Undo	Reverses last action.
Select All	Selects all text in the program.

### Programming with object and device names

When **Use Object Names** is selected, objects and devices can be referenced by their BACnet name.

- ◆ For details on using device and object names, see [Programming with names on page 103](#).
- ◆ The device or object name must be in the object list. See [Objects List on page 166](#) to add names to the object list.

### Errors, Warnings and Information

These three tabs lists information about the program after it is compiled or an operator has attempted to compile a program.

- ◆ Errors are problems that prevent the program from compiling.
- ◆ Warnings are problems in programs the compile but may not execute correctly.
- ◆ Information is additional details about a compiled program.

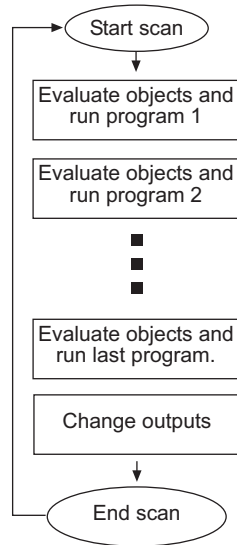
## About Control Basic scans

Control Basic is the process that creates the automation in a KMC controller. Each controller has several program objects for storing and executing Control Basic instructions. When running Control Basic programs, the microprocessor in the controller does the following:

1. Evaluates the state of each object.
2. Executes the Control Basic programs.
3. Changes the state of all outputs when all programs have been executed.

This process—referred to as a scan—is normally performed several times a second. See the illustration *The scan process* on page 101 for an example of the scanning process.

**Illustration 6-2 The scan process**



The processor evaluates all program areas before making changes. For example, if programs 1, 3 and the last program includes instructions for Lights ON, and programs 2 and 4 had instructions Lights OFF, the lights will not flash, they will only be set to ON at the end of the scan.

**Tip:** Program the most important events in the highest numbered program area. This prevents programs with less importance from overriding critical actions.

**Writing Control Basic statements**

Control Basic programs are entered with the BACstage Control Basic editor. See the topic *Basic Programs* on page 67 for details on the editor.

**Multiple statements**

Multiple statements can be used on the same program line, but must be separated by a colon.

```
10 A=B:F=C+D
```

**Functions**

A function is a keyword that—when evaluated by Control Basic—returns a value. This returned value is the result computed by the function. Functions

save time for complex calculations such as calculating square roots. They may also be used to retrieve common system data such as time.

### Expressions

A Control Basic expression describes a symbol or combination of symbols which represent a numeric value. Expressions may take the form of a simple equality such as  $A=7$  or a comparison between symbols such as  $X < Y$ . Expressions can be derived also from a function such as *TIME*, a controller point such as AI2 (analog input 2), or by the result of a series of calculations such as  $A * B - AI2 - 2 / 9$ .

An expression must evaluate to a real number.

**Table 6–3 Examples of expressions**

Expression	Example
Functions	Time, DOW, DOY, etc.
Local Variables	A through Z
BACnet objects	AI1, BI1, AO2, BO4, etc.
KMD Controller Points	OUT1, IN3, VAR16, etc.

## Labels and line numbers

Depending on the model of the controller, Control Basic programs will use either line numbers or labels.

### *Standard Control Basic*

- ◆ BAC-7000 series controllers
- ◆ BAC-5800 series controllers

### *Next Generation Control Basic*

- ◆ BAC-A1616 Building Controller
- ◆ BAC-10000 FlexStat Controllers

### Line numbers—Standard Control Basic only

When writing programs for controllers that support Standard Control Basic, enter a line number at the beginning of each line. Each Control Basic program line must include a line number and at least one function or statement.

```
10 A = B
20 P = PI
```

Programs written for controllers with Next Generation Control Basic do not use program lines.

### Labels in Next Generation Control Basic

Labels are used instead of line numbers with Next Generation Control Basic. Use labels when redirecting program flow with the following statements:

- ◆ [GOSUB on page 123](#)
- ◆ [GOTO on page 124](#)
- ◆ [ONERROR on page 135](#)
- ◆ [ON GOSUB on page 133](#)
- ◆ [ON GOTO on page 134](#)

Declare a label by typing a name followed immediately by a colon (:).

- ◆ A label can be any combination of letters (A-Z or a-z), numbers (0-9) or the underscore (\_).
- ◆ Labels are not case sensitive.
- ◆ Labels are unique to the program in which they are declared.
- ◆ A label cannot duplicate a keyword, constant, local variable or alias.

In the following program example, *CoolMode* and *HeatMode* are program labels.

```

IF T > 55 THEN GOTO CoolMode
IF T <= 55 THEN GOTO HeatMode
END
CoolMode:
REM Cooling sequence runs here
END
HeatMode:
REM Heating sequence runs here
END

```

## Programming with names

When writing Control Basic programs, you may use names to identify devices and objects on the BACnet internetwork. The following Control Basic examples function identically; the first uses names and the second example uses mnemonics.

With the *Object Names* check box selected

```
50 IF BLR THEN RLQ HWP@9 ELSE HWP = 0
```

With the *Object Names* check box is not selected

```
50 IF BO1 THEN RLQ AO2@9 ELSE AO2 = 0
```

Object names are always stored in the memory of the controllers on a network. However, to keep transfer time to a minimum, BACstage also stores the object names on the computer. If the system is programmed from more than one computer then the internal list may become out-of-date. To update the current list of object names choose the *Objects List* from the System menu.

### Control Basic compliant names

Device or object names may be substituted for a device instance or mnemonic of an object if the names meet the following criteria:

- ◆ Are composed only of the letters (A-Z or a-z), numbers (0-9) or the underscore (\_).
- ◆ Are not Control Basic keywords
- ◆ Are not Control Basic mnemonics
- ◆ The names are in the object list.

Examples of a BACstage local name

```
10 A = Space_Temperature
20 Boiler@7 = 1
```

Example of reading an object value from a remote device

```
10 A = MechanicalRoom.TempMechRoom
20 A = 1214.TempMechRoom
30 A = MechanicalRoom.AI1
```

### Noncompliant Control Basic names

If a device or object name is not fully Control Basic compliant, enclose both names in quotation marks (""). Noncompliant Control Basic names must meet the following criteria:

- ◆ The name is in the object list.
- ◆ The name does not contain a period(.).

```
10 A = "Dewpoint"
20 B = "Gymnasium.Time!"
30 AV40 = "1213.Time!"
```



### Invalid Control Basic names

Names that include periods (.) can only be compiled by using a reference to devices by their instance numbers and to the objects by their mnemonics. Any other name that does not compile will also have to be referenced the same way.

See also [Writing Control Basic statements on page 101](#) and [Transferring values between BACnet controllers on page 109](#).

## Programming format and notation

Control Basic programs consist of a series of numbered lines. On each line there are one or more statements.

Throughout these instructions the following notations are used to describe formats:

**Table 6-4** Typographic conventions

CAPS	Words in capital letters are key words and must be entered as shown.
lowercase	Items in lowercase letters represent information such as expressions that you supply.
...	An ellipsis (...) indicates that an item may be repeated as many times as necessary.
spaces ( )	Required spaces in syntax are illustrated with an underline ( ).
:	A colon (:) separates statements on the same line.
[ ]	Optional items are shown in brackets [ ].
	All other punctuation, including commas, are part of the syntax and must be included as shown in each example.

## Real numbers

Real numbers are any logical number between  $-3.4 * 10^{38}$  and  $3.4 * 10^{38}$ . Notation of the number is recognized in any of the following formats:

- ◆ Whole numbers (100)
- ◆ Decimal format (.0000123)
- ◆ Engineering notation (7.879 E-12)

## Hierarchy of operators

Control Basic arithmetic operators have an order of precedence. When several operation take place in the same program statement, some operations are performed before others. Control Basic uses the operator-precedence shown in the illustration [Order of operator precedence on page 106](#) when performing

operations on an expression. Operations at the same level of precedence are evaluated from left to right.

**Illustration 6–3 Order of operator precedence**

Operator	Type	Precedence	
( )	Expression in parenthesis	Highest (performed first)	
NOT	Logical NOT		
^	Exponentiation		
*, /	Multiplication and division		
\	Integer division		
MOD	Modulus (remainder)		
+, -			
<, >, <=, >=	Relational		
=, <>			
AND	Logical		
OR			
XOR			
			Lowest (performed last)

*Related topics*

- ◆ [Using arithmetic operators on page 106](#)
- ◆ [Relational operators on page 107](#)
- ◆ [Using Boolean logic on page 107](#)

## Using arithmetic operators

Operators are listed in their order of priority. The formats for using operators are listed in the table [Arithmetic order of precedence on page 106](#).

**Table 6–5 Arithmetic order of precedence**

Symbol	Operation	Example
*	Multiplication	2*4=8
/	Division	10/4 =2.5
\	The integer portion of a division	13\5= 2
MOD	The remainder of a division	13 MOD 5=3
+	Addition	2+2=4
-	Subtraction	4-3=1
^	Exponentiation Raises a value to a power	A = AI1 ^ AV1

*Related topics*

- ◆ [Relational operators on page 107](#)

- ◆ [Using Boolean logic on page 107](#)
- ◆ [Hierarchy of operators on page 105](#)

## Using Boolean logic

Control Basic recognizes four logical operators. The operators are listed in their order of precedence.

**NOT** NOT is a Boolean operator that performs a logical NOT operation on an expression. If the expression is 0, the result is 1. If the expression is non-zero, the result is 0.

For additional details on this operator, see the keyword [NOT on page 133](#).

**AND** AND performs the logical *AND* of the two expressions. The result is *true* if both expressions are non-zero; otherwise, the result is *false*.

For additional details on this operator, see the keyword [AND on page 115](#).

**OR** OR performs the logical *OR* of the two expressions. The result is *true* if either expression is *true*. The result is *false* if both expressions are *false*.

For additional details on this operator, see the keyword [OR on page 136](#).

**XOR** XOR performs the logical *exclusive or* of the two expressions. The result is *true* if the two expressions are different; otherwise, the result is *false*.

For additional details on this operator, see the keyword [XOR on page 149](#).

### Related topics

- ◆ [Using arithmetic operators on page 106](#)
- ◆ [Relational operators on page 107](#)
- ◆ [Hierarchy of operators on page 105](#)

## Relational operators

Relational operators are used to compare two values. The result is *true* if the comparison is *true*; otherwise, the result equals *false*. This result can then be used to make a decision regarding program flow. All relational operators have the same level of precedence.

**Table 6–6 Relational operators**

Operator	Relation Tested	Example	Result
=	Equality	5 = 2	False
<>	Inequality	5 <> 2	True
<	Less than	5 < 2	False

**Relational operators (continued)**

>	Greater than	5 > 2	True
<=	Less than or equal to	5 <= 2	False
>=	Greater than or equal to	5 >= 2	True

*Related topics*

- ◆ [Using arithmetic operators on page 106](#)
- ◆ [Using Boolean logic on page 107](#)
- ◆ [Hierarchy of operators on page 105](#)

**Programming with variables**

Variables are place holders for information such as setpoints, time delays, and operating modes. Control Basic uses two types of variables, value objects and local variables.

**BACnet value objects as variables**

Analog and binary value objects are used as program variables in BACnet controllers.

**Local variables**

Local variables can only be used within the Control Basic program that refers to them. The values they represent cannot be directly transferred to other Control Basic programs. Local variables are useful for counters or to store the results of local calculations.

**Standard Control Basic local variables** Within each Control Basic program there are 26 local variables. These variables are assigned the letters of the alphabet (A-Z).

**Next Generation Control Basic** Local variables in controllers that use Next Generation Control Basic can be either a single-letter variable (A-Z) or a declared local variable. Use the command LOCALS to declare local variables within each Control Basic program. All single letter local variables are automatically declared unless LOCALS declares any variable. If LOCALS declares any local variable then any single letter variable must also be declared. For details on using and declaring local variables, see the keyword [LOCALS on page 130](#).

## Transferring values between BACnet controllers

BACnet controllers from KMC Controls can read from and write to the present values in objects in other controllers on the internetwork.

### Reading properties from other BACnet devices

With Control Basic, a KMC BACnet controller can request the present value from any other controller on the BACnet internetwork. However, the following limitations must be observed.

- ◆ Each controller can request a present value from no more than 32 other devices.
- ◆ Each controller can request up to a total of 64 present values from the 32 devices.

For example, controller 50 can request two values each from controllers 1-32, four values each from controllers 1-16 or eight values from panels 1,5,6,8 and thirty-two values from panel 9.

To read a value from another panel, you must know the panel's device instance. The instance is separated from the value by a period (.).



In the following example, the WAIT statement is required. Do not delete it or the program will not run correctly.

---

**Syntax:** *device instance.object*

```
10 REM * POINTS TRANSFERRED FROM DEVICE 1213
20 REM * OUTSIDE AIR TEMPERATURE *
30 AV24 = 1213.AI7
40 WAIT 0:05:00
50 END
```

### Writing values to other controllers

To change the value in another BACnet device, you must know the devices instance number. The instance is separated from the value by a period(.).

**Syntax:** *device instance.object*

```
201.AO1 = AV1
```

## Programming with mnemonics

Mnemonics are a short, easy to remember abbreviations to use when writing Control Basic programs to refer to various parts of a controller. For example, a physical input is entered as *AI1* or *BI1* in BACnet controllers instead of typing *Input1*.

Control Basic mnemonics for BACnet objects are listed in the table [Control Basic mnemonics for BACnet objects on page 110](#). The following line of Control Basic is an example of using mnemonics to refer to an analog input object and a binary output object.

```
If AI08 > 10 Then Start BO2
```

- ◆ Mnemonics listed as *Read Only* can read a value— such as its value or status— from that object.
- ◆ Mnemonics listed as *Read and Write* describes a property—such as its value or status—that may be changed through programming or by direct access.



Caution

If a Control Basic program uses a mnemonic to refer to an invalid local object or property within an object, the program will compile but it will halt execution. The reason for the halt is listed in the Description of Halt column of the Program Objects dialog.

**Table 6–7 Control Basic mnemonics for BACnet objects**

Object type	Mnemonic	Property	Action
Accumulator	ACC#	Present Value	Read Only
Accumulator	ACC#-PR	Pulse Rate	Read Only
Analog Input	AI#	Present value	Read and Write
Analog Input	AI#-LL	Low Limit	Read and Write
Analog Input	AI#-HL	High Limit	Read and Write
Analog Output	AO#	Present value	Read and Write
Analog Output	AO#-LL	Low Limit	Read and Write
Analog Output	AO#-HL	High Limit	Read and Write
Analog Value	AV#	Present value	Read and Write
Analog Value	AV#-LL	Low Limit	Read and Write
Analog Value	AV#-HL	High Limit	Read and Write
Binary Input	BI#	Present value	Read and Write

**Control Basic mnemonics for BACnet objects (continued)**

<b>Object type</b>	<b>Mnemonic</b>	<b>Property</b>	<b>Action</b>
Binary Output	BO#	Present value	Read and Write
Binary Value	BV#	Present value	Read and Write
Loop	LOOP#	Present value	Read Only
Loop	LOOP#-B	Bias value	Read and Write
Loop	LOOP#-D	Derivative constant	Read and Write
Loop	LOOP#-P	Proportional constant	Read and Write
Loop	LOOP#-I	Integral constant	Read and Write
Loop	LOOP#-SP	Setpoint	Read and Write
Multi-State Input	MSI#	Present value	Read and Write
Multi-State Output	MSO#	Present value	Read and Write
Multi-State Value	MSV#	Present value	Read and Write
Schedule	SCHED#	Present value	Read Only
Trend	TL#-EN	Log enable	Read and Write





## Section 7: **Keywords for Control Basic**

This section covers the keywords for the Control Basic programming language.

---

The Control Basic keywords for operators, commands and functions are reserved for Control Basic. They may not be used for descriptors, labels or names of objects, variables, or procedures.

### **Using example programs from help**

You can use example programs from the help system. Highlight the example and then copy the example and paste it into a Control Basic program.

### **Syntax for commands and functions**

Required spaces are shown with underscore marks ( `_` ) and indicate that a space must be included for proper syntax. Optional items are shown in brackets [ ].

### **ABS**

This function returns the absolute value of the expression. The expression can be a single number or the result of a calculation.

**Syntax:** `ABS(_expression_)`

*Standard Control Basic example*

Returns 2.3, the absolute value of -2.3.

```
10 A = ABS ( -2.3 )
```

Returns the absolute value from the result of the calculation.

```
10 C = ABS ( AV1 - AI1 )
```

*Next Generation Control Basic example*

Returns 2.3, the absolute value of -2.3.

```
A=ABS (-2.3)
```

Returns the absolute value from the result of the calculation.

```
C=ABS (AV1 - AI1)
```

**ALIAS**

Declares a local variable and dynamically binds the value of a property to the variable. It also sets two intervals at which Control Basic will read from or write to the property bound to the variable.

**Syntax:** *ALIAS(device, object, property, local, read interval, write interval)*

**Note:** Next generation Control Basic only.

The same point may be bound to an ALIAS in more than one program. However, the lowest read or write interval of all ALIAS statements within the device is used in all programs.

See the related topic [FLUSH on page 122](#).

**Table 7–1 ALIAS parameters**

Parameter	Description	Comments
device	The device instance number	Enclose the name of a device with quotation marks (""). Names are case sensitive.
object	A valid mnemonic	See <a href="#">Programming with mnemonics on page 110</a> .
property	The property and priority for writing.	Priority is ignored for read only objects such as inputs.
local	The local name to use within the program.	Use as a local variable within the Control Basic program in which the alias is declared.
read interval	The interval at which Control Basic will read the property.	To never read from the object, use <i>NONE</i> . The default value is 60 seconds.
write interval	The interval at which Control Basic will write to the property.	To never write to the object, use <i>NONE</i> . The default value is <i>NONE</i> .

In the following example Control Basic binds the present value of binary output BO1 in device 1212 to the local name *Lights*. Control Basic reads the value of output BO1 once an hour and writes the value every 60 seconds.

```
ALIAS(1212, BO1, PV@4, Lights, 1:00:00, 60)
```

The last example binds the value of the input object AI2 to the local variable *OutsideAirTemp*. When the device argument is omitted, Control Basic binds the local variable to the device with the lowest device instance that contains an input object AI2.

```
ALIAS("", AI2, PV, OutsideAirTemp, 100, NONE)
```

## AND

AND is a Boolean operator that performs the logical *AND* of two expressions. The result is *true* if both expressions are non-zero; otherwise, the result is *false*.

**Syntax:** *result* = *expression1* AND *expression2*

In the following example, local variable C will always equal 1 as long as both local variables A and B = 1

```
10 A = 1 : B = 1 : C = A AND B
```

See the related topic [Using Boolean logic](#) on page 107.

## ARCCOS

Returns the arccosine of the specified angle. *Angle* is expressed in radians.

**Syntax:** ARCCOS( *angle* )

```
10 A = ARCCOS ( AV1 )
```

See the related topic [COS](#) on page 118.

## ARCSIN

Returns the arcsine of the specified angle. *Angle* is expressed in radians.

**Syntax:** ARCSIN( *angle* )

```
10 A = ARCSIN ( AI8 )
```

See the related topic [SIN](#) on page 145.

## ARCTAN

Returns the arctangent of the specified angle. *Angle* is expressed in radians.

**Syntax:** ARCTAN( *angle* )

```
10 A = ARCTAN ( AV12 )
```

See the related topic [TAN](#) on page 146.

**AVG**

This statement returns the average value of the items enclosed in parenthesis. In the following example, local variable *D* equals the average of analog inputs 1, 3 and 6.

**Syntax:** *AVG( \_expression\_ , \_expression\_ ...)*

*Standard BACnet Control Basic example*

10 D = AVG( AI1 , AI3 , AI6 )

*Next Generation Control Basic example*

D=AVG(AI1, AI3, AI6)

**BIND**

Binds a BACnet device instance to a physical network address. This is typically used to bind an MS/TP slave device to a master device.

**Syntax:** *BIND (device, network, mac, option)*

**Note:**

For Next generation Control Basic only.

**Table 7–2 BIND parameters**

Parameter	Description	Comments
device	The instance number of the device.	
network	The number of the BACnet network on which the device resides	May be expressed as decimal or hexadecimal notation. Use zero (0) as the local network.
mac	The MAC address of the device.	

**Table 7–3 BIND options**

Option	Description
Hint	Sets the default address but uses whatever can be found by the controller. This is the default state.
Locked	Forces the default address back to this every time it is changes.

Examples:

BIND (550013, 1, 13)

BIND(123456, 678, 0x24 )

BIND(123456, 0x44, 09:88:77:55:44:55 )

BIND(123456, 0x4, 10.1.2.3:678 )

```
Bind(123456, 0x4, 10.1.2.3:678 , LOCKED)
BIND(123456, 0, 10.1.2.3:678 , HINT )
```

**CLEAR**

Resets the value of all local variables—variables labeled A-Z and declared variables—to zero.

```
10 CLEAR
```

**CLOSE**

Sets the value of a named point, binary output or value object to *off*.

**Syntax:** *CLOSE\_point*

*Standard BACnet Control Basic example*

```
10 CLOSE BO2
20 CLOSE A
```

*Next Generation Control Basic example*

```
CLOSE BO2
CLOSE A
```

*Related topics*

- ◆ [OPEN on page 136](#)
- ◆ [START on page 145](#)
- ◆ [STOP on page 146](#)

**CONST**

Use to declare a variable and assign to it a fixed value. Do not use with variables that change with subsequent steps in the program.

**Syntax:** *CONST, variable, variable, ...*

**Note:**

For Next generation Control Basic only.

- ◆ Constants must be declared before they are used in a program. A constant may be declared anywhere in the program but typically it is at the beginning of the program.
- ◆ Must start with a letter A-Z, a-z, or an underscore (\_). Constants are not case sensitive.
- ◆ Can be any combination of letters (A-Z or a-z), numbers (0-9) or the underscore (\_).
- ◆ A constant may be used only within the program in which it is declared.
- ◆ A constant cannot duplicate a keyword, local variable, label or alias.

```
CONST Freeze = 32
```

```
CONST Boiling = 212
```

To declare local variables, see the keyword [LOCALS on page 130](#).

**COS-1**

Returns the arccosine of the specified angle. *Angle* is expressed in radians.

**Syntax:** `COS-1(_angle_)`

**Note:**

Deprecated for BACnet controllers. See the keyword [ARCCOS on page 115](#).

**COS**

Returns the cosine value of a specified angle. *Angle* is expressed in radians.

**Syntax:** `COS(_angle_)`

*BACnet examples*

*Standard BACnet Control Basic example*

```
10 A = COS ( AV1 )
```

*Next Generation Control Basic example*

```
A = COS (AV1)
```

**DEC**

Decrements the value of *point* by the value of *step*. If *step* is omitted, the *step* value is 1.

**Syntax:** `DEC(_point_,_step_) DEC(_point_)`

See the related topic [INC on page 127](#).

*Standard BACnet Control Basic example*

```
10 DEC ( AV1 , A + B )
```

```
20 DEC ( AV2 )
```

*Next Generation Control Basic example*

```
DEC ( AV1 , A + B )
```

```
DEC ( AV2 )
```

**DEWPOINT**

Returns the dew point in degrees Fahrenheit based on Outside Air Humidity (OAH) and Outside Air Temperature (OAT). OAT is in degrees Fahrenheit.

**Syntax:** `DEWPOINT(_OAH_,_OAT_)`

See the related topic [DEWPOINTS on page 119](#) to express temperature in degrees Celsius.

*Standard Control Basic example*

```
10 D = DEWPOINT( AI1 , AI2 )
```

*Next Generation Control Basic example*

```
D=DEWPOINT(AI1, AI2)
```

**DEW-POINT**

Returns the dew point in degrees Fahrenheit based on Outside Air Humidity (OAH) and Outside Air Temperature (OAT). OAT is in degrees Fahrenheit.

**Syntax:** *DEW-POINT(\_OAH\_,\_OAT\_)*

**Note:**

Deprecated for BACnet controllers. See the keyword [DEWPOINT](#) on page 118.

**DEWPOINTS**

Returns the dew point in degrees Celsius based on Outside Air Humidity (OAH) and Temperature (OAT). OAT is in degrees Celsius.

**Syntax:** *DEWPOINTS( OAH , OAT )*

See the related topic [DEWPOINT](#) on page 118 to express temperature in degrees Fahrenheit.

```
10 D=DEWPOINT(AI1, AI2)
```

**DISABLE**

DISABLE sets the value of a point, which can be the present value of an input, output or value object, to *off*.

**Syntax:** *DISABLE\_point*

*Standard BACnet Control Basic example*

```
10 DISABLE AO1
```

```
20 DISABLE A
```

*Next Generation Control Basic example*

```
DISABLE AO1
```

```
DISABLE A
```

*Related topics*

- ◆ [ENABLE](#) on page 121
- ◆ [START](#) on page 145
- ◆ [STOP](#) on page 146

**DOM**

Returns the current day of the month.

*Standard BACnet Control Basic example*

```
10 IF+ DOM = 15 THEN 20 ELSE END
20 REM Continue program execution
```

*Next Generation Control Basic example*

```
IF+ DOM=15 THEN GOTO Continue ELSE END
Continue:
REM Continue program execution
```

**DOW**

Returns a numerical value for the day of the week.

*BACnet example*

In BACnet controllers the days of the week are numbered 1-7.

- ◆ Monday is day 1.
- ◆ Sunday is day 7.
- ◆ The day can also be identified by the first three letters (SUN, MON, etc.).

```
10 IF DOW = MON THEN START BO1
```

**DOY**

Returns the day of the year.

- ◆ The year always begins on January 1.
- ◆ December 31st is day 366.
- ◆ February is always counted as having 29 days which means March 1 is always day 61.
- ◆ On non-leap years, February 29 (day 60) is skipped.

The day of the year may be expressed as either a number or the first three letters of the month and the day of the month.

*Standard BACnet Control Basic example*

```
10 IF DOY = 92 THEN START BO1
```

*Next Generation Control Basic example*

```
IF DOY=92 THEN START BO1
```



**ENABLE**

ENABLE sets the value of an object, which can be the present value of an input, output or value object to 1 or *on*.

**Syntax:** *ENABLE\_point*

*BACnet example*

```
10 ENABLE AO1
20 ENABLE A
```

*Related topics*

- ◆ [DISABLE on page 119](#)
- ◆ [START on page 145](#)
- ◆ [STOP on page 146](#)

**END**

Terminates the execution of a program. When the END statement is encountered, the program stops reading lines and exits the program. All programs lines that follow an encountered END statement are *not* executed.

In the following example, the last line is ignored and the analog output will always equal 10.

*BACnet example*

```
10 AO1 = 10
20 END
30 AO1 = 7
```

*KMD example*

```
10 OUT1 = 10
20 END
30 OUT1 = 7
```

**ENTHALPY**

Calculates enthalpy based on Outside Air Temperature (OAT) and Outside Air Humidity (OAH). The value returned is expressed as BTUs per pound of air. OAT is in degrees Fahrenheit.

**Syntax:** *ENTHALPY(\_OAH\_,\_OAT\_)*

See the topic [ENTHALPYSI on page 122](#) to enter OAT in degrees Celsius.

*BACnet example*

```
10 E = ENTHALPY( AI1 , AI2 )
```

**ENTHALPYSI**

Calculates enthalpy based on Outside Air Temperature (OAT) and Outside Air Humidity (OAH). The value returned is expressed as joules per kilogram of air. OAT is in degrees Celsius.

**Syntax:** `ENTHALPYSI(_OAH_,_OAT_)`

See the topic [ENTHALPY](#) on page 121 to enter OAT in degrees Fahrenheit.

*Standard Control Basic example*

```
10 E = ENTHALPY-SI ( AI1 , AI2 )
```

*Next Generation Control Basic example*

```
E=ENTHALPYSI (AI1, AI2)
```

**ENTHALPY-SI**

Calculates enthalpy based on Outside Air Temperature (OAT) and Outside Air Humidity (OAH). The value returned is expressed as joules per kilogram of air. OAT is in degrees Celsius.

**Syntax:** `ENTHALPY-SI(_OAH_,_OAT_)`

**Note:** Depreciated for BACnet controllers. See [ENTHALPYSI](#) on page 122.

**FLUSH**

When a FLUSH statement runs, Control Basic immediately reads from or writes to the property bound to the local variable declared by ALIAS.

**Syntax:** `Flush (LocalAlias1)`

**Note:** Next generation Control Basic only.

```
ALIAS (1212, BO1, PV@4, Lights, 1:00:00, 60)
FLUSH (Lights)
```

See the related topic [ALIAS](#) on page 114.

**FOR TO NEXT**

The FOR TO NEXT loop repeats a set of instructions a specific number of times.

**Syntax:** `FOR_ControlVariable=_StartValue_to_EndValue(_Step_Increment_)`

- ◆ *ControlVariable* is the variable that *FOR* increments each time the loop repeats. It controls whether or not Control Basic repeats the loop. *ControlVariable* must be local to the controller in which the Control Basic program is running.
- ◆ *StartValue* is the initial value that Control Basic assigns to *ControlVariable*.
- ◆ *EndValue* is the value that the *ControlVariable* must equal before the loop ends.
- ◆ *Increment* is the amount that Control Basic adds to *ControlVariable* with each iteration of the loop. *Increment* can be a positive or negative value. If *STEP* and *Increment* are omitted, the default value is 1.
- ◆ *NEXT* ends *FOR TO* statements. It directs Control Basic to increment *ControlVariable* and to test whether it is greater than *EndValue*. If it is not, the loop continues at the first statement within the loop; if not, the program continues at the first statement following *NEXT*.

In the following examples, the value of *A* increases from 0 to the value of *AV2* in 0.1 increments, pausing 10 seconds between steps.

*Standard Control Basic example*

```
10 FOR A = 0 TO AV2 STEP .1
20 AO1 = A
30 WAIT 0:00:10
40 NEXT A
50 END
```

*Next Generation Control Basic example*

```
FOR A = 0 TO AV2 STEP .1
    AO1 = A
    WAIT 0:00:10
NEXT A
END
```

## GOSUB

GOSUB is the preferred way of branching to a subroutine in a program and then returning to the original point and continuing execution. When Control Basic encounters a GOSUB statement, the program jumps to the location specified and continues reading program lines until a RETURN statement is encountered. At that point the program returns to the line following the GOSUB statement.

**Syntax:** *GOSUB\_line#*

In the following examples, the program reads the first line, jumps to the third line and then to the fourth line. The RETURN statement on the fourth line sends the program back to the second line and the program ends.

See the related topics [GOTO on page 124](#) and [RETURN on page 139](#).

*BACnet example*

```
10 GOSUB 30
20 END
30 REM
40 RETURN
```

*Next generation Control Basic*

```
GOSUB DoSubRoutine
END

DoSubRoutine:
RETURN
```

## GOTO

This function redirects the program to a new location in the program. In the following examples, the program does not run the second line and output 1 is never changed.

See the related topic [GOSUB on page 123](#).

**Syntax:** *GOTO\_line#*

*BACnet example*

```
10 GOTO 30
20 START BO1
30 REM Program continues here
40 END
```

*Next generation Control Basic*

```
GOTO JumpToEnd
START BO1

JumpToEnd:
END
```

## HALT

Stops the program from running and sets the *Program State* property to *Halted*. The string *Message* is displayed in the property *Description of Halt*.

**Syntax:** *HALT "Message"*

**Note:** For Next generation Control Basic only.

Once stopped the program cannot be restarted from Control Basic. It can be restarted only by doing one of the following:

- ◆ Performing a warm start or cold start
- ◆ Cycling controller power
- ◆ Changing the *Program Change* property on the program object to *Run*.

```
HALT "Shutting down the program"
```

## HSEL

Selects the highest (second highest, etc.) value of the expressions listed. The value for *N* defines whether it selects the highest (1) or the second highest (2) etc. The expressions can be variables, inputs, outputs, calculations, etc.

**Syntax:** *HSEL( N , expression , expression...)*

This example returns the local variable *A* equal to the second highest value of the items listed.

*Standard BACnet Control Basic example*

```
10 A = HSEL( 2 , AI1 , AI2 , AI3 , AV1 )
```

*Next Generation Control Basic example*

```
A=HSEL(2, AI1, AI2, AI3, AV1)
```

## IF THEN

IF THEN is a decision making statement. The *expression* parameter can be any expression capable of being true or false (high or low, on or off, etc.) If *expression* is *true* the THEN statement will be executed. If the expression is *false* (not true) the ELSE statement will be executed. The ELSE statement and associated clause are optional. If they are not included the program reads and executes the next program line.

**Syntax:** *IF\_expression\_THEN\_clause(ELSE\_clause)*

*Standard BACnet Control Basic example*

In this example, the program stops analog output #5 if analog input #1 is less than analog input #2. If analog input #1 is not less than analog input #2, analog output #5 will be turned on (started). If the *ELSE START AO5* statement was not included, the program will stop analog output AO5 if analog input AI1 is less than input AI2. Otherwise, it will do nothing and end the program.

```
IF AI1 < AI2 THEN STOP AO5 ELSE START AO5
END
```

**Note:** Use commas to separate multiple commands in an IF statement.

```
IF T > S THEN START BO1 , STOP BO2
IF T > S THEN START BO1, STOP BO2 ELSE STOP BO1
```

#### *Next generation Control Basic*

By using ENDIF, Next Generation Basic supports block and nested IF THEN statements.

```
IF TIME > 7:00 THEN
  a=b
ENDIF
```

```
IF TIME > 7:00 THEN
  IF TIME < 9:00 THEN
    B=C
  ENDIF
ENDIF
```

## **IF+ THEN**

IF+ is similar to IF THEN, except that it detects the first time a condition changes from *false* to *true*. If the expression is true and on the previous scan it was not true, the THEN clause will be executed.

**Syntax:** *IF+\_expression\_THEN\_clause(\_ELSE\_clause)*

The ELSE statement and associated clause are optional. If they are not included the program reads and executes the next program line.

When a button closes the circuit in the sensor analog input 1 to which it is connected, the program will branch down to line 30, which increases the setpoint (AV13 or VAR13) by one degree. This will happen only once for each time the button is pressed and released. Even if the button is held for several minutes it will only increment the setpoint by one degree.

See the related topic [IF THEN on page 125](#) and [IF- THEN on page 127](#).

#### *BACnet example*

```
10 IF+ SENSOR-ON( AI1 ) THEN GOSUB 30
20 END
30 AV13 = AV13 + 1 : REM Line 30 starts here
40 RETURN
```

*Next generation Control Basic*

```

IF+ SENSORON( AI1 ) THEN GOSUB 30
END
30: : AV13 = AV13 + 1
RETURN

```

**IF- THEN**

IF- is similar to IF THEN except that it detects the first time a condition changes from *true* to *false*. In this case the THEN clause would only be executed if the expression is *false* and on the previous scan it was *true*.

**Syntax:** *IF-\_expression\_THEN\_clause(\_ELSE\_clause)*

**Note:** The *ELSE* and associated clause is optional.

See the related topic [IF THEN on page 125](#) and [IF+ THEN on page 126](#).

**INC**

Increments the value of the argument *point* by the value of the argument *step*. If *step* is omitted, the step value is 1. *Point* may be the present value of any analog object.

**Syntax:** *INC(\_point\_ step\_) INC(\_point\_)*

See the related topic [DEC on page 118](#).

*Standard BACnet Control Basic example*

```

10 INC( AV1 , A + B )
20 INC( AV2 )

```

*Next Generation Control Basic example*

```

INC( AV1 , A + B )
Inc( AV2 )

```

**INT**

INT returns the integer portion of the numeric value *expression*. The value returned is the greatest integer that is less than or equal to the value of *expression*.

**Syntax:** *INT(\_expression\_)*

The following examples calculate the hour of the day (0-23) without minutes or seconds. The result is stored in analog value object AV1.

*Standard BACnet Control Basic example*

```

10 AV1 = INT( TIME / 100 )

```

*Next Generation Control Basic example*

```
AV1=INT (TIME/100)
```

## INTERVAL

The INTERVAL command performs an operation at a regular time interval. The statement is *true* at each expression time; otherwise it is *false*. The time format is in *hh:mm:ss* format.

**Syntax:** *INTERVAL*(*\_expression\_*)

The program sequence in this example increases the setpoint temperature—stored in value object AV1—by 0.1° every 45 seconds.

*Standard BACnet Control Basic example*

```
10 IF INTERVAL( 00:00:45 ) THEN AV1 = AV1 + .1
20 END
```

*Next Generation Control Basic example*

```
IF INTERVAL(00:00:45) THEN AV1 = AV1 + .1
END
```

## INVLN

The function INVLN returns the inverse natural logarithm of the numeric expression.

**Note:** Next generation Control Basic only.

**Syntax:** *INVLN*(*\_expression\_*)

*Standard BACnet Control Basic example*

```
10 B = INVLN( AI4 * 125 )
```

*Next Generation Control Basic example*

```
10 B = INVLN( AI4 * 125 )
```

See the related topics [LN on page 130](#).

## ISNAN

ISNAN tests the value of *expression* to determine if it is a valid number. If the value of *expression* is equal to NAN (Not A Number), then ISNAN returns *true*.

**Syntax:** *ISNAN*( *\_expression\_* )

**Note:** Next generation Control Basic only.



A typical use of *ISNAN* is to test the present value property of an object in a remote device.

**Note:** If the remote device goes offline, the last good value is held until the controller is reset with a cold start, warm start, or power cycle. After the reset, the value in the remote property becomes NAN until it is read by another controller.

In the following example the program tests the present value of analog input 4 in device instance 4410 once every minute. If the value is a usable number then the remote value is stored in value object AV503. If the remote value is not valid, the value object is set equal to 55, the default value.

```

IF INTERVAL ( 00:01:00 ) THEN
  REM Verify that the value is good
  IF ISNAN( 4410.AI4 ) THEN
    REM Set a default value
    AV503 = 55
  ELSE
    REM Use the received value
    AV503 = 4410.AI4
  ENDIF
ENDIF

```

## LN-1

LN-1 returns the inverse natural logarithm of the numeric expression.

**Syntax:** *LN-1*(*\_expression\_*)

**Note:** Depreciated for BACnet controllers. See the keyword [INVLN](#) on page 128.

## LET

The LET function assigns *expression1* to equal *expression2*. Use this function assign initial values to inputs, outputs, variables, PID control loops or schedule.

**Syntax:** *LET\_expression1=\_expression2*

```

10 LET OUT1 = CON1
20 LET A = OUT1

```

The LET function is optional. Both of the following examples will produce the same results.

```

30 VAR3 = IN2 - 23

```

```
40 LET VAR3 = IN2 - 23
```

**LN**

The function *LN*( ) returns the natural logarithm of the numeric expression.

**Syntax:** *LN*(*\_expression\_*)

*Standard BACnet Control Basic example*

```
10 B = INVLN( AI4 * 125 )
```

*Next Generation Control Basic example*

```
B = INVLN( AI4 * 125 )
```

**LOCALS**

Use to declare local variables. A local variable may be used only within the program in which it is declared.

**Note:** Next generation Control Basic only.

**Syntax:** *LOCALS* *variable*[, *variable*, ...]

```
Locals ChilledWaterSetpoint, a, b
```

- ◆ Local variables must be declared before they are used in a program.
- ◆ LOCALS may declare variables anywhere in the program but typically variables are declared at the beginning of the program.
- ◆ Must start with a letter A-Z, a-z, or an underscore (\_). They are not case sensitive.
- ◆ Can be any combination of letters (A-Z or a-z), numbers (0-9) or the underscore (\_).
- ◆ Variables A-Z are automatically declared unless LOCALS declares another variable.
- ◆ A local variable cannot duplicate a keyword, constant, label or alias.

See the related topic [CONST](#) on page 117.

**LSEL**

LSEL returns the lowest, second lowest, etc. value of the expression listed. The value *N* defines whether it selects the lowest (1) or second lowest (2) etc. Expressions can be variables, inputs, outputs, calculations, etc.

**Syntax:** *LSEL*(*\_N\_*,*\_expression\_*,*\_expression\_*...)

In the examples local variable *A* will be set equal the second lowest value of the items listed.

*Standard BACnet Control Basic example*

```
10 A = LSEL( 2 , BI1 , BI2 , BI3 , BV1 )
```

*Next Generation Control Basic example*

```
A=LSEL(2, BI1, BI2, BI3, BV1)
```

## MAX

MAX returns the maximum value of the expression listed. Expressions can be the present value of an input, output, or value object or the result of a calculation.

**Syntax:** *MAX(\_expression\_,\_expression\_...)*

*Standard BACnet Control Basic example*

```
10 A = MAX( AI1 , AI2 , AI3 , AV1 )
```

*Next Generation Control Basic example*

```
A=MAX(AI1, AI2, AI3, AV1)
```

## MIN

MIN returns the minimum value of those expression listed. Expressions can be the present value of an input, output, or value object or the result of a calculation.

**Syntax:** *MIN(\_expression\_,\_expression\_...)*

*Standard BACnet Control Basic example*

```
10 B = MIN( BI1 , BI2 , BI3 , BV1 )
```

*Next Generation Control Basic example*

```
B=MIN(BI1, BI2, BI3, BV1)
```

## MOD

MOD is an arithmetic operator that returns *true* if the division operation returns a remainder equal to *remainder* in the expression; returns *false* if the remainder of the division is not equal to *remainder* in the expression.

**Syntax:** *Dividend MOD Divisor=\_remainder*

*Standard BACnet Control Basic example*

```
10 IF AV1 MOD 5 = 0 THEN START BO2 ELSE STOP BO2
```

*Next Generation Control Basic example*

```
IF AV1 MOD 5=0 THEN START BO2 ELSE STOP BO2
```

The following example uses MOD to calculate leap year. If the year in the controllers's internal clock is a leap year, local variable *L* is set to *true*. For other years the variable *L* is set to *false*.

```
10 IF YEAR MOD 4 = 0 AND YEAR MOD 100 <> 0 OR YEAR MOD 400 =
0 THEN L = 1 ELSE L = 0
```

See the related topic [Using arithmetic operators on page 106](#).

## MODELNUMBER

Returns the numerical portion of the model number of the controller.

**Syntax:** *MODELNUMBER*

```
10 AV1=MODELNUMBER
```

## MODEL-NUMBER

Returns the numerical portion of the model number of the controller.

**Note:**

Deprecated for BACnet controllers. See the keyword [MODELNUMBER on page 132](#).

## MONTH

Returns the current month of the year.

*Standard BACnet Control Basic example*

```
10 M = MONTH
```

*Next Generation Control Basic example*

```
M=MONTH
```

## NAN

Use NAN to set a variable or property to a *Not A Number* constant or to test if the variable or property is equal to *Not A Number*.

**Note:**

Next generation Control Basic only.

```
IF A <> NAN THEN GOTO CONTINUE
B = 55
CONTINUE:
B = A
```

See the related topic [ISNAN on page 128](#).

**NETSENSORSTATUS** Returns the connection status of a NetSensor with which the program can take appropriate action. The function returns *true* if a functional NetSensor is connected to the controller and *false* if the controller does not detect a NetSensor.

*Standard Control Basic example*

```
10 IF NOT NETSENSORSTATUS THEN STOP BV1
```

*Next Generation Control Basic example*

```
IF NOT NETSENSORSTATUS THEN STOP BV1
```

## NETSENSOR-STATUS

Returns the connection status of a NetSensor so the program can take appropriate action. The function returns *true* if a functional NetSensor is connected to the controller and *false* if the controller does not detect a NetSensor.

**Note:**

Deprecated for BACnet controllers. See the keyword [NETSENSORSTATUS on page 133](#).

## NOT

NOT is a Boolean operator that performs a logical negation operation on an expression. If the expression is 0, the result is 1. If the expression is non-zero, the result is 0.

**Syntax:** *result = NOT expression*

```
IF NOT BV1 THEN STOP BO2
```

See the related topic [Using Boolean logic on page 107](#).

## ON GOSUB

ON GOSUB is a control statement. The program branches to the location from the list passed by the statement. The value of *expression* determines the location in the list to which Control Basic will continue. *Expression* is rounded to an integer. For example, if *expression* = 3 the program will branch to the location in the list. If the value of *expression* is greater than the number of locations listed or if *expression* is less than 1, no branch will occur.

**Syntax:** *ON\_expression\_GOSUB\_location1[\_location2\_location3\_...]*

See the related topic [RETURN on page 139](#).

*Standard BACnet Control Basic example*

In this example Value Object AV1 is equal to 3 which will cause the program

to branch to Line 80. If AV1 were equal to 2, the program would branch to Line 60, etc.

```

10 AV1 = 3
20 ON AV1 GOSUB 40 , 60 , 80
30 END
40 RETURN
50 RETURN
60 RETURN

```

#### *Next generation Control Basic*

In this example Value Object *AV1* is equal to 3 which will cause the program to branch to label 80. If *AV1* equals 2, the program will branch to label 60, etc.

```

AV1 = 3
ON AV1 GOSUB 40 , 60 , 80
END
40:
RETURN
60:
RETURN
80:
RETURN

```

## ON GOTO

ON GOTO is a control statement. The program branches to the locations from the list passed by the statement. The value of *expression* determines the location in the list to which the program will branch. *Expression* is rounded to an integer. For example, if *expression* = 3 the program will branch to the third location in the list. If the value of *expression* is greater than the number of locations listed or if *expression* is less than 1, no branch will occur.

**Syntax:** *ON\_expression\_GOTO\_location1[\_location2\_location3\_...]*

#### *Standard BACnet Control Basic example*

In this example Value Object *AV1* is equal to 3 which will cause the program to branch to Line 60. If AV1 were equal to 2, the program would branch to Line 60, etc.

```

10 AV1 = 3
20 ON AV1 GOTO 40 , 50 , 60
30 END
40 REM Program continues here

```

```

50 REM Program continues here
60 REM Program continues here

```

*Next generation Control Basic*

In this example Value Object *AV1* is equal to 3 which will cause the program to branch to label 60. If *AV1* equals 2, the program will branch to label 50, etc.

```

AV1 = 3
ON AV1 GOTO Forty, Fifty, Sixty
END
Forty:
Fifty:
Sixty:

```

## ONERROR

When an error is detected on the line previous to the line containing **ONERROR**, the program continues at the line specified by *location*. In the following examples, the program attempts to read an off-panel object and if the object is not found it substitutes the value 70.

**Syntax:** *ONERROR location*

*Standard BACnet Control Basic example*

```

10 AV16 = 101-AV1
20 ONERROR 40
30 GOTO 60 : REM Jump around error recovery
40 REM Error recovery
50 AV16 = 70
60 REM Continue program

```

*Next generation Control Basic*

```

AV16 = 101.AV1
ONERROR 40
GOTO 60 : REM Jump around error recovery
40:
REM Error recovery
AV16 = 70
60:
REM Continue program

```

**ON-ERROR**

*ON-ERROR* is a control statement. The program branches to the line specified by *location* when the previous Control Basic line detects an error.

**Syntax:** *ON-ERROR location*

**Note:** Deprecated for BACnet controllers. See the keyword [ONERROR](#) on page 135.

**OPEN**

Use OPEN to set the present value of an object to *on* or *true*.

**Syntax:** *OPEN\_point*

*Standard BACnet Control Basic example*

```
10 OPEN V
20 OPEN A
30 OPEN BO1
```

*Next Generation Control Basic example*

```
OPEN V
OPEN A
OPEN BO1
```

*Related topics*

- ◆ [CLOSE](#) on page 117
- ◆ [START](#) on page 145
- ◆ [STOP](#) on page 146

**OR**

OR is a Boolean operator that performs the logical *OR* of the two expressions. The result is *true* if either expression is *true*. The result is *false* if both expressions are *false*.

**Syntax:** *result = expression1 OR expression2*

In the following example, local variable C will equal 1 if either of the variables A and B are equal to 1.

```
10 A = 1 : B = 0 : C = A OR B
```

See the related topic [Using Boolean logic](#) on page 107.



**OUTPUTOVERRIDE** Returns the switch position of an optional HPO-6700 series output board installed in the controller in which Control Basic is running.

**Syntax:** *OUTPUTOVERRIDE( output )*

The argument *output* is returned *false* if the switch is in *AUTO* and *true* if the switch is set to either the *OFF* or *HAND* position. *Output* can be expressed as either of the following:

- ◆ The instance number of the output.
- ◆ A local variable whose value represents the number of an output object.

*Standard BACnet Control Basic example*

```
10 BV20 = OUTPUTOVERRIDE ( 2 )
```

*Next Generation Control Basic example*

```
BV20=OUTPUTOVERRIDE (2)
```

**OUTPUT-OVERRIDE** Returns the switch position of an optional HPO-6700 series output board installed in the controller in which Control Basic is running.

**Syntax:** *OUTPUT-OVERRIDE(\_expression\_)*

**Note:** Deprecated for BACnet controllers. See the keyword [OUTPUTOVERRIDE on page 137](#).

**PANELADDRESS** Returns the device instance number of the controller on which the Control Basic program is running.

*Standard Control Basic example*

```
10 P = PANELADDRESS
```

*Next Generation Control Basic example*

```
P=PANELADDRESS
```

**PANEL-ADDRESS** Returns the KMD network address of the controller on which Control Basic is running.

**Note:** Deprecated for BACnet controllers. For BACnet controllers see [PANELADDRESS on page 137](#)

**PI** Inserts the value of pi. The following examples convert angle *D* from degrees to radians.

*Standard Control Basic example*

```
10 A = PI * ( D / 180 )
```

*Next Generation Control Basic example*

```
A=PI*(D/180)
```

## POWERLOSS

Use POWERLOSS to detect loss of power to the controller. It will also detect any other condition that causes the controller to run its restart sequence. This function returns *true* on the first scan of all Control Basic programs after power is restored. After the first scan, it returns as *false*.

The following examples are useful for monitoring intermittent power failures at a controller. The value object AV32 increments by 1 each time power is restored.

*Standard Control Basic example*

```
10 IF POWERLOSS THEN AV32 = AV32 + 1
20 END
```

*Next Generation Control Basic example*

```
IF POWERLOSS THEN AV32 = AV32+1
END
```

## POWER-LOSS

Use POWER-LOSS to detect loss of power to the controller or any condition that forced the controller to reset. This function returns *true* on the first scan of all Control Basic programs after power is restored. After the first scan, it returns as *false*.

**Note:** Deprecated for BACnet controllers. See [POWERLOSS on page 138](#).

The following example is useful for monitoring intermittent power failures at a controller. The variable VAR32 increments by 1 each time power is restored. POWER-LOSS may also be used to detect any other condition that causes the controller to perform its restart sequence.

## REM

Place a REM statement at the beginning of a program line to insert explanatory comments or remarks. REM is a method to document the use of a subroutine or to explain a formula used in a calculation.

**Syntax:** *REM\_string*

*Standard BACnet Control Basic example*

```

70 REM ** Step temperature every minute by 1 degree **
80 IF INTERVAL( 0:01:00 ) THEN AV1 = AV1 + 1
90 REM **calculation for velocity (FPM)**
100 AV1 = 4004.4 * SQR( AI1 )
110 END

```

*Next Generation Control Basic example*

```

REM ** Step temperature every minute by 1 degree **
IF INTERVAL(0:01:00) THEN AV1=AV1+1
REM **calculation for velocity (FPM)**
AV1=4004.4*SQR(AI1)
END

```

**RETURN**

This command returns control from a subroutine to a calling procedure. RETURN is always used in conjunction with GOSUB or ONGOSUB statements to RETURN from a subroutine.

See the related topics [GOSUB on page 123](#) and [ON GOSUB on page 133](#).

**RLQ**

Relinquishes the priority level of a BACnet output or value object.

**Syntax:** *RLQ\_object@priority*

*Standard Control Basic example*

```
10 RLQ AO1@7
```

*Next Generation Control Basic example*

```
RLQ AO1@7
```

**RND**

RND is a numeric function which returns a random number between 0 and *expression*-1. It is useful for applications such as security lighting.

**Syntax:** *RND(\_expression\_)*

*Standard BACnet Control Basic example*

```
10 IF TIME = 20:00:00 + RND( 10:00:00 ) THEN START AO1
```

*Next Generation Control Basic example*

```
IF TIME=20:00:00+RND(10:00:00) THEN START AO1
```

## SCANS

SCANS returns the rate a controller is processing all Control Basic programs. The value returned is expressed in scans per second. As the complexity or length of a program increases it takes longer to process and the number of scans per second decreases.

A useful application for SCANS is to create a time-based counter similar to those used for time-proportioning relays. If you use the INTERVAL or WAIT statements you are limited to a time division no smaller than one second. By programming a counter based on SCANS, the smallest time increment can range between 1/5 of a second to 1/50 of a second depending on how busy the controller is.

If a time proportioning relay sequence is based on a 5 second cycle for example, having time increments in only 1 second divisions would likely not be sufficient.

See the related topic [About Control Basic scans on page 100](#).

### *Standard BACnet Control Basic example*

```
10 A = 1 / SCANS
20 B = A + B : REM B Will increment by 1 every second (based
on scan rate)
30 IF B > 10 THEN B = 0 : REM B counts 0-10 in 10 seconds
40 END
```

### *Next Generation Control Basic example*

```
A=1/SCANS
B=A+B:REM B Will increment by 1 every second (based on scan
rate)
IF B>10 THEN B=0: REM B counts 0-10 in 10 seconds
END
```

## SCHEDOFF

Use this function to determine the time of day that a schedule object will set its reference object to *Inactive* or a value of zero (0).

**Syntax:** *SCHEDOFF(\_schedule object #, \_time\_)*

- ◆ The returned value is the difference—in seconds—from the time specified to the time that the schedule's present value will be *Inactive* (or zero).
- ◆ The schedule object must be within the same device.
- ◆ Both *schedule object #* and *time* may be expressed as either a local variable (a), value (12:00, 6) or value object (AV1).
- ◆ The parameter *time* is entered in 24-hour format.
- ◆ The value returned is based on the current day of the week.
- ◆ A returned value of 0 indicates that the schedule is already set to off.
- ◆ A return of 86,400 indicates that there are no more scheduled Off times for the current day.

**Note:** KMC Controls recommends that, because it is computationally intensive, Control Basic does not continuously run the SCHEDOFF function.

See [SCHEDON on page 141](#) for calculating the time when a schedule becomes active.

## SCHED-OFF

Deprecated for BACnet controllers. See [SCHEDOFF on page 140](#).

## SCHEDON

Use this function to determine the time of day that a schedule object will set its reference object to *Active* or a non-zero value.

**Syntax:** `SCHEDON(_schedule object #_,_time_)`

- ◆ The returned value is the difference—in seconds—from the time specified to the time that the schedule's present value will become *Active* (or non-zero).
- ◆ The schedule object must be within the same device.
- ◆ Both *schedule object #* and *time* may be expressed as either a local variable (A), value (12:00, 6) or value object (AV1).
- ◆ The parameter *time* is entered in 24-hour format.
- ◆ The value returned is based on the current day of the week.
- ◆ A returned value of 0 indicates that the schedule is already on.
- ◆ A return of 86,400 indicates that there are no more scheduled On times for the current day.

**Note:** KMC Controls recommends that, because it is computationally intensive, Control Basic does not continuously run the SCHEDON function.

In the following example, line 10 calculates a value once every five minutes. The time parameter of 0:00:00 indicates when time the schedule will be on. If the schedule is set to change to On at 5:00 AM, when line 10 executes, value object AV1 will return a value of 18,000 seconds.

See [SCHEDOFF on page 140](#) for calculating the time when a schedule becomes inactive.

*Standard Control Basic example*

```
10 IF INTERVAL( 0:05:00 ) THEN AV1 = SCHEDON( 1 , 0:00:00 )
```

*Next Generation Control Basic example*

```
IF INTERVAL(0:05:0) THEN AV1=SCHEDON(1,0:00:00)
```

## SCHED-ON

Deprecated for BACnet controllers. See [SCHEDON on page 141](#).

## SENSOROFF

Use SENSOROFF to detect an open-circuit condition on an input that is configured as an analog input. A typical application is to detect momentary conditions such as a pressed button. If the opened contact condition lasts longer than two minutes the function will be disabled. After three minutes, the object will change *Out Of Service* to *true* but the commands will still execute.

**Syntax:** `SENSOROFF(_IN#_)`

When used with [SENSORON on page 143](#) and [IF THEN on page 125](#), [IF+ THEN on page 126](#), or [IF- THEN on page 127](#) you can determine three separate conditions from one input:

- ◆ A temperature or other analog reading.
- ◆ A sensor with open contacts (SENSOROFF).
- ◆ A sensor with closed contacts (SENSORON).

SENSOROFF can also be used with inputs using a table if the minimum value in the table is set to a value greater than zero and its maximum value is less than 5.00 volts.

**Table 7–4 Example table for SENSOROFF in BACnet controllers**

Input Voltage	Detected condition
0	Closed circuit
0.4	Temperature-55 degrees
4.9	Temperature-95 degrees
5.0 or greater	Open circuit

In the table [Example table for SENSOROFF in BACnet controllers on page 142](#), the input voltage under normal temperature conditions will never fall below 0.4 volts. When a sensor is shorted to ground, the input voltage will fall to zero, which is a condition SENSORON can detect. Similarly, if the circuit is opened, the controller will read the open circuit voltage, which is higher than the maximum 4.9 volts in the table which will be detected by SENSOROFF.

*Standard Control Basic example*

```
10 IF- SENSOROFF( AI1 ) THEN AV11 = 02:00:00
```

*Next Generation Control Basic example*

```
IF- SENSOROFF(AI) THEN AV11=02:00:0
```

## SENSOR-OFF

Use SENSOR-OFF to detect an open-circuit condition on an input that is configured as an analog input.

**Syntax:** `SENSOR-OFF(_IN#_)`

**Note:**

Deprecated in BACnet controllers. See the keyword [SENSOROFF on page 142](#).

## SENSORON

Use SENSORON to detect 0 volts (closed-circuit) condition on an input that is configured as an analog input. A typical application is to detect momentary conditions such as a pressed button. If the opened contact condition lasts longer than two minutes the function will be disabled. After three minutes, the object will change *Out Of Service* to *true* but the commands will still execute.

**Syntax:** `SENSORON(_IN#_)`

When used with [SENSOROFF on page 142](#) and [IF THEN on page 125](#), [IF+ THEN on page 126](#), or [IF- THEN on page 127](#) you can determine three separate conditions from one input:

- ◆ A temperature or other analog reading.
- ◆ A sensor with open contacts (SENSOROFF).
- ◆ A sensor with closed contacts (SENSORON).

SENSORON can also be used with analog inputs using a table if the minimum value in the table is set to a value greater than zero and its maximum value is less than 5.00 volts.

**Table 7–5 Input conditions for SENSORON for BACnet controllers**

Input Voltage	Detected condition
0	Closed circuit
0.4	Temperature-55 degrees
4.9	Temperature-95 degrees
5.0 or greater	Open circuit

In the table *Input conditions for SENSORON for BACnet controllers* on page 144, the input voltage under normal temperature conditions would never fall below 0.4 volts. When a sensor is shorted to ground, the input voltage will fall to zero, which is a condition SENSORON can detect. Similarly, if the circuit is opened, the controller will read 5.00 volts, which is higher than the maximum 4.9 volts in the table which will be detected by SENSOROFF.

*Standard Control Basic example*

```
10 IF+ SENSORON( AI1 ) THEN AV11 = 02:00:00
```

*Next Generation Control Basic example*

```
IF+ SENSORON(AI1) THEN AV1=02:00:00
```

## SENSOR-ON

Use SENSOR-ON to detect 0 volts (closed-circuit) condition on an input that is configured as an analog input.

**Syntax:** `SENSOR-ON(_IN#_)`

**Note:** Depreciated for BACnet controllers. See [SENSORON on page 143](#).

## SIN-1

Returns the arcsine of the specified angle. The value *angle* is expressed in radians.

**Syntax:** `SIN-1(_angle_)`

**Note:** Depreciated for BACnet controllers. See the keyword [ARCSIN on page 115](#).



**SIN**

Returns the sine of the specified angle. *Angle* is expressed in radians.

**Syntax:** *SIN*(*\_angle\_*)

*Standard Control Basic example*

```
10 A = SIN( AI1 )
```

*Next Generation Control Basic example*

```
A=SIN(AI1)
```

**SQR**

The SQR function returns a value equal to the square-root of the value *expression*.

**Syntax:** *SQR*(*\_expression\_*)

*Standard BACnet Control Basic example*

```
10 A = SQR( AI1 )
```

*Next Generation Control Basic example*

```
A=SQR(AI1)
```

**START**

START sets the value of a point to *on*.

**Syntax:** *START**\_point*

*Standard BACnet Control Basic example*

```
10 START AO1
```

```
20 START F
```

```
30 START A
```

*Next Generation Control Basic example*

```
START AO1
```

```
START F
```

```
START A
```

*Related topics*

- ◆ [STOP on page 146](#)
- ◆ [DISABLE on page 119](#)
- ◆ [ENABLE on page 121](#)

**STOP**

STOP sets the value of a *point* to *Off*.

**Syntax:** *STOP\_point*

*Standard BACnet Control Basic example*

```
10 STOP AO1
20 STOP F
```

*Next Generation Control Basic example*

```
STOP AO1
STOP F
```

*Related topics*

- ◆ [START on page 145](#)
- ◆ [DISABLE on page 119](#)
- ◆ [ENABLE on page 121](#)

**TAN-1**

A function that Rreturns the arctangent of the specified angle.The value of *angle* is expressed in radians.

**Syntax:** *TAN-1(\_angle\_)*

**Note:** Deprecated for BACnet controllers. See the keyword [ARCTAN on page 115](#).

**TAN**

A function that returns the tangent of the specified angle. The value *angle* is expressed in radians.

**Syntax:** *TAN(\_angle\_)*

*Standard Control Basic example*

```
10 A = TAN( AV10 )
```

*Next Generation Control Basic example*

```
A=TAN(AV10)
```

**TBL**

Use TBL to look up the value of an expression such as a variable in a custom created table. Use look-up tables when the value of the expression is nonlinear or requires a complicated calculation to arrive at the proper value. Use ON-ERROR after TBL to recover from problems within the table.

**Syntax:** *TBL(\_expression\_,\_table#\_)*

When referencing a table within Control Basic, use the form TBL (x , N) where "N" is the table number and "x" is the value within the table. The function returns the interpolated y coordinate-ordinate of the table. N must be a whole number, x can be an integer.

*Standard Control Basic example*

```
10 AV1 = TBL( AI3 , 2 )
20 ON-ERROR 40
30 GOTO 60 : REM Jump around error recovery
40 REM Start error recovery
50 REM End error recovery
60 REM Continue program
```

*Next generation Control Basic*

```
AV1 = TBL( AI3 , 2 )
ONERROR 40
GOTO 70 : REM Jump around error recovery
40:
REM Start error recovery
REM End error recovery
70:
REM Continue program
```

## TIME

A function that returns a value based on the time as maintained in the controller running Control Basic.

*BACnet example*

KMC BACnet controllers return a value for system time as the number of seconds after midnight.

```
10 T=TIME
```

The following program returns hours, minutes and seconds in local variables H, M, and S.

```
10 H = INT( TIME / 60 / 60 ) : REM Hours
20 M = INT( ( TIME - H * 60 * 60 ) / 60 ) : REM Minutes
30 S = TIME - 60 * M - 60 * 60 * H : REM Seconds
```

## TIMEOFF

Use this statements to determine if the present value of object has been in the *off* state for a specific period of time. If *point* is a value object, it must be

configured as a unit of time.

**Syntax:** *TIMEOFF(\_point\_)*

*Standard Control Basic example*

```
10 IF TIMEON(AO1)>0:10 THEN START AO2
20 REM AO2 will turn on if AO1 has been on for longer than
10 minutes
```

*Next Generation Control Basic example*

```
IF TIMEON(AO1)>0:10 THEN START AO2
REM AO2 will turn on if AO1 has been on for longer than 10
minutes
```

See the related keyword topic [TIMEON on page 148](#).

## TIMEON

Use this statements to determine if the present value of object has been in the *on* state for a specific period of time.

**Syntax:** *TIMEOFF(\_point\_)*



**Caution**

TIMEON responds to the time a property is set to *On* as maintained by the controller running the program. This time may not be the same as the actual time if the object containing the property is in a different controller.

In the following example, analog output AO2 will be set to *On* if output AO1 has been set to *On* for more than 10 minutes.

```
10 IF TIMEON( AO1 ) > 0:10 THEN START AO2
```

See the related keyword topic [TIMEOFF on page 147](#).

## WAIT

Use WAIT to control timed events. The program waits for the time period specified before reading the next program line. Other programs in the controller will not be affected as WAIT applies only to the program in which it is listed.

**Syntax:** *WAIT\_period*

**Tip:** The value for *period* can be expressed in 24-hour format (14:15) or converted to decimal format (1425). See the related topic [TIME on page 147](#).

**Note:** Plan carefully when using WAIT before a conditional branch such as with IF-THEN. Conditions within a controller may change the value of points or properties during the waiting period.

*Standard BACnet Control Basic example*

```
10 START AO2
20 WAIT 0:10
30 REM * * Waits 10 Minutes at line 20 * *
40 WAIT 00:00:10
50 REM * * Waits 10 seconds at line 40 * *
60 END
```

*Next Generation Control Basic example*

```
START AO2
WAIT 0:10 : Rem Line 20
REM Line 30 * * Waits 10 Minutes at line 20 * *
WAIT 00:00:10 : REM Line 40
REM * * Waits 10 seconds at line 40 * *
END
```

## XOR

XOR performs a logical exclusion on two Boolean expressions. The result is *true* if the two expressions are different; otherwise, the result is *false*.

**Syntax:** *result = \_expression1\_XOR\_expression2*

In the following example, local variable *C* will equal 1 as long as variables *A* and *B* are not equal to each other.

```
10 A = 1 : B = 0 : C = A XOR B
```

See the related topic [Using Boolean logic on page 107](#).

## YEAR

Returns the four-place value of the current year.

```
10 Y = YEAR
```



## Section 8: The System Menu

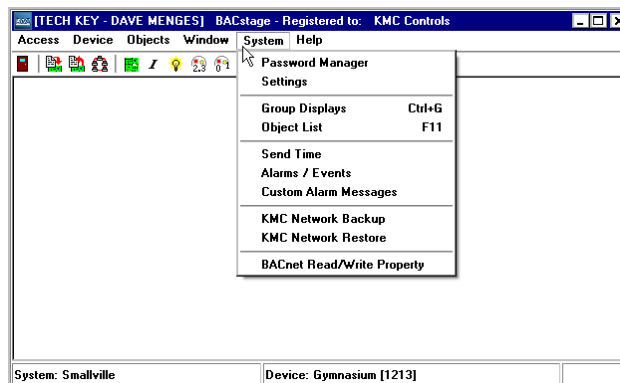
Use the settings in the system menu for all controllers in the internetwork.

---

Use the System Menu to select the following functions:

- ◆ *Password Manager* on page 152
- ◆ *Settings* on page 155
- ◆ *Group Displays* on page 157
- ◆ *Objects List* on page 166
- ◆ *Send Time* on page 169
- ◆ *Alarms/Events* on page 170
- ◆ *Custom Alarm Messages* on page 172
- ◆ *Network Backup* on page 167
- ◆ *Network Restore* on page 168
- ◆ *BACnet Read/Write Property* on page 173

**Illustration 8–1** System menu



## Password Manager

Password Manager is an administrator's tool that sets permission to use specific objects and functions in BACstage. Use the Password Manager to:

- ◆ Add or delete the names of operators that have permission to view or change a system.
- ◆ Delete the names of operators that no longer have permission to view or change a system.
- ◆ Assign a password and grant permissions to individual operators.
- ◆ Assign a group display that BACstage will open when an operator signs on.

**Note:** Only the master administrator or a user with *Operator Level of Administrator* privileges can make changes in Password Manager.

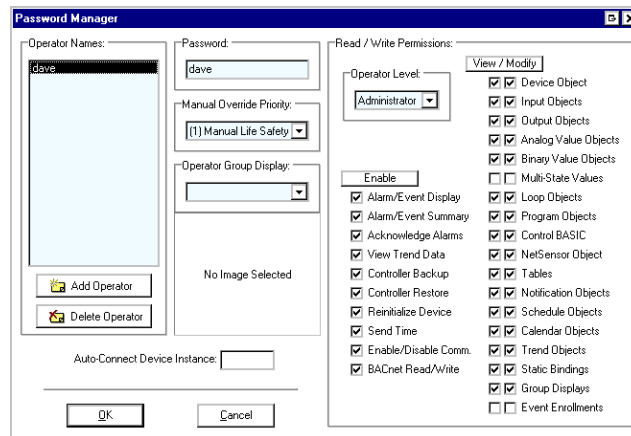
To use Password Manager, do the following:

1. Sign-on to a system from the system list.
2. Choose *System* from the menu bar and then *Password Manager*.
3. Select or add an operator name. Assign, or if required, modify the password.
4. Assign permissions, a Group Display, and a Manual Override Priority level.
5. Click *Ok* to save the changes. Click *Cancel* to close Password Manager without saving changes.

### *Related topics*

- ◆ [System List on page 14](#)
- ◆ [Automatic Sign Off on page 156](#)
- ◆ [AutoConnect on page 19](#)



**Illustration 8-2 Password Manager dialog box**

### Password Manager functions

**Operator Names** A list of all operators with permission to view or modify the system to which BACstage is presently connected.

- ◆ Use the *Add User* or *Delete User* buttons to change the names in the list.
- ◆ Click the name and then make changes to the operator's password or permissions.
- ◆ Clicking *OK* closes Password Manager and saves the changes.
- ◆ Clicking *Cancel* closes Password Manager without making changes.

**Add Operator** Add an operator name to the *User Names* list.

**Delete Operator** Remove an operator from the *User Names* list.

**Password** Assign a password to the operator. The password is not encrypted in Password Manager but is encrypted when stored in the Settings.INI file for the system.

**Manual Override Priority** Assigns the highest level of write priority that the operator may use when changing a present value. See the section [Priority arrays on page 92](#) for additional information on using priority arrays.

**Operator Group Display** Selects a group display that will open when a user signs on. The group display is choose from list of group displays already configured. See [Group Displays on page 157](#) for details about configuring a group display.

**Auto-Connect Device Instance** Enter the device instance number of the BACnet device to which BACstage will automatically connect when the operator signs in.

**View/Modify button** Grant to permission to use specific BACstage functions.

- ◆ Clicking the *View/Modify* button first selects all of the *View* check boxes for the objects below the button.
- ◆ The second click selects all of the *View* and *Modify* check boxes.
- ◆ Third click clears all of the check boxes.

**Enable button** Clicking Enable selects all of the check boxes for the functions below the *Enable* button. Clicking it a second time clears all of the check boxes.

**User Level** Choose from three predefined user levels or choose *Custom* to create a custom level for the user. User levels are listed in the table [Preset permissions on page 154](#).

**Table 8–1** Preset permissions

Level 0	Level 1	Level 2	Admin	Master Admin	Capability
	R	W	W	W	Device Objects
	R	W	W	W	Input Objects
	R	W	W	W	Output Objects
	R	W	W	W	Analog Value Objects
	R	W	W	W	Binary Value Objects
	R	W	W	W	Loop Objects
	R	W	W	W	Program Objects
	R	W	W	W	NetSensor Objects
	R	W	W	W	Program Objects Code (CB Programs)
	R	W	W	W	Device Type Tables
	R	W	W	W	Notification Class Objects
	R	W	W	W	Schedule Objects
	R	W	W	W	Calendar Objects
	R	W	W	W	Trend Objects
	R	W	W	W	Accumulator Object
8	8	8	8	1	Manual Override Priority
		R	W	W	Alarm Summary
		R	W	W	Alarm/Event Display

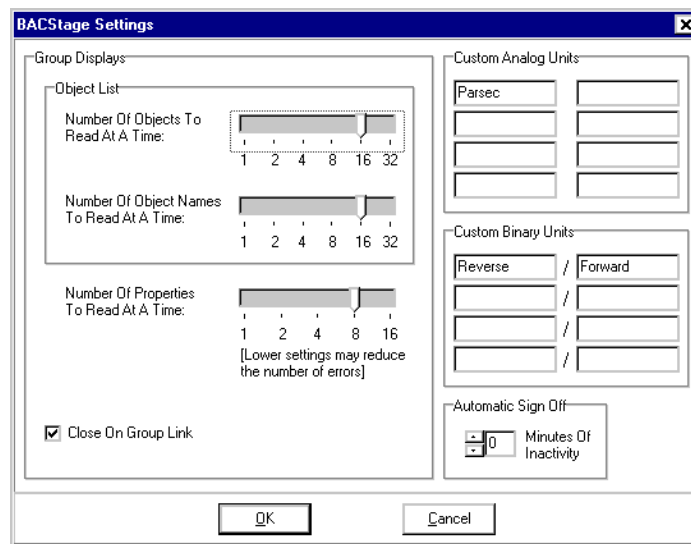
**Preset permissions (continued)**

Level 0	Level 1	Level 2	Admin	Master Admin	Capability
		W	W	W	Alarm Acknowledge
	R	R	R	R	Trend Log Display
		W	W	W	Reinitialize Device
			W	W	Panel Backup
			W	W	Panel Restore
Group #	R	W	W	W	Group Displays
		W	W	W	Send Time
			W	W	Password Assignment (Must be Master password)
				W	Initialize systems with live communication connections
				W	Adjust automatic user sign off time

**Settings**

The Settings dialog controls system behavior for several BACstage functions. To open the Settings dialog box, choose *Settings* from the *System* menu.

**Illustration 8-3 Settings dialog box**



**Note:** Only the master administrator may open the *Settings* dialog box.

**Number Of Objects to Read At A Time** Limits the number of objects that BACstage reads from a single controller at one time. For KMC BACnet controllers, this can be set to a higher number. If you receive error messages, lower this setting.

**Number Of Object Names to Read At A Time** Limits the number of object names that BACstage reads from a single controller at one time. For KMC BACnet controllers, this can be set to a higher number. If you receive error messages, lower this setting.

**Monitor Settings—Number Of Properties to Read At A Time** Limits the number of properties that BACstage reads from a single controller at one time. For KMC BACnet controllers, this can be set to a higher number. If you receive error messages, lower this setting.

**Close On Group Link** When checked, a group display automatically closes when an operator chooses a link to another group display. The topic [Group Displays on page 157](#) explains viewing, creating and editing Group Displays.

### Custom Analog Units

Enter up to eight custom analog units. BACstage places custom units at the bottom of the *Units* list for selection when on configuring analog input, output, value and loop objects.

**Tip:** The labels for the custom units are not stored in the controller but stored only in the file *options.txt* which is within the current job folder (see [BACstage job file location on page 179](#)). To duplicate the custom units in another system—including systems on the same computer— place a copy of the file *options.txt* in the job folder of the new system. An alternate method is to use Microsoft Notepad to edit *options.txt*.

### Custom Binary Units

Enter up to four pairs of custom binary units. The first unit on of each pair is the inactive unit. Custom binary units are stored in the file *options.txt* within the current job folder and are sent also to the controller during an update notification.

### Automatic Sign Off

BACstage automatically signs out an operator after a period of inactivity which is specified in *Minutes Of Inactivity*. To maintain time synchronization, BACstage does not disconnect from the system. Setting *Minutes Of Inactivity* to 0 disables the automatic sign-off function.

## Group Displays

Group displays are custom designed windows that provide quick access to the most often used parts of a system. Use *Group Displays* to:

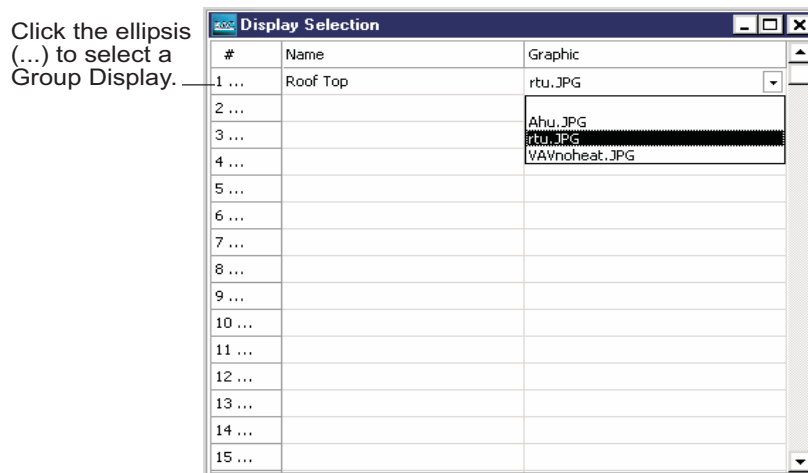
- ◆ Create easy-to-use navigation windows based on floor plan drawings or building photos.
- ◆ Display—on one window—the values and equipment status from multiple locations.
- ◆ Control equipment operation.

A group display can be as simple as a few text links or a complex graphical user interface that includes animated displays and site plans. With the library of graphics supplied with BACstage, you can display all parts of a system such as temperature, setpoints and equipment settings. In addition to displaying the values of object properties, links can be placed in group display that opens other group displays.

### Opening a Group Display

To open a *Group Display* for viewing or editing, choose *Group Display* from the *System* menu and then click the ellipsis (...) next to the group number.

#### Illustration 8-4 Group Display Selection List



#### Related topics

- ◆ [Creating a group display on page 158](#)
- ◆ [Adding and modifying object properties on page 160](#)
- ◆ [Adding and modifying links to group displays on page 163](#)
- ◆ [Changing a present value from a group display on page 165](#)

#### *Features of the Group Display selection list*

**#(Group number)** Click the group number to chose a group display for viewing or editing.

**Name** Enter a group name to create a new group. A group name is required.

**Graphic** Choose from the background graphics previously placed in the *Images* folder within the current job folder. See [BACstage job file location on page 179](#) for the location of the images folder.

## Creating a group display

Before creating a group display, see the topic [About graphic formats on page 165](#) for details about background and animated graphic files.

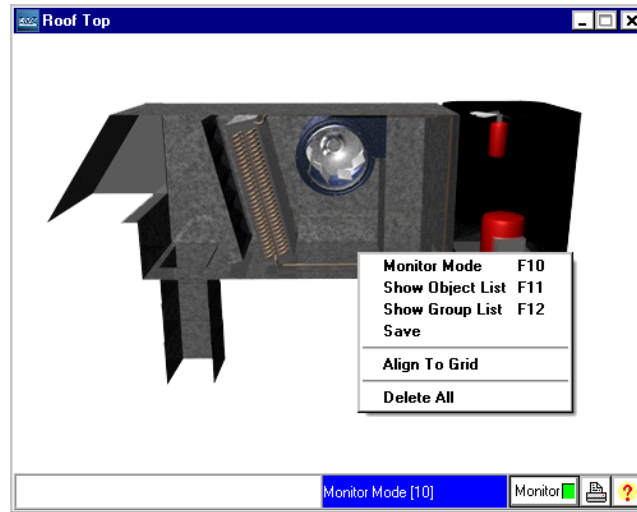
1. Place the background graphics in the *Images* folder inside of the current job folder. See [BACstage job file location on page 179](#) for the location of the *Images* folder. The supported background graphic formats are:
  - Non-animated GIF
  - JPG
  - WMF
  - BMP
2. Place the animated gif files in the images folder with the background graphics.
3. Name the group in the *Groups Display Selection* list.
4. Select a background graphic from the drop down list in the *Graphic* column.
5. Click the ellipsis (...) next to the *Group Display* number. The group display opens.

## Modifying a group display

To add items to a group display or to edit existing items, the group display must be in edit mode. Use one of the following methods to change to edit mode:

- ◆ Right-click over an open area in the group display and choose the *Edit Mode* pop-up.
- ◆ Press F10.

Illustration 8-5 Group display with edit popup



The status bar at the bottom of the group displays changes from *Monitor Mode* to *Edit Mode*. When in edit mode:

1. Right-click over an open area.
2. Choose from the modify pop-up.
3. Select one of the following functions.

**Monitor Mode F10** End the edit session and return to monitor mode.

**Show Object List F11** Select the *Show Object List* to add an object property to the group display. See the topic, [Adding and modifying object properties on page 160](#) for additional details.

**Show Group List F12** Choose the *Show Group List* to add a link from the group display to another group display. See the topic [Adding and modifying links to group displays on page 163](#) for details about links to other group displays.

**Save** Stores the group display in the *Groups* folder of the current job folder. See [BACstage job file location on page 179](#) for the location of the *Groups* folder.

**Align to Grid** Aligns all of the items on the background to an invisible 25 x 25 pixel grid. The upper left corner of the item aligns to the nearest grid intersection.

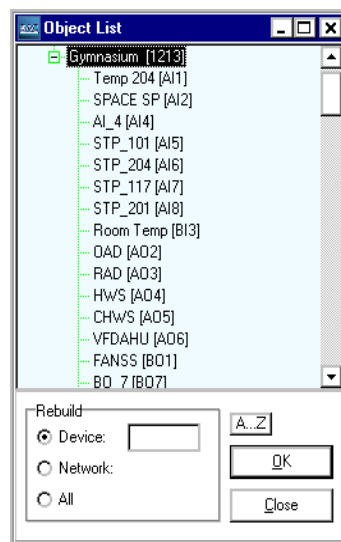
**Delete All** Removes all items from the group display.

## Adding and modifying object properties

To add an object to an existing group display:

1. Open the group display from the *Group Display Selection* list.
2. Change to edit mode.
3. Right-click and choose *Show Object* list from the popup. The object list opens.
4. To show each of the properties in the object, click the + sign next to the object to expand the list. The supported object properties are:
  - Input objects including accumulator
  - Output objects
  - Value objects
5. Drag a property from the object list and drop it onto the background.
6. Edit the appearance of the item.

### Illustration 8–6 Object list

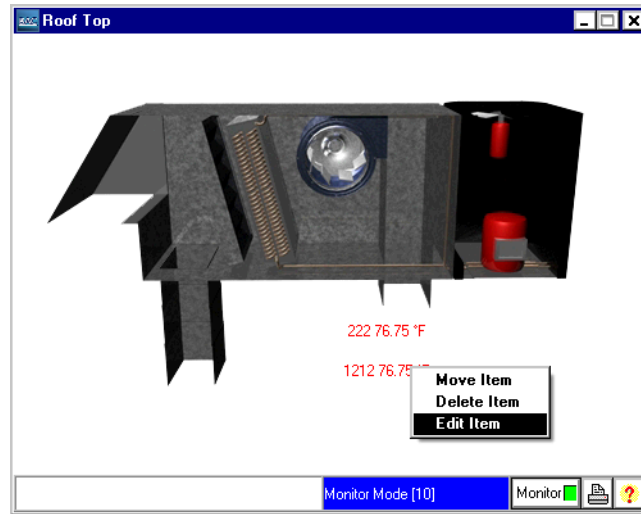


### Adding animation and modifying the text

To modify the text of an object or to attach animation to the object.

1. Open a group display.
2. Change to edit mode.
3. Right-click over the text of an item.
4. Choose a function from the edit popup.



**Illustration 8-7 Group display with edit popup**

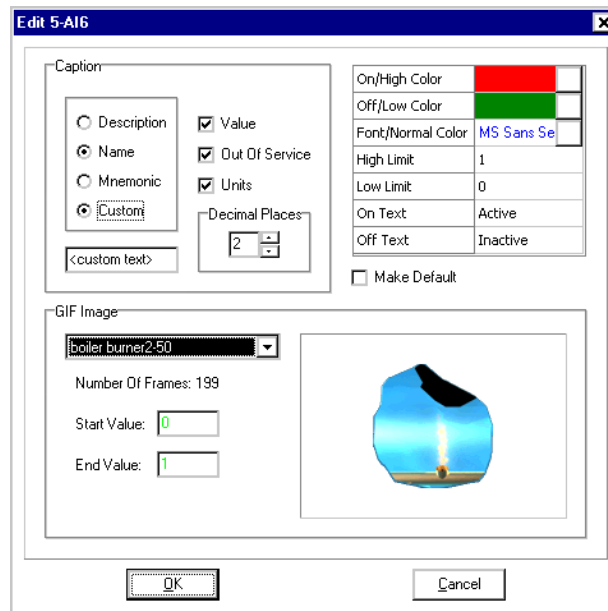
**Move Item** Use to change the location of the item. Place the point over the item and then right-click and drag. The pointer drags an outline of the item which can then be dropped anywhere on the background.

**Delete Item** Removes the item from the display.

**Edit Item** Opens the *Property Display* dialog box. Use this dialog box to change the appearance of the text and attach animation to the object.

## Features of the Property Display dialog

Illustration 8–8 Property display dialog



**Description** When selected, displays the description of the object.

**Name** When selected, displays the name of the object.

**Mnemonic** When selected, displays the device instance followed by the Control Basic mnemonic of the object. See the topic [Programming with mnemonics on page 110](#) for a list of available mnemonics.

**Custom** When selected, you may add custom text to Description, Name or Mnemonic.

**Value** When selected, displays the present value of the object.

**Out of Service** When selected, displays the status of the out-of-service property.

**Units** When checked, displays the units of measure associated with the present value of the object.

**Decimal Places** Sets the precision of the displayed values.

**On/High and Off/Low Color** Sets the colors associated with *On* and *Off* text.

**Font/Normal Color** Opens a dialog box with which you can choose the font, size and color of the text.

**High and Low Limit** The values at which the text will change color. High and low limits are not associated with the BACnet high and low limit

properties. The limits are used only to define a condition that sets text color based upon the present value of the object.

**On and Off Text** *On Text* is the text displayed when the property is *On* or equal to 1. *Off Text* is displayed when the property is *Off* or equal to 0.

### **Adding an animated Gif**

Use the *GIF Image* features to attach a GIF animation file to an object in a group display.

**Drop-down list** Select an animated GIF file from the listed files. BACstage displays a thumbnail image of the file in the area to the right of the drop-down list.

**Number of frames** Displays the number of frames in the selected GIF file.

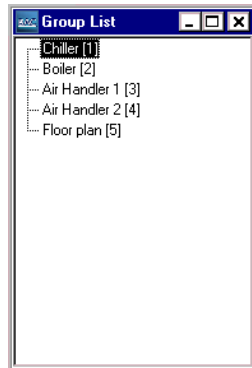
**Start and End Value** Use to scale the present value of an analog object to the GIF animation. *Start Value* is associated with the first frame in the GIF animation; *End Value* is associated with the last frame of the GIF animation.

**Example:** If the GIF is a 100 frame bar graph and you want to use it to display a present value that represents temperature from 32 to 100 degrees, set *Start Value* to 32 and *End Value* to 100. When the temperature is 32, the bar graph will be at minimum. When the temperature is at 100, the bar graph is at maximum.

## **Adding and modifying links to group displays**

To add a link from a group display to another group display:

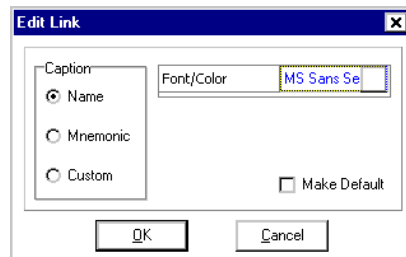
1. Open the group display from the *Group Display Selection* list.
2. Change to *Edit* mode.
3. Right-click and choose *Show Group List* list from the popup. The object list opens.
4. Drag a group display name from the group list and drop it onto the background.
5. Edit the appearance of the link as required.

**Illustration 8–9 Group list****Features of the group list edit pop-up**

**Move Item** Use to change the location of the item. The pointer drags an outline of the text field which can then be dropped anywhere on the background.

**Delete Item** Remove the text field from the display.

**Edit item** Opens the text edit dialog box. Use this dialog to change the appearance of the text link.

**Illustration 8–10 Group Edit Link dialog****Features of Edit Link Dialog**

**Name** Displays the name of the group display as entered in the group display selection list.

**Mnemonic** Displays the mnemonic GRP followed by the group display number from the group display selection list.

**Custom** Displays the text entered in the text box next to Custom.

**Font/Color** Opens a dialog box with which you can choose the font, size and color of the text.

**Make Default** Saves the settings in the Group Edit Link dialog box as the default values.

### About graphic formats

Group displays are created with two types of graphic files:

- ◆ Background graphics which display the overall view of the system.
- ◆ Animated graphics which display motion and provide control.

**Background graphics** A background graphic is the base graphic for a group display and must one of the formats in the table, [Group display graphic file formats](#).

**Table 8–2 Group display graphic file formats**

File type	Description
Non-animated GIF	A loss less compression technique that supports only 256 colors. Use the GIF format for images with only a few distinct colors, such as line drawings, black and white images and small text that is only a few pixels high.
JPG	A lossy compression format that supports over 16 million colors. Use JPG where picture quality detail must be preserved.
BMP	An uncompressed file format developed for Windows.
WMF	An uncompressed file format developed to exchange graphics information between Microsoft Windows applications. WMF files can hold both vector and bit-mapped images. Group displays can use only files with bit-mapped images
EMF	A 32-bit, enhanced version of WMF.

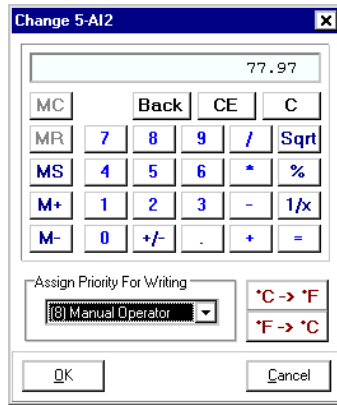
**Animated graphics** Animated graphic files must be the GIF format. To use the animated graphics, copy the files you need for animation into the *Images* folder in the job folder.

### Changing a present value from a group display

To manually change the present value of an object in a group display:

1. If in edit mode, press F10 or right-click and choose *Edit Mode*.
2. Click the item. The manual change dialog opens.
3. Click the on-screen buttons with the mouse or use the computer keyboard numeric keypad to enter a new present value. Enter also the *Priority for Writing* value.
4. Click *OK* when finished.

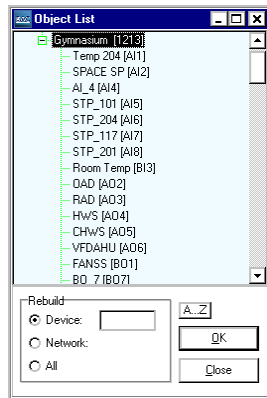
Illustration 8–11 Manual change dialog



## Objects List

Loads into memory the object names for one or more devices on the internetwork. The Object List is an internal list for using local names while writing Control Basic programs. See *Programming with names* on page 103 for details on programming with object names.

Illustration 8–12 Object list



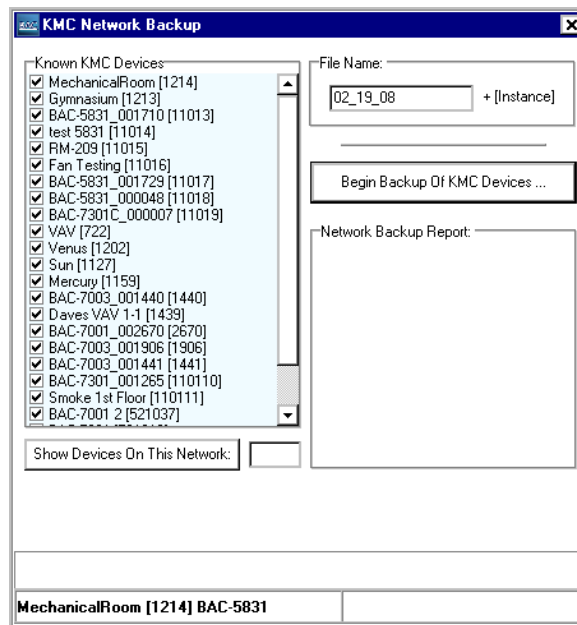
## Network Backup

Use *Network Backup* to save to disk a *BAC* configuration file for each of the selected controllers in the *Known KMC Devices* list. BACstage stores object, device properties and programming information in a *BAC* file inside of a backup folder which is inside of the *BAC* folder. See [BACstage job file location on page 179](#).

### Related topics

- ◆ [Network Restore on page 168](#)
- ◆ [Backup Device on page 38](#)
- ◆ [Restore Device on page 38](#)

### Illustration 8-13 Network Backup dialog



**Known KMC Devices** A list of all discovered devices. Check only those devices that require backup. Use *Show Devices On This Network* to limit the list of devices to those found on a single network.

**File Name** BACstage uses the text in *File Name* in two places.

- ◆ BACstage creates a backup folder named from the entry in *File Name*. The backup folder is in the *BAC* folder inside of the current job folder.
- ◆ For each controller selected for backup, BACstage saves a *BAC* file. Each *BAC* file uses *File Name* with the device instance appended to it as the file name.

File\_Name[Device\_Instance.BAC

The default entry for *File Name* is the current calendar date. Other names may be entered to replace the default.

**Begin Backup of KMC Devices...** Click when devices are selected for backup.

**Network Backup Report:** A list of errors or other significant information.

**Show Devices On This Network:** Click to discover the devices on the network whose number is entered next to this button. If the network number is empty, then BACstage displays all devices it discovers on all of the networks.

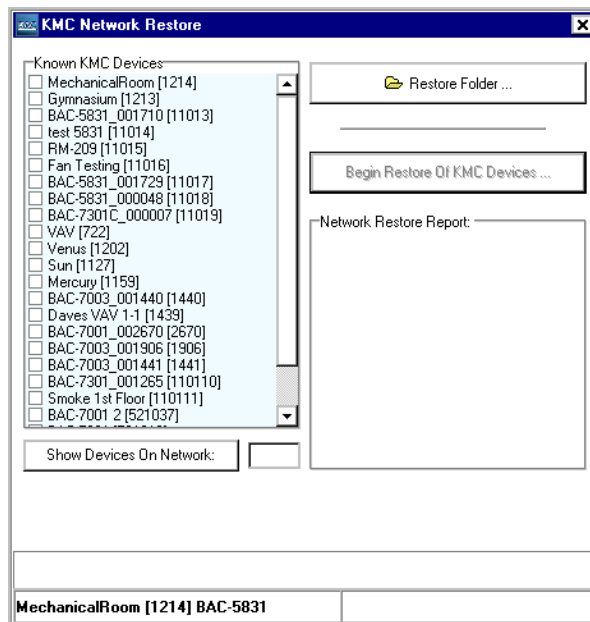
## Network Restore

Use *Network Restore* to retrieve controller configurations from a *BAC* files and send them to the selected controllers in the *Known KMC Devices* list. The *BAC* files must all be in one folder and the file names must follow the convention described in *File Name* on page 167.

### Related topics

- ◆ [Network Backup on page 167](#)
- ◆ [Backup Device on page 38](#)
- ◆ [Restore Device on page 38](#)

**Illustration 8–14 KMC Network Restore Dialog**





**Known KMC Devices** A list of all discovered devices. Check only those devices to which you want to send a BAC file. Use *Show Devices On This Network* to limit the list of devices to those found on a single network.

**Restore Folder...** Click to browse to the location of the BAC files you will send to controllers.

**Begin Restore of KMC Devices** Starts the process to read a BAC file, match the file to a controller with the same device instance and send the file to the controller.

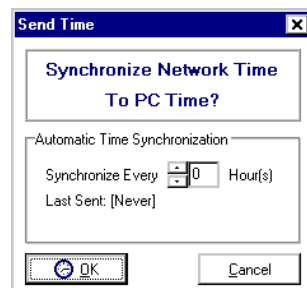
**Network Backup Report:** A list of errors or other significant information.

**Show Devices On This Network:** Click to discover the devices on the network whose number is entered next to this button. If the network number is empty, then BACstage displays all devices it discovers on all of the networks.

## Send Time

Use *Send Time* to synchronize the time and date of all controllers on the internetwork with the time and date of the computer on which BACstage is operating. The synchronization takes place at the interval specified in *Hours*. If *Hours* is zero or empty, BACstage does not perform synchronization. BACstage must be connected to the system to synchronize the time.

### Illustration 8-15 Synchronize Time dialog



Each KMC BACnet controller maintains its own time. Time synchronization may be performed by any of the following devices:

- ◆ A BACstage Operator Workstation
- ◆ A third-party device with time synchronization capability
- ◆ The KMC time master service
- ◆ A BACnet operator workstation with time synchronization capability.

See the topic [Timekeeping](#) on page 185 for additional information.

## Alarms/Events

Open the *Alarms/Events* list from the *Settings* menu to view, acknowledge or erase alarms and events received by BACstage.

- ◆ To view alarms and events in individual devices, see [Alarm/Event Summary](#) on page 37.
- ◆ To set up an alert sound, see [Global Settings](#) on page 18.

**Illustration 8–16 Alarms list**

#	Ack. Req'd	Alarm Message	Date/Time	Object ID	From State	To State	Notification Class	Priority	Event Type	Notify Type
1	Yes	<AV5> Temperature Present_Value =	2008/02/19 Tue 12:44:44.17	1213-AV5	Normal	Low Limit	1	255	Out Of Range	Event
2	Yes	<AV5> Temperature Present_Value =	2008/02/19 Tue 12:45:17.44	1213-AV5	Low Limit	Normal	1	255	Out Of Range	Event
3	Yes	Freeze stat New_State = Off, is Normal	2008/02/19 Tue 12:45:21.06	1213-BV6	OffNormal	Normal	1	255	Change Of State	Event
4	Yes	<AV5> Temperature Present_Value =	2008/02/19 Tue 12:47:37.74	1213-AV5	Normal	High Limit	1	255	Out Of Range	Event
5	Yes	Freeze stat New_State = On, is Offnormal	2008/02/19 Tue 12:47:41.76	1213-BV6	Normal	OffNormal	1	255	Change Of State	Event
6	Yes	<AV5> Temperat			Normal	Low Limit	1	255	Out Of Range	Event
7	Yes	Freeze stat New_s			Normal	Normal	1	255	Change Of State	Event
8	Yes	Freeze stat New_s			OffNormal	Normal	1	255	Change Of State	Event
9	Yes	<AV5> Temperature Present_Value =	2008/02/19 Tue 12:51:51.38	1213-AV5	Normal	Low Limit	1	255	Out Of Range	Event

### Features of the Alarms/Events list

**Ack.Req'd** The *Ack.Req'd* column is both a button and a status indicator. To acknowledge an alarm, click *Yes* when it appears in the *Ack.Req'd* column.

- ◆ *Yes* indicates the alarm or event requires an acknowledgement.
- ◆ *No* indicates the alarm or event does not require and acknowledgment.
- ◆ *Ack.d* indicates the alarm requires an operator acknowledgment.

**Alarm Message** A plain text explanation of the alarm or event.

**Date/Time** The time and date of the event as determined by the clock in the device that originated the event.

**Object ID** The device instance number and object that originated the event.

**From State** The event state of the originating object prior to the event that created the notification.

**To State** The event state of the originating object after the event that created the notification.

**Notification Class** The number of the notification class object associated with the event.

**Priority** This property specifies the priority of the event that has occurred. Priority is specified by the notification class object associated with this event. For a list of priority definitions, see the table *Alarm and event priority*.

**Event Type** This property displays the type of event that has occurred.

**Notify Type** Displays the type of event received. This property will be either Alarm or *Event*.

**Managing the Alarms/Events list**

Right-click the mouse to open the alarm list pop-up menu.

**Refresh** Retrieves the current alarms list and updates the list.

**Erase #** Removes only the selected alarm from the list.

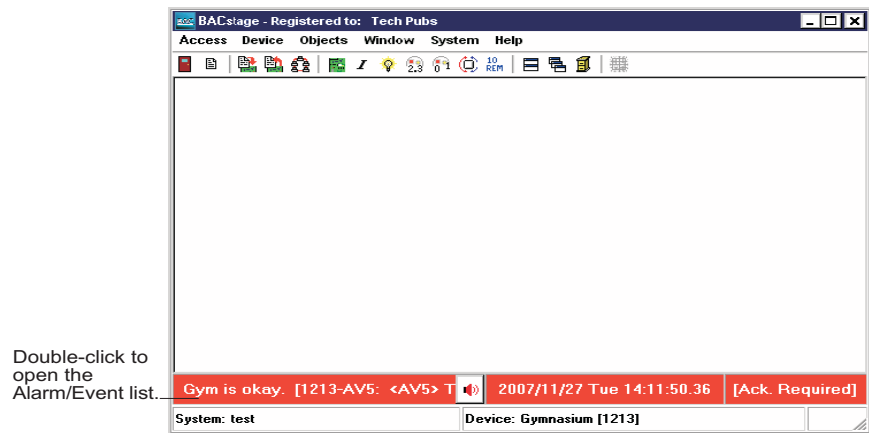
**Erase All** Deletes all alarms from the list.

**Open File** Opens and displays a selected alarm and event LOG file. The LOG files are stored in the Alarms folder inside of the job folder. See [BACstage job file location on page 179](#) for details about the job folder.

**Opening from alarm message**

The Alarm/Event summary list may be opened also by double-clicking on an alarm message in the BACstage work window. BACstage displays the alarm message only for alarms and not events.

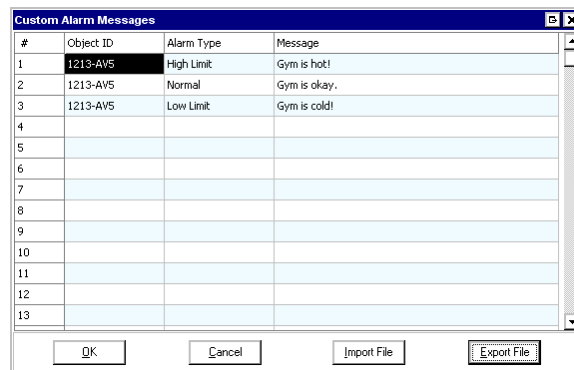
**Illustration 8-17 Alarm bar**



## Custom Alarm Messages

Custom alarm messages are text messages that are added by BACstage to the alarm/event list. When BACstage receives a notification, it compares the incoming device instance, object instance and alarm type from the notification to the device instance, object instance in the Custom Alarm Message list. If a match is found, the text of the custom message is added to the notification in the Alarm/Events list.

**Illustration 8–18 Custom Alarm Messages list**



#	Object ID	Alarm Type	Message
1	1213-AV5	High Limit	Gym is hot!
2	1213-AV5	Normal	Gym is okay.
3	1213-AV5	Low Limit	Gym is cold!
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			

**Device ID** Enter the device instance and object mnemonic of the object that initiated the notification. The format is device instance number, a hyphen (-) and followed by the mnemonic of the object. For example, 1213-AI3. For a list of mnemonics see the topic [Programming with mnemonics on page 110](#).

**Alarm Type** Choose from the following alarm types to associate with the notification:

- ◆ High Limit
- ◆ Normal
- ◆ Low Limit
- ◆ Off Normal
- ◆ Fault
- ◆ Life Safety Alarm

**Message** Add the custom text to appear in the Alarms/Events list.

**Export File** Use Export File to save the information in the Custom Alarm Messages list to a comma separated file. The file can then be used for system back up or it can be moved to other computers running BACstage that receive notifications.

**Import File** Use Import File for either of the following:

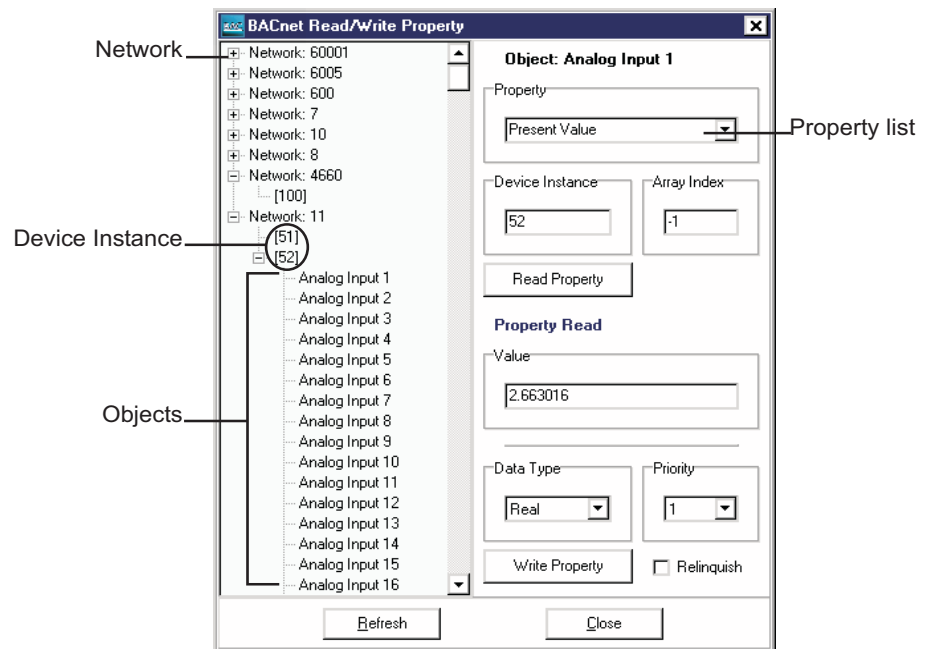
- ◆ Restore the Custom Alarm Messages list with the contents of a back up file.
- ◆ Set up a computer running BACstage with the same list as another computer running BACstage.

## BACnet Read/Write Property

Use the BACnet *Read/Write Property* dialog to change the value of properties of objects in controllers to which BACstage is connected. Through this dialog box, properties can be changed in not only KMC Controls BACnet controllers but also controllers from other vendors. Only those properties from other vendor's controllers that are also supported by KMC BACnet controllers are available in the *Read/Write Property* dialog box.

Choose *BACnet Read/Write Property* from the *System* menu to open the dialog box.

**Illustration 8-19 Read/Write Property**



All devices on the internetwork to which BACstage is connected are displayed when *Read/Write Property* opens. To modify the value of a property:

1. Scroll through the list until the targeted device is at the top of the list.
2. Select the + check box next to the device to expand the list of objects within the selected device.
3. Choose an object from the list.
4. Choose the property to modify.
  - Choose the property from the Property drop-down list.
  - Enter a BACnet property ID number
5. Click *Read Property*.
6. Change the value as required.
7. Click *Write Property* when finished.

**Tip:**

Not all properties may be changed with *Read/Write Property* because of the manner in which the property is implemented within the device.

## Section 9: The Help menu

Choosing *Contents* menu within *Help* displays additional topics. Click on a book next to the main topic. Sub topics are displayed beneath the main topic.

---

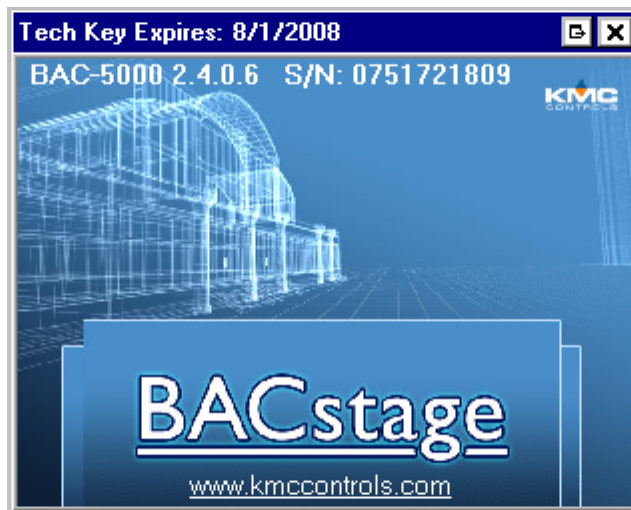
Topics included in the Help Menu

- ◆ [About BACstage on page 175](#)
- ◆ [Contents on page 3](#)

### About BACstage

Displays the current BACstage software version number, hardware key serial number and copyright information.

**Illustration 9–1 About BACstage**

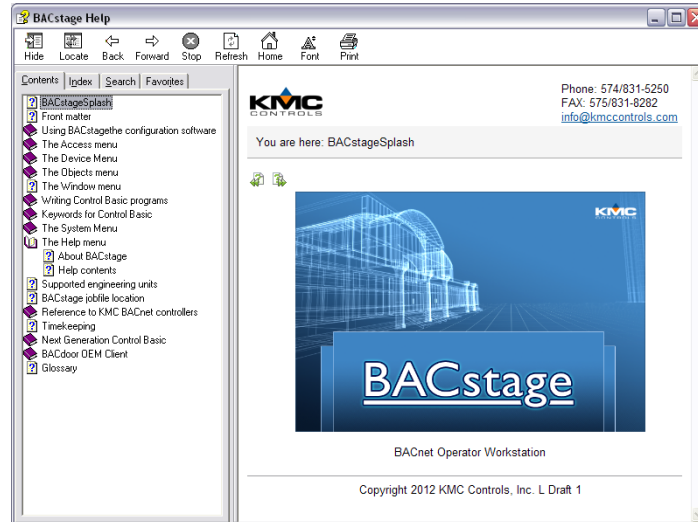


Use the version number to differentiate between new and old versions of software. It is also a convenient reference in determining what features are included in this version of software. If you have any questions concerning BACstage and wish to get assistance, be sure to check the version number before calling for technical support.

## Contents

Opens BACstage help to the Contents tab. Help also includes an index and search tab.

**Illustration 9–2 BACstage help**





# Appendix A: Supported engineering units

BACstage supports the engineering units listed in this section.

**Table A-1 Analog units**

%	DegC/min	joules/kg-dry-air	m2	no-units
%/sec	deg-days-C	joules/sec	m2/newton	ohm-meters
%Obscuration/ft	deg-days-F	KBTU	m3	ohms
%Obscuration/m2	DegF	KBTU/hr	m3/hr	Pa
%RH	DegF/hr	kg	m3/min	PF
/hr	DegF/min	kg/hr	m3/sec	ppb
/min	DegK	kg/m3	mA	ppm
/sec	DegK/hr	kg/min	MBTU	psi/DegF
amperes	DegK/min	kg/sec	meters	radians
amps/m	degrees-phase	Khertz	Meters/s/s	radians/sec
amps/m2	delta-deg-F	KJ/kg	Mhertz	RPM
amps-sq-m	delta-deg-K	KJoules	millibars	sdays
bars	farads	Kjoules/DegK	minutes	sec/100
BTU	foot-candbles	Kjoules/kg-dry-air	Mjoules	seconds
BTU/hr	ft	KMH	Mjoules/DegK	siemens
BTU/lb	ft/min	KOhms	Mjoules/ft2	siemens/m
BTU/lb-dry-air	ft/sec	Kpa	Mjoules/kg-dry-air	teslas
candelas	ft2	KV	Mjoules/m2	therms
candelas/m2	ft3	KVA	mm	ton-hours
cm2	ft3/min	KVAR	mm/min	tons
cm2	ft3/sec	KWH	mm/sec	tons/hr
cm-mercury	grams/min	KWH	mm-mercury	Tons-R
cm-water	grams/sec	KWH/ft2	MOhms	US-Gal
currency1	gr-water/kg-dry-air	KWH/m2	mOhms	US-Gal/min
currency10	hectoPa	L/hr	months	V/DegK
currency2	henrys	L/min	MPH	V/m
currency3	hertz	L/sec	msec	VA
currency4	horsepower	lbf/in2	MV	VAR
currency5	hours	lb-mass	MVA	Volts
currency6	lgal/min	lb-mass/hr	MVAR	watt-hours
currency7	Imperial-Gal	lb-mass/min	mVolts	watts
currency8	in	lb-mass/sec	MW	watts/ft2
currency9	in2	liters	mwatts	watts/m/DegK
cycles/hr	in-mercury	lixes	MWH	watts/m2
cycles/min	inw	lumens	newton	watts/m2/DegK
Deg-Angular	joules	m/hr	newton-m	webers
DegC	joules/DegK	m/min	newtons/m	weeks
DegC/hr	joules/kg-DegK	m/sec	newton-sec	years

**Table A-2 Binary unit pairs**

<b>Normal Inactive</b>	<b>Normal Active</b>
OFF	ON
Stop	Start
Normal	Alarm
Closed	Open
Cool	Heat
Unocc	Occupied
Disable	Enable
Normal	High
Normal	Low
No	Yes
Low	High
Inactive	Active

## Appendix B: BACstage job file location

This section describes the location and contents of the BACstage job folders.

---

As systems are added to the BACstage system list, job folders are created in the BACstage folder. The name of the job folder is the same as *System Name* in the system list. The job folder includes other folders in which BACstage stores specific data and information about the job.

Windows XP job folder location:

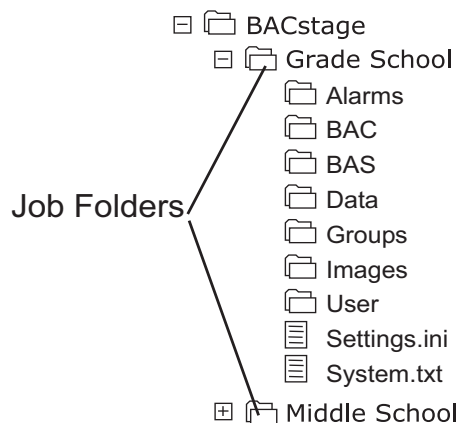
C:\Documents and Settings\All Users\Application Data\KMC Controls

Windows Vista and Windows 7 job folder location:

C:\Program Data\KMC Controls\BACstage

See [System List on page 14](#) for additional information.

### Illustration B-1 BACstage job folders



**Alarms** The folder in which BACstage automatically stores the alarm and event LOG files. The alarm and event files use the following naming format.

091404.LOG  
Month |  
Day |  
Year |

**BAC** The default location for storing the BAC controller configuration files. The files are in text format and can be opened with Notepad or Wordpad.

**BAS** The default location for storing Control Basic files. The files are in text format and can be opened with Notepad or Wordpad.

- ◆ Standard Control Basic files are saved with the extension of BAS.
- ◆ Next Generation Control Basic files are saved with the extension of NG.

**Data** Stores the trend log data. The value #*n* is a log sequence number assigned by BACstage as each log file is saved.

111111-TR1#n.LOG

Device Instance of the controller \_\_\_\_\_  
 Trend number \_\_\_\_\_

**Groups** Stores the data for the groups object. Reserved

**Images** Stores the background and animated graphics for group displays.

Background graphic file formats are:

- ◆ Non-animated GIF
- ◆ JPG
- ◆ WMF
- ◆ BMP

Animation files must be in the animated GIF format.

**User** Stores the user log files. A log file is created for each day that BACstage is started. The log file stores a record of operator sign-on, sign-off and other significant events. The files are in text format and can be opened with Notepad or Wordpad.

## Appendix C: Reference to KMC BACnet controllers

The information in this appendix is a quick reference to the major features of BACnet controllers.

---

The information in the following sections lists the characteristics of the BACnet objects in KMC BACnet controllers.

- ◆ For detailed specifications for each controller see the installation and operation guide packed with the controller.
- ◆ For information about connecting controllers to a building automation system see the installation and operation guide packed with the controller or SP-022, “The Digital Designer’s Guide”.

Controllers are grouped as follows:

- ◆ For general purpose controllers, see [General purpose controllers](#).
- ◆ For controllers with factory programming for specific applications, see [Job specific controllers on page 182](#).

### General purpose controllers

The general purpose controllers include both BACnet Advanced Applications Controllers and a BACnet Building Controller.

- ◆ Do not include factory Not factory supplied Control Basic programming.
- ◆ Inputs can be configured as analog, accumulator or binary objects.
- ◆ Outputs are 0-10 volt DC outputs. Accessory cards are available for 0-12 volts DC, relay, triac or 4-20 milliampere current loop outputs.

Table C-1 BACnet General purpose controllers

	BAC-5801 BAC-5802*	BAC-5831	BAC-A1616BC
Function and type	General AAC	General AAC	General BBC
Universal inputs Analog, accumulator or binary	8	16	16 Expandable
Outputs Analog or binary	8	12	16 Expandable

**BACnet General purpose controllers (continued)**

	<b>BAC-5801</b>	<b>BAC-5831</b>	<b>BAC-A1616BC</b>
			<b>BAC-5802*</b>
Binary value objects	40	40	100 Configurable to 1,000
Analog value objects	40	40	100 Configurable to 1,000
Multistate value objects			10 Configurable to 256
PID loop objects	8	12	16 Configurable to 32
Weekly schedule objects	8	8	10 Configurable to 100
Calendar objects	3	3	10 Configurable to 32
Tables (User Defined)	2 (+3)	2 (+3)	16
Trend objects	8	8	10 Configurable to 256
Notification objects	8	8	10 Configurable to 64
Event enrolment objects			10 Configurable to 512
Programs objects	10	10	32
Control Basic	Standard	Standard	Next Generation

\* The clock in BAC-5802 is software based.

## Job specific controllers

The job specific controllers include Control Basic programs for specific HVAC functions. All programs can be modified or deleted with BACstage or TotalControl.

- ◆ Inputs can be configured as analog, accumulator or binary objects.
- ◆ Outputs are either 0-10 volt DC or where noted, relay or triac.
- ◆ All VAV models include one input dedicated to the airflow sensor and one output dedicated to the damper motor.
- ◆ Models ending with C do not have a hardware based real-time clock.

**Table C-2 Job specific controllers**

	<b>BAC-7001</b> <b>BAC-7051</b>	BAC-7003 BAC-7053	BAC-7301 BAC-7301C	BAC-7302 BAC-7302C	BAC-7303 BAC-7303C	BAC-7401 BAC-7401
Function and type	VAV* AAC	VAV* AAC	AHU AAC	RTU AAC	FCU AAC	HPU AAC
Universal inputs Analog, accumulator or binary	3	3	4	4	4	4
Airflow input	Yes	Yes				
Outputs Analog or binary	3	1	3	1	2	4
Outputs, single stage triac		1	1	1	1	
Outputs, dual-stage triac				2	1	
Outputs, relay		1				
Binary value objects	40	40	40	40	40	40
Analog value objects	40	40	40	40	40	40
PID loop objects	4	4	4	4	4	4
Weekly schedule objects	8	8	8	8	8	8
Calendar objects	3	3	3	3	3	3
Programs objects	10	10	10	10	8	8
Tables (User Defined)	2 (+3)	2 (+3)	2 (+3)	2 (+3)	2 (+3)	2 (+3)
Trend objects	8	8	8	8	8	8
Notification objects	8	8	8	8	8	8
Control Basic	Standard	Standard	Standard	Standard	Standard	Standard





## Appendix D: Timekeeping

This section describes several aspects of BACnet timekeeping systems on KMC BACnet networks.

- ◆ Standard BACnet time services
- ◆ The KMC Controls unconfirmed private transfer (UPT) timekeeping system
- ◆ A typical small system that uses the UPT timekeeping system

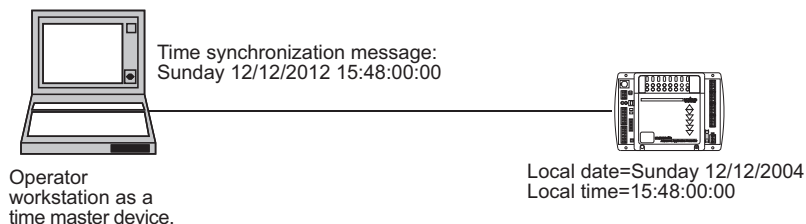
### Standard BACnet time services

BACnet controllers and devices that must perform functions on a schedule keep track of time with internal real-time clocks. A time master device synchronizes these internal clocks by sending time synchronization messages. The messages are sent either at regular intervals or on command of an operator. All receiving devices, upon receipt of the message, update their local clocks from the time and date in the message. The BACnet standard specifies two types of time synchronization services.

**Simple time synchronization** The time master device sends the time synchronization message in local time and all controllers then use the date and time from the message to update their own internal clock. This is the time service supported by KMC Controls controllers and BACstage.

**UTC time synchronization** The time master device sends a time synchronization message in universal time (UTC). The controllers then apply an offset to calculate local time and date. The offset, which is stored in the controllers, is the difference, expressed in minutes, between local time and UTC. Controllers that support the UTC service also support daylight saving time.

### Illustration D-1 BACnet time synchronization



### KMC time master devices

BACstage and its BACdoor OEM Client function as a time master device. When BACstage is connected and time synchronization is enabled, all

controllers on the internetwork synchronize their clocks to the time message from BACstage. BACstage uses computer time for the time message. When using BACstage, time synchronization may be enabled in either of two places:

- ◆ *Send Time* under the *System* menu in BACstage sets synchronization in one-hour intervals.
- ◆ *Time Sync Interval* in BACdoor sets synchronization in one-minute intervals.

### **Hardware and software clocks**

Controllers with software based real-time clocks will not maintain correct time without power. After a power outage, the time in the controller will be incorrect by the period of the outage. For example, the clock in a controller that loses power for 30 seconds will restart when power returns but will operate 30 seconds behind actual time. Unless a time master device resets the clocks in these controllers, they will continue to operate with incorrect time. The KMC Controllers with hardware-based real-time clocks maintain correct time without power for up to 72 hours. These controllers, however, are not designed as time master devices. For systems without a permanent time master device, KMC Controls developed an additional method for time synchronization.

### **KMC UPT time synchronization**

Beginning with firmware release 1.4, KMC BACnet controllers include a background method that automatically restores correct time to KMC BACnet controllers with software real-time clocks. This service shares time information among KMC BACnet controllers with the BACnet unconfirmed private transfer (UPT) service.

The KMC UPT time synchronization service is a background method that controllers with software clocks use to acquire the correct time and date from a controller with a hardware clock. The process begins when a KMC BACnet controller with a software real-time clock powers up. The controller begins sending Time\_Request UPT messages to each controller on the network starting with address MAC 1. It pauses four seconds and then sends a request to the next address. The controller continues to poll, in sequence, all MAC addresses, except its own, from MAC 1 to the address specified by the value set by MAX MASTER. Polling continues until the controller receives a valid time synchronization message.

A KMC controller with a hardware clock that has a valid date and time that receives the Time\_Request UPT responds with a Time\_Announce UPT message. Because a controller may be responding to multiple requests, responses are sent with at least two seconds between Time Announce\_UPT messages.

All controllers with software clocks on the local network that require date and time updates will use any Time Announce UPT message to reset their clocks. Once a controller has a valid date and time, it stops sending the Time\_Request UPT messages. If a controller sees a time synchronization message from a time master device, it will stop sending Time\_Request UPT messages. The KMC UPT time synchronization requires no special configuration but some key points must be followed for the system to work efficiently and correctly.

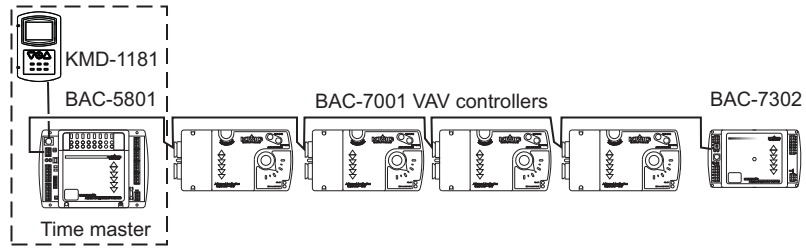
- ◆ The KMC UPT time service augments standard BACnet time keeping where a permanent time source—such as BACstage or third-party time master device—is not available.
- ◆ The KMC UPT time service is functional only between KMC BACnet controllers on the same local network.
- ◆ At least one controller with a hardware based real-time clock must reside on the MS/TP network where the service is required.
- ◆ Assign to the controller with the hardware clock a MAC address less than 10 (BACstage limits MAC address for controllers to 1 or higher). Assigning the controller a low value MAC address will ensure that time request messages are answered early in the polling cycle.
- ◆ Configuring Max\_Masters consistently within each of the KMC controllers on the network will limit unnecessary polling of nonexistent MAC addresses.
- ◆ Controllers with hardware real-time clocks will not synchronize their clocks by the UPT time service method. They must be synchronized by BACstage or other time master device.

### **A typical system**

The following illustration is a small system that uses the UPT time keeping method. This system is typical of small retail stores or schools. It is composed of a BAC-5801 with a NetSensor connected, several BAC-7001 VAV controllers and a BAC-7302 rooftop unit controller.

- ◆ The NetSensor connected to the BAC-5801 is configured as described below. Use the NetSensor to set system time if BACstage is not available.
- ◆ The BAC-5801 will maintain the time as set by the KMD-1181 NetSensor.
- ◆ As each of the BAC-7001 controllers and the BAC-7302 power up, they will seek a time master device which will trigger the BAC-5801 to send the Time\_Request UPT message.

**Illustration D-2 BAC-5801 as a time master**



Beginning with firmware release 1.4 and BACstage 2.1, NetSensors with firmware 1-E support setting system time. This feature is for single controllers or very small networks without readily available time master devices. The NetSensor changes time in only the controller to which it is connected.

## Appendix E: Next Generation Control Basic

Next Generation Control Basic is an advanced version of Control Basic that is supported in new BACnet controllers. This section explains the details of Next Generation Control Basic and the differences between standard and Next Generation Control Basic.

---

With the release of BACstage 2.3 and TotalControl Design Studio 1.5, KMC Controls introduced several changes to Control Basic.

- ◆ Controllers with Standard Control Basic are programmed with little change. However, when existing programs are loaded from a controller you will see some changes to the keywords and references to remote points. See the topic [Control Basic versions in controllers on page 190](#) for a listing of controllers that support Standard Control Basic.
- ◆ Controllers with Next Generation Control Basic have several new keywords available. In addition there are other changes to the language.

Review the following topics to become familiar with the new features of Control Basic:

- ◆ [Control Basic versions in controllers on page 190](#)
- ◆ [Changes to IF THEN on page 190](#)
- ◆ [Deprecated keywords on page 191](#)
- ◆ [File compatibility on page 192](#)
- ◆ [Keywords for new functions and statements on page 193](#)
- ◆ [Labels in Next Generation Control Basic on page 194](#)
- ◆ [Line numbers on page 195](#)
- ◆ [Local variables on page 195](#)
- ◆ [References to objects in remote devices on page 195](#)
- ◆ [Using the conversion tool on page 196](#)

## Control Basic versions in controllers

The following table lists the versions of Control Basic that are supported in the BACnet controllers from KMC Controls.

**Table E-1 Versions of Control Basic**

<b>Model number</b>	<b>Control Basic version</b>	<b>Control Basic file format</b>
BAC-A1616	Next Generation	NG
BAC-5801 BAC-5802	Standard	BAS
BAC-5831	Standard	BAS
BAC-5841 BAC-5842	Standard	BAS
BAC-7001 BAC-7051	Standard	BAS
BAC-7003 BAC-7053	Standard	BAS
BAC-7301 BAC-7301C	Standard	BAS
BAC-7302 BAC-7302C	Standard	BAS
BAC-7303 BAC-7303C	Standard	BAS
BAC-7401 BAC-7401C	Standard	BAS
FlexStat BAC-10000 BAC-11000 BAC-12000 BAC-13000 BAC-14000	Next Generation	NG

**Changes to IF THEN** Next Generation Basic now supports block and nested IF THEN statements.

```
Locals ChilledWaterSetpoint
```

```
AV24 = ChilledWaterSetpoint
```

```
IF BV258 THEN
```

```

        ChilledWaterSetpoint = 52
        ELSE
        Chilledwatersetpoint = 48
    ENDIF

    IF TIME > 7:00 THEN
        IF TIME < 9:00 THEN
            START BO1
        ENDIF
    ENDIF
ENDIF

```

## Deprecated keywords

The use of the keywords in the table “Control Basic deprecated keywords” on page 1 change in BACstage version 2.3. Only the keywords are changed; the functions and statements they represent remain the same.

- ◆ When writing programs for controllers with Standard Control Basic (see [Control Basic versions in controllers on page 190](#)), BACstage will accept and compile *either* the deprecated keywords or the replacement versions of the keywords. For example either DEW-POINT or DEWPOINT may be used when writing a program.
- ◆ When BACstage or TotalControl retrieves a Control Basic program from a controller and decompiles it, the deprecated keywords are replaced with the new keywords. For example DEW-POINT becomes DEWPOINT and TIME-ON becomes TIMEON.
- ◆ For controllers with Next Generation Control Basic, BACstage or TotalControl will not accept or compile the deprecated keywords in the following table.

**Table E-2 Control Basic deprecated keywords**

Deprecated keyword	Replacement keyword
COS-1	ARCCOS
DEW-POINT	DEWPOINT
DEW-POINT-SI	DEWPOINTS
ENTHALPY-SI	ENTHALPYSI
LN-1	INVLN
MODEL-NUMBER	MODELNUMBER
NETSENSOR-STATUS	NETSENSORSTATUS
ON-ERROR	ONERROR

**Control Basic deprecated keywords (continued)**

<b>Deprecated keyword</b>	<b>Replacement keyword</b>
OUTPUT-OVERRIDE	OUTPUTOVERRIDE
PANEL-ADDRESS	PANELADDRESS
SCHED-ON	SCHEDON
SCHED-OFF	SCHEDOFF
SENSOR-OFF	SENSOROFF
SENSOR-ON	SENSORON
SIN-1	ARCSIN
TAN-1	ARCTAN
TIME-ON	TIMEON
TIME-OFF	TIMEOFF

**File compatibility**

When saving and opening files with versions of BACstage other than BACstage 2.3, be aware of the following compatibility issues.

- ◆ Programs sent to a controller with BACstage 2.3 can be loaded from a controller with earlier versions of BACstage. BACstage 2.2 and earlier will list the programs using the deprecated keywords and original syntax for remote points.
- ◆ If a .BAS file includes any of the new or deprecated keywords, BACstage versions earlier than 2.3 will open but not compile the program. The new keywords and syntax must be changed to the original format.
- ◆ When transferring a .BAS file to a controller with Next Generation basic, the .BAS files can be converted to Next Generation format (.NG extension) with the conversion tool. See [Using the conversion tool on page 196](#). Controllers that support Next Generation Control basic are listed in the topic [Control Basic versions in controllers on page 190](#).
- ◆ Files created with the *Backup Device* (.BAC files) in BACstage 2.3 are backwards compatible with BACstage 2.2 and earlier.



## Keywords for new functions and statements

The following keywords are added to Control Basic:

- ◆ ALIAS
- ◆ BIND
- ◆ CONST
- ◆ FLUSH
- ◆ HALT
- ◆ ISNAN
- ◆ LOCALS
- ◆ NAN

A brief description of each of the new keywords follow. Unless noted otherwise, the new keywords can only be used when writing programs for controllers with Next Generation Control Basic.

### ALIAS

Declares a local variable and dynamically binds the value of a property to the variable. It also sets two intervals at which Control Basic will read from or write to the property bound to the variable.

**Syntax:** *ALIAS(device, object, property, local, read interval, write interval)*

See [ALIAS on page 114](#) for a more detailed description.

### BIND

Binds a device instance to a physical network address. This is typically used to bind an MS/TP slave to a master device. BIND is required in only one program within the device.

**Syntax:** *BIND (device, network, mac, option)*

See the keyword [BIND on page 116](#) for a more detail description.

### CONST

Use to declare one or more variables and assign to them a fixed value. Do not use with variables that change with subsequent steps in the program.

**Syntax:** *CONST, variable, variable, ...*

See [CONST on page 117](#) for a more detailed description.

### FLUSH

When a FLUSH statement runs, Control Basic immediately reads from or writes to the property bound to the local variable declared by ALIAS.

**Syntax:** *Flush (LocalAlias)*

See the topics [FLUSH on page 122](#) and [ALIAS on page 114](#) for more detailed descriptions.

### HALT

Stops the program from running and sets the *Program State* property in the program object to *Halted*. The string *Message* is displayed in the property *Description of Halt*. The program can be restarted by doing any of the following:

**Syntax:** `HALT "Message"`

See the keyword [HALT on page 124](#) for a more detailed description.

### ISNAN

ISNAN tests the value of *expression* to determine if it is a valid number. If the value of *expression* is equal to *NAN* (Not A Number), then ISNAN returns *true*. A typical use of ISNAN is to test the present value property of an object in a remote device.

**Syntax:** `ISNAN( _expression_ )`

See [ISNAN on page 128](#) for a more detailed description.

### LOCALS

Use to declare local variables. A local variable may be used only within the program in which it is declared.

**Syntax:** `LOCALS variable[, variable, ...]`

See the keyword [LOCALS on page 130](#) for a more detailed description.

### NAN

Use NAN to set a variable or property to a *Not A Number* constant or to test if the variable or property is equal to *Not A Number*. NAN can be used in both Standard and Next Generation Control Basic.

See [NAN on page 132](#) for a more detailed description.

## Labels in Next Generation Control Basic

In Next Generation Control Basic, labels are used instead of line numbers when program flow is redirected with any of the following statements.

- ◆ GOSUB
- ◆ GOTO
- ◆ ONERROR
- ◆ ON GOSUB
- ◆ ON GOTO

In the following program example, *CoolMode* and *HeatMode* destinations of a program redirection.

```

        IF T > 55 THEN GOTO CoolMode
        IF T <= 55 THEN GOTO HeatMode
        END
CoolMode:
        REM Cooling sequence runs here
        END
HeatMode:
        REM Heating sequence runs here
        END

```

Declare labels by typing a name followed immediately by a colon (:).

- ◆ A label can be any combination of letters (A-Z or a-z), numbers (0-9) or the underscore (\_).
- ◆ Labels are not case sensitive.
- ◆ Labels are unique to the program in which they are declared.
- ◆ Labels cannot be a Control Basic keyword.

## Line numbers

Line numbers are not used in Next Generation Control Basic programs. However, a line number is displayed in the Control Basic editors for both BACstage and TotalControl Design Studio. The line numbers displayed are only for identification of problems when the program is compiled.

Line numbers continue to be used in controllers with Standard Control Basic.

## Local variables

The single-letter local variables a-z and A-Z may still be used without program modification. In addition to single letters, more descriptive variables may be used by declaring a variable with the statement LOCALS. However, once a variable is declared, all single letter variables used in the program must also be declared. For details on declaring local variables, see the keyword [LOCALS on page 130](#).

## References to objects in remote devices

When referring to an object in a remote device, the device name or instance is now separated from the object reference by a period(.). In previous versions, the instance and name were separated with a dash (-).

```
10 A = 1214.AI1
```

In BACstage the name of the device or object can be used in place of an instance number.

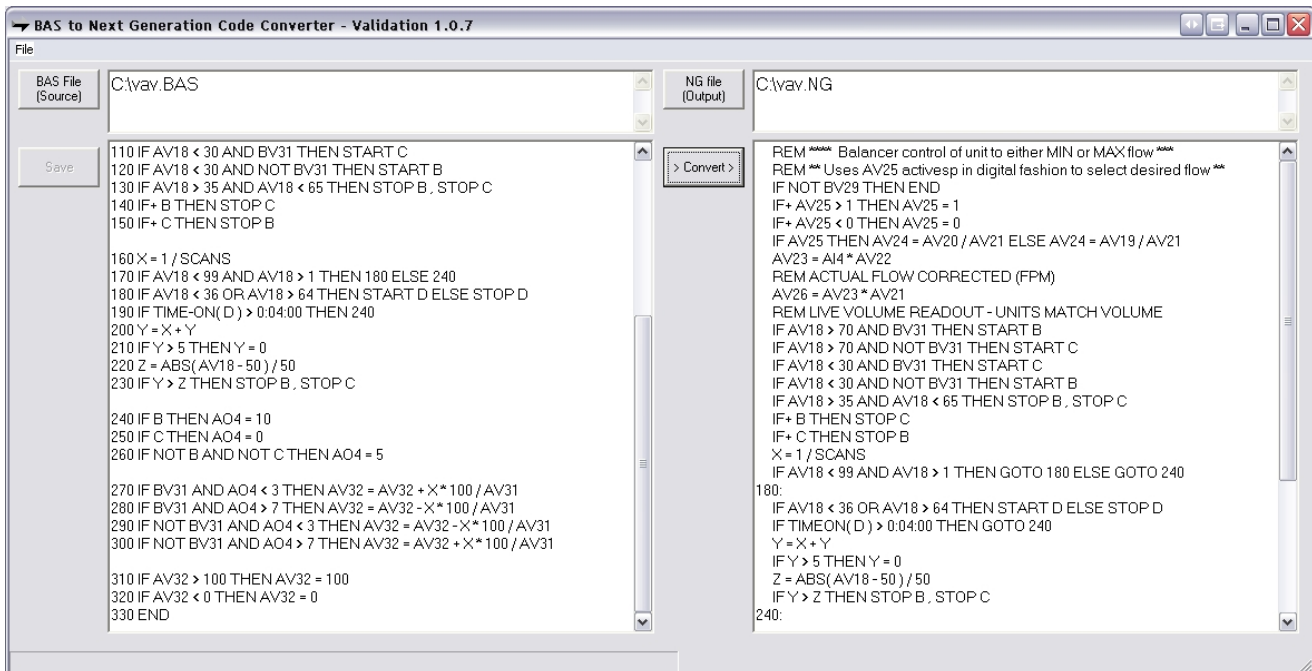
```
20 A = MechanicalRoom.TempMechRoom
30 A = 1214.TempMechRoom
40 A = MechanicalRoom.AI1
```

## Using the conversion tool

A conversion tool is available on the KMC Controls web site that converts .BAS files to .NG files. Open an existing .BAS file and then convert it to the Next Generation format. Once converted you can do either of the following:

- ◆ Save the file to disk.
- ◆ Copy all or part of the text and then paste it into a Control Basic editor.

Illustration E-1 Control Basic converter tool



## Appendix F: BACdoor OEM Client

BACdoor OEM client is the driver that connects BACstage to the BACnet internetwork.

---

- ◆ [Opening BACdoor on page 197](#)
- ◆ [Settings in BACdoor OEM client on page 199](#)
- ◆ [Installing drivers for BACnet 8802-3 \(Ehternet\) on page 203](#)

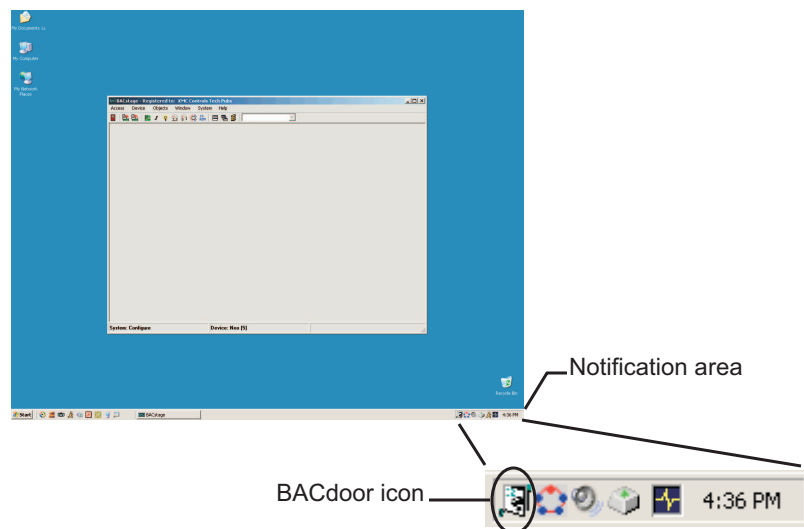
### Opening BACdoor

Most functions of BACdoor can be configured by choosing Connection Parameters from the BACstage access menu.

To configure BACdoor with the BACdoor configuration, do the following:

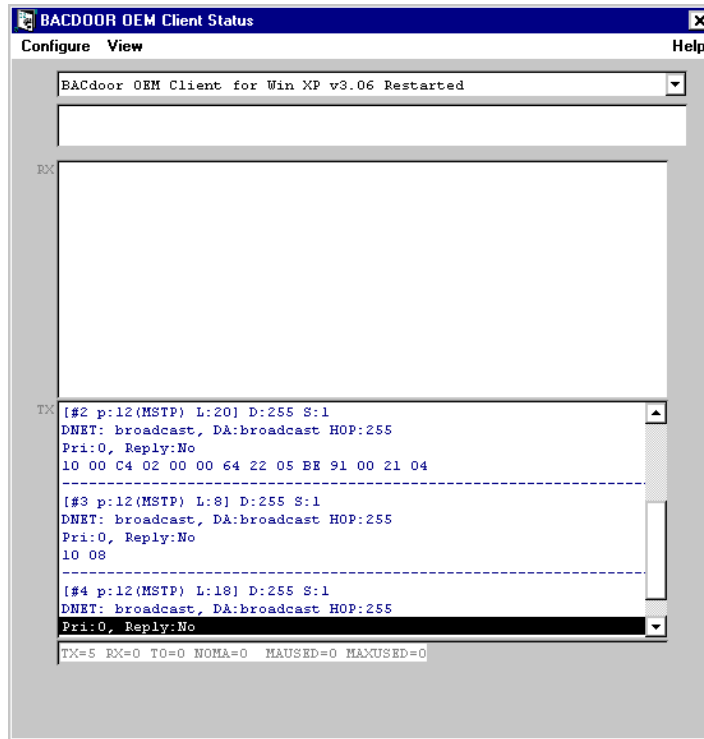
1. Start BACstage.
2. If the BACdoor icon is not in the system tray, choose *System List* from the *Access* menu and then choose any system. The BACdoor icons appear in the Windows Notification Area.

#### Illustration F-1 BACdoor icon in Notification Area



3. Click the BACdoor icon in the system tray. The BACdoor Client Status window opens.

Illustration F-2 BACdoor client status



4. Click *Configuration* in the upper left corner of the client status dialog. The *BACdoor Client Configuration* opens.

**Illustration F-3 BACdoor client configuration**

5. Change the settings as required for the internetwork on which BACstage is operating.
6. Click *Close* to return to BACstage.

*Related topics*

- ◆ [Settings in BACdoor OEM client on page 199](#)
- ◆ [System List on page 14](#)
- ◆ [Establishing point-to-point communications on page 20](#)

**Settings in BACdoor OEM client**

Use BACdoor OEM Client to configure BACstage for the internetwork on which it running.

- ◆ [BACnet/IP Parameters on page 201](#)
- ◆ [MS/TP Parameters on page 201](#)
- ◆ [PTP Parameters on page 202](#)
- ◆ [Parameters for Connecting to Routers on page 203](#)

**Illustration F–4 BACdoor client configuration**

BACDOOR OEM Client Configuration	
Our Device Instance:	90
Our Peername:	Tech Pubs
Time Sync Interval:	5 Minutes (0=None)
Whols/Iam Interval:	1 Minutes (0=None)
Nretry:	3
Window Size:	1
TX Length:	1470 Octets
RX Length:	1470 Octets
Maximum Length Assembled APDU:	8192 Octets
BACnet/IP Parameters	
169.254.95.22 [255.255.0.0] 3Com 3C920 Integrated Fast Ethernet Contr...	
UDP port:	0xBAC0
Subnet:	255.255.0.0
MS/TP Parameters	
Com Port Init (restart):	COM1:38400,N,8,1
TS (MS/TP Node):	0
MaxMaster:	127
MaxInfoFrames:	20
PTP Parameters	
<input type="checkbox"/> Dialed	Non-Dialed Com Port Init (restart): COM1:9600,N,8,1
SNET	4660
SLEN	1
SADR	00
Parameters for Connecting to Routers	
Time to wait for Connection	30 Seconds
Time to wait for Disconnect	5 Seconds
Default Tactive	5 Seconds

### BACstage internetwork

These are parameters that must be configured for BACstage regardless of the type of connection.

**Our Device Instance** A number that uniquely identifies BACstage as a BACnet device on the internetwork. The device instance number is assigned by the BACnet system designer. Valid instance number's range from 0 to 4,194,303. It is by reference to the device instance number that data is exchanged between BACnet devices.

**Our Peername** A 16-character name for the BACstage operator workstation and must be unique among all devices on the internetwork. The set of characters used in *Our Peername* is restricted to printable characters.

**Time Sync Interval** Sets the interval at which BACstage will update the master time device with the time in the computer on which BACstage is running.

**Who Is?/I Am Interval** Sets the interval between automatic Who Is? broadcast messages.



### BACnet/IP Parameters

Use the BACnet/IP Parameter to configure BACstage for the IP network to which it is connected.

**Register as Foreign Device with BBMD at IP** Select to register BACstage as a foreign device with a BACnet Broadcast Management Device. Foreign device registration to a BBMD is a technique for crossing an IP-only router with BACnet broadcast messages.

Enter also the IP address of the BBMD. If network address translation (NAT) is used between the BACstage computer and the BBMD, contact the network system administrator for the correct public IP address.

**Registration Time-to-Live** (For Foreign Device only) Sets the interval at which BACstage sends a registration message to the BBMD with which it is registered.

If the BBMD does not receive a registration message within the Time-To-Live period plus 30 seconds, the BBMD will remove BACstage from its foreign device table and will not send broadcast messages to BACstage .

- ◆ The valid time range is 1 to 65535 seconds.
- ◆ If the entry is zero (0), the registration is forever.
- ◆ The default value is 30.

**Subnet** The IP subnet to which the computer running BACstage and BACdoor is connected.

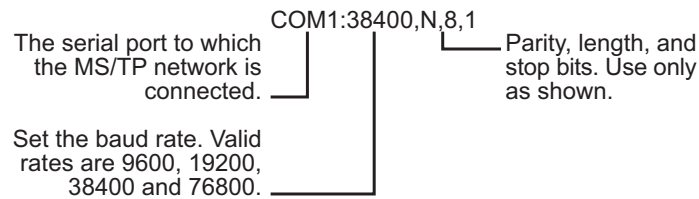
**UDP Port** Sets the BACnet UDP port number which is supplied by the network system administrator.

- ◆ The port must match the port in use by BACnet devices on the network to which BACstage is connecting. Valid port numbers are 0xBAC0 in hexadecimal notation (47808 in decimal notation) to 0xBAC9 (47817).
- ◆ When registered as a foreign device, enter the port number of the BBMD. If port address translation (PAT) is used between the local router and the PAD or BBMD, contact the network system administrator for the correct public IP address.

### MS/TP Parameters

Use these settings when connecting BACstage to an internetwork with an MS/TP connection.

**Com Port Init (restart)** Enter the character string to match the parameter of the MS/TP network to which BACstage is connected. Use only the settings shown below.

**Illustration F-5 MS/TP parameters command string**

**TS (MS/TP Node)** TS (This Station) is equivalent to the MAC address in a KMC BACnet controller or router. This number assigns to BACstage a node number on the MS/TP network to which it is connected. The number must be unique on the local network but, may be duplicated on remote MS/TP networks.

**Max Master** Indicates the highest Media Access Control (MAC) address assigned to any device on the MS/TP network to which the device is connected.

- ◆ Setting *Max Master* significantly higher than the highest numbered device may result in increased polling and slower response times.
- ◆ Setting *Max Master* lower than the highest numbered device will result in devices not appearing on the network.

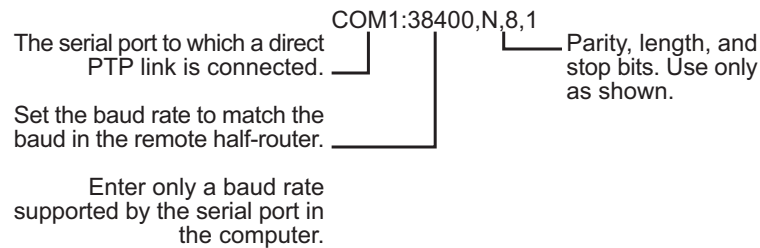
**MaxInfoFrames** Sets the maximum number of packets that are sent before passing the token. *Max. Info Frames* affects response time and throughput. The recommended setting is 20.

**PTP Parameters**

Use *PTP Parameters* to configure BACdoor for a point-to-point communications link over either a modem or a direct serial connection.

**Dialed** When checked, BACdoor establishes a PTP link using an installed modem which can then dial a modem connected to another half-router. If more than one modem is installed, the Windows *Dialer* dialog will open.

**Non-Dialed Com Port(reset)** When *Dialed* is not checked, BACdoor establishes a PTP communication link over the designated serial port. The parameters in *Non-Dialed Com* must match the parameters of the serial port in the remote half-router.

**Illustration F-6 PTP non-dialed configuration string**

**SNET** Assigns a BACnet network number. This number must be unique among all network numbers on the BACnet internetwork.

**SLEN and SADR** used only by PTP connections. *Source Address Length* (SLEN) is the number of bytes in the field *Source Address* (SADR). SLEN must be non-zero; SADR can be any value that meets the number of bytes specified in SLEN. For connecting to a BAC-5050 set SLEN to 1 and SADR to 00. Connections to half-routers from other vendors may require different values.

**Parameters for Connecting to Routers**

Use these parameters to configure communications between BACdoor and a half-router such as the modem port on a BAC-5050.

**Time to wait for Connection** Sets the time in seconds before BACdoor assumes that a point-to-point connection could not be established.

**Time to wait for Disconnect** The time in seconds before BACdoor assumes that a half-router has disconnected from another, since half-routers give no confirmation of disconnection.

**Default T active** The time in seconds that BACdoor will maintain a point-to-point connection without traffic. If Default T active is set to 0, the connection will be maintained indefinitely.

**Installing drivers for BACnet 8802-3 (Ehternet)**

Following installation of the BACDOC components, the BACnet MAC-layer protocol component must be installed and bound to one network adapter. Following the procedure in this section results in the installation of BACMAC2K.SYS.



BACDOC will not function on computers with 64-bit operating systems. For computers running a 64-bit Windows operating system, use only MS/TP or an Internet protocol (IP) to connect BACstage to an internetwork.

### Before you install

- ◆ Install one or two network adapters (such as an NE5500) through the standard Windows installation procedure. This installation is outside the scope of this topic. Refer to the installation procedures supplied with the adaptor.
- ◆ Make sure you have installed the BACDOC components for Windows 2000. These components are typically installed with BACstage.

### Installing the driver

To install the BACnet 8802-3 driver, do the following:

1. Do one of the following
  - Choose **Start, Settings, Network**, and then **Dial-up Connections**.
  - Choose **My Computer, Control Panel, Network**, and then **Dial-up Connections**.
2. Double-click **Local Area Connection** for your Ethernet/ARCNET Adapter.
3. In the Local Area Connection Status dialog click **Properties**.
4. In the Local Area Connection Properties dialog click **Install**.
5. In the Select Network Component Type dialog select **Protocol** and click **Add**.
6. In the Select Network Protocol dialog select **Manufacturer: PolarSoft Inc**. You should see Network Protocol: BACMAC2K BACnet MAC Layer Protocol. Click **OK**.
7. In the Files Needed dialog click **Browse** and then navigate to the folder where the BACDOC Client was installed and select the subfolder Drivers\bacmac2k. Select **bacmac2k.sys** and click **Open**.
8. In the Files Needed dialog click **OK**.
9. You should be returned to the Local Area Connection Properties dialog. BACMAC2K BACnet MAC Layer Protocol should have been added to your list of protocols.
10. For Windows XP only: Clear the QoS packet scheduler check box as this will interfere with BACMAC2K transmissions.
11. Click **Close**.
12. Close the Local Area Connection Status and Network and Dial-up Connections dialog boxes.

### Removing the driver

To remove the BACnet 8802-3 driver, do the following:

1. Do one of the following:
  - Choose **Start, Settings, Network**, and then **Dial-up Connections**.
  - Choose **My Computer, Control Panel, Network**, and then **Dial-up Connections**.
2. Double-click **Local Area Connection** for your Ethernet/ARCNET Adapter.
3. In the Local Area Connection Status dialog click **Properties**.
4. In the Local Area Connection Properties dialog select BACMAC2K BACnet MAC Layer Protocol and then click **Uninstall**.
5. Close the Local Area Connection Properties dialog, Local Area Connection Status dialog, and Network and Dial-up Connections dialog.



## Appendix G: **Glossary**

This glossary is a list of the more common terms you may encounter when designing and installing a BACnet system.

---

### **alarms**

Audible or visual messages indicating a value is out of range or an abnormal condition is present.

### **analog**

Analog describes any fluctuating, evolving, or continually changing process. Examples of analog units are temperatures, setpoints, percent, volts and amperes.

### **anchor controller**

A BACnet controller that has firmware designated to assist in an automatic process to assign MAC addresses to nomad controllers on an MS/TP network. An anchor controller is always MAC address 0, 1, 2, or 3.

### **APDU**

Application Layer Protocol Data Unit. An APDU is the significant data in a network packet.

### **ASHRAE**

The American Society of Heating, Refrigerating and Air-Conditioning Engineers. Founded in 1894, it is an international organization of 55,000 persons with the mission of advancing heating, ventilation, air conditioning and refrigeration.

### **BACnet**

Building Automation Control Network. A data communications protocol for building automation systems. Developed and maintained by ASHRAE, it is an American National Standards Institute standard (ASHRAE/ANSI 135-1995). BACnet defines how information is packaged for transportation between building automation system (BAS) vendors.

### **BACnet broadcast**

A message that is intended to be received by a group of devices on an internetwork. There are three types of BACnet broadcasts:

- ◆ Global Broadcast - All devices in the BACnet internetwork get the message.
- ◆ Remote Broadcast - All the devices in a remote network get the message.
- ◆ Local Broadcast - All the devices in the local network get the message.

**BACnet broadcast management device (BBMD)**

A special type of routing device that is used in BACnet/IP networks to distribute broadcast messages across multiple IP subnetworks.

**BACnet device**

Any device, real or virtual, that supports digital communication using the BACnet protocol. Examples of devices are operator terminals, routers, unitary controllers, etc.

**baud**

Pronounced *bawd*, it is commonly a reference to the speed at which a modem or other serial device can transmit data. In KMC networks the speed at which a Tier 2 or BACnet MS/TP network operates is referenced in baud.

The term is named after J.M.E. Baudot, the inventor of the Baudot telegraph code.

**conformance class**

Conformance classes describe the capabilities of a BACnet device for communicating data and interoperating with other BACnet devices. A device's protocol implementation conformance statement (PICS) details its conformance class.

**BACnet device**

Any device, real or virtual, that supports digital communication using the BACnet protocol. Examples of devices are operator terminals, routers, unitary controllers, etc.

**Control Basic**

A program embedded in KMC controllers that interprets a set of instructions. Control Basic programs are either written by the installer or supplied the manufacturer with the controller.

**device instance**

A number that uniquely identifies the device on the internetwork. The device instance number is determined by the BACnet system designer. Valid instance number's range from 0 to 4,194,303 and are assigned to the controller



during configuration with BACstage. It is by reference to the device instance number that data is exchanged between BACnet devices.

**directly connected network**

A BACnet network that is accessible from a router without messages being relayed through an intervening router. A PTP connection is to a directly connected network if the PTP connection is currently active and no intervening router is used.

**end of line**

A set of switches or jumpers that indicates the controller is the last physical panel at the end of a network cable. End of line switches are found on controllers connected together on an MS/TP network.

**enthalpy**

Enthalpy is a measure of the heat content within a given sample of air and is expressed in BTUs per pound or as joules per kilogram of dry air. It is used to determine the amount of outside air to add for best economy.

**Ethernet**

Ethernet is a widely-installed *local area network* (LAN) technology specified by the IEEE standard, IEEE 802.3. Original versions of the Ethernet LAN used coaxial cables and were referred to as thicknet or thinnet. Newer versions (10baseT and 100baseT) connect with unshielded twisted pairs of wires in a cable. Ethernet operates also on fiber optics and as a wireless LAN.

**firewall**

A security mechanism, or combination of mechanisms, designed to prevent unauthorized or unwanted communications between sections of a computer network. Firewalls are usually both software and hardware based.

**frame**

See *packet* on page 212.

**gateway**

A device that connects two or more different communication protocols so that information can be passed from devices on one network to the other. Gateways are similar to human language translators. A BACnet gateway uses BACnet as a common language on one side and some non-BACnet (usually proprietary) communication scheme on the other side.

**half-router**

In BACnet, a device that can participate as one partner in a point-to-point (PTP) connection. Two half-routers form an active PTP connection and act as a single router. See the topic [point-to-point on page 213](#) for details about using half-routers to link BACnet networks.

**hub**

A common connection point for nodes on a network. Hubs connect segments of a LAN and contain multiple ports. When a packet (message) arrives at one port, it is copied to all other ports.

**I-Am service**

The I-Am service is used to respond to Who-Is service requests. However, the I-Am service request may be issued at any time. It does not need to be preceded by the receipt of a Who-Is service request. A device may be programmed to broadcast an I-Am service request when it powers up. The network address is derived either from the MAC address associated with the I-Am service request, if the device issuing the request is on the local network, or from the NPCI if the device is on a remote network. See [Who-Is service on page 215](#).

**instance**

See [device instance on page 208](#).

**internetwork**

A BACnet internetwork can be as simple as a single network but is usually two or more BACnet networks connected by routers. The BACnet protocol permits up to 65,534 interconnected networks in an internetwork. Internetworks may contain similar or dissimilar physical types such as an Ethernet LAN and several MS/TP LANs or several MS/TP LANs. A BACnet internetwork permits only one message path between any two devices.

**IP address**

Short for Internet Protocol address, it is the address of a computer or other network device on a network using the IP protocol. The number *10.1.1.2* is an example of a typical IP address. The IP address is usually assigned by the network administrator.

**local area network**

A collection of interconnected equipment that can share data, applications, and resources. It may include computers, printers, data storage devices and industrial controllers and machines. A LAN device can send and receive signals from all other devices in the network. Networks use protocols, or

rules, to exchange information through a single shared connection. These protocols prevent collisions of data caused by simultaneous transmission between two or more computers.

The physical connection between LAN devices can be a coaxial cable, pairs of copper wires, or optical fiber. Wireless connections also can be made using infrared or radio-frequency transmissions.

### **MAC address**

The MAC address uniquely identifies a device on its network. Each network type—Ethernet 8802-3, IP or MS/TP—has its own MAC addressing scheme.

### **master and slave devices**

MS/TP devices come in two varieties:

- ◆ Slave devices are suited for the lowest-cost implementations but they lack the capability to initiate requests; they can only reply to messages from other devices.
- ◆ Master devices are able to initiate requests, but they must also be able to negotiate for a time slot in which to make their requests. This adds some processing and memory requirements to the Master device which can result in higher cost than the slave.

### **MS/TP**

The MS/TP (master slave/token passing) protocol is unique to BACnet and is implemented using the EIA-485 signaling standard. This is a shielded twisted-pair LAN operating at speeds from 9600 baud to 76,800 baud.

### **network segment**

A logical or physical subdivision of network.

**Ethernet segments** In Ethernet, bridges, hubs, switches, and repeaters can couple multiple physical network segments into one logical network segment.

**BACnet network segments** An electrically separate section of a network.

### **nomad controller**

A BACnet controller from KMC Controls that does not have a permanent MAC assignment. Through an automatic process, a nomad controller acquires a MAC address from an anchor controller on the same MS/TP network.

**Ethernet** In Ethernet, bridges, hubs, switches, and repeaters can couple multiple physical network segments into one logical network segment.

**BACnet network segment** An electrically separate section of a network.

**network numbers**

A number from 1 to 65,534 that identifies specific BACnet network. It is assigned by the BACnet system designer at the time a router is initialized for network operation.

**node**

A device such as a computer or a controller on a network that is capable of communicating with other network devices.

**object**

Objects are the means by which a BACnet device represents information that can be observed or changed. The object may represent a physical point such as an input or output or a logical grouping of data such as a PID loop, schedule or variable. Objects have a set of properties and a group of functions that can be applied to them.

The BACnet standard defines a standard set of objects, these include analog and binary inputs, outputs, and values; control loops; and schedules.

**packet**

A packet (or frame) is piece of a message transmitted over a packet-switching network. One of the key features of a packet is that it contains the destination address in addition to the data.

**PAD router**

A BACnet IP PAD is a special type of router that connects two or more BACnet network segments that are separated by at least one IP-only router. The PAD router monitors network traffic for BACnet messages addressed to the other subnet and repackages the message that can pass through IP routers, in effect forming a "tunnel" between the two network segments. A companion PAD router unpacks and retransmits the message on the remote BACnet network.

**PID controller**

A Proportional Integral Derivative loop is an algorithm built into each KMC controller that calculates a value between 0 and 100 percent. The output of the loop can then be used to control the position of an actuator. The output value is based on the sensed value and the required setpoint.

**port**

An interface on a computer, either physical or logical, to which you can connect a device. Examples of physical ports are connections for disk drives, display screens, keyboards, networks, etc. Ports may also be logical connection on networks. For example, port 80 is used for HTTP traffic.

**priority array**

BACnet devices use the priority array property to control *Present Value* in certain objects. For KMC Controls BACnet devices this property is part of both analog and binary output and value objects.

**properties**

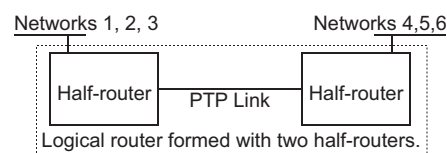
Properties are used to read information from objects or write information to objects. Each BACnet object is provided with a standard set of properties that describes the object and its current status. Certain properties of an object may be required, while others may be optional.

**PICS**

See [Protocol Implementation Conformance Statement on page 213](#)

**point-to-point**

In BACnet, it is a method of data transmission to provide serial communications between two BACnet devices. Typically used for remote and roaming access to BACnet systems, dial-up communications over modems or a portable computer connection to a controller, for example. PTP is based on the EIA-232 direct point-to-point connection or via dial-up telephone modems. The point-to-point link in [BACnet half-routers on page 213](#) connects networks 1, 2 and 3 to networks 4,5 and 6 as if they were on the same LAN.

**Illustration G-1 BACnet half-routers**

BACnet PTP provides for internetworked communications over modems and voice grade phone lines. PTP accommodates modem protocols V.32bis and V.42 and also supports direct cable connections using the EIA-232 signaling standard.

**protocol**

A definition or rules of communication for a computer network. A formal set of conventions governing the format and relative timing of message exchange between two communications terminals.

**Protocol Implementation Conformance Statement**

A Protocol Implementation Conformance Statement (PICS) is a document that specifies exactly which portions of the BACnet Standard a device actually

implements. Manufacturer's issue a PIC statement for each BACnet device.

**repeater**

A network device used to regenerate analog or digital signals distorted by transmission loss. A repeater cannot do the intelligent routing performed by bridges and routers.

**routers, BACnet**

BACnet routers connect different types of BACnet networks. The KMC Controls BAC-5050 BACnet router links BACnet 8802-3, BACnet IP and MS/TP networks. It can also be configured as a *PAD router*.

**services**

BACnet services control the transfer of information between BACnet devices. Examples of services include scheduled commands and alarms between BACnet devices. BACnet defines 26 standard services. Some services read or write properties of objects in the receiving device. Others convey notification of alarms or other special events, others read and write files, and so on. The services provided by a BACnet device are generally described by the device's PIC. statement.

**slave device**

See *master and slave devices* on page 211.

**subnet**

A subdivision of an IP network. Each subnet has its own unique network ID.

**subnet mask**

Short for subnetwork mask, a subnet mask is method of dividing a network of IP addresses into groups. It enables the recipient of IP packets to distinguish the network ID and host ID portions of the IP address. A common example of a subnet mask used is 255.255.255.0. Subnet masks are assigned by the network administrator.

**switch**

A special type of hub that forwards packets to the appropriate port based on the packet's address. Switching hubs improve performance over conventional hubs. A switch may also be referred to as a switching hub.

**TCP/IP**

An abbreviation for *Transmission Control Protocol* and *Internet Protocol*. TCP/IP is two separate protocols that are used together.

- ◆ The Internet Protocol standard defines how packets of information are sent out over networks. IP has a packet-addressing method that lets any computer on the Internet forward a packet to another computer that is a step (or more) closer to the packet's recipient.
- ◆ The Transmission Control Protocol ensures the reliability of data transmission across Internet connected networks. TCP checks packets for errors and submits request for re-transmissions if errors are found; it also will return the multiple packets of a message into a proper, original sequence when the message reaches its destination.

**token**

A special network message that circulates around a token ring network. Only the device that has the token can transmit data on the token ring network. A BACnet MS/TP network is a token passing network.

**tunnel router**

See [PAD router](#) on page 212.

**UDP/IP**

An abbreviation for *User Datagram Protocol* and *Internet Protocol*, a connectionless protocol that, like TCP, runs on top of IP networks. Unlike TCP/IP, UDP/IP provides very few error recovery services, offering instead a direct way to send and receive datagrams over an IP network. It's used primarily for broadcasting messages over a network.

**universal serial bus**

An external bus standard that supports data transfer rates of 12 Mbps. A single USB port can be used to connect up to 127 peripheral devices such as mice, modems and keyboards.

**Who-Is service**

A BACnet device sends The Who-Is service message to determine the device object identifier and network addresses of all devices on the network, or to determine the network address of a specific device whose device object identifier is known, but whose address is not. See [I-Am service](#) on page 210.





# Index

## A

about

- BACstage 175
- Control Basic 97
- objects 91
- properties 91
- services 91

ABS 113

Accept Private Transfers 16

Access menu 13

accumulator

- alarms and events 53
- pulse rate 52
- trend log 53
- working with 51

acknowledging alarms 170

action in PID loops 70

add operator 153

Alarm/Event Summary 37

alarms 207

- acknowledging 170
- custom messages 172
- File location 179
- for accumulator pulse rate 53
- GEST 44
- log 179
- sound file 19
- viewin alarms received by BACstage 170
- viewing in a device 37
- working with 83

ALIAS 114

Align to Grid 159

analog 207

analog value object 59

anchor controller 207

AND 115

animated graphics 165

Annual schedule

- See Calendar object 76

APDU 207

ARCCOS 115

arccosine for KMD controllers 118

archieved indicators 38

ARCSIN 115

arcsine for KMD controllers 144

ARCTAN 115

arctangent for KMD controllers 146

arithmetic operators 106

ASHRAE 207

Auto Addressing 40

Auto Run 68

automatic

- display blanking 30
- sign in to a device 153
- sign off 156
- system connection 19

averaging inputs 51

AVG 116

## B

BAC configuration files 179

BAC files 38, 167

background graphics 165

backup

- all devices 167
- BACnet files 39
- device 38

BACnet 207

- device or controller 208

BACnet Broadcast Management Device 208

Basic

- programs 67
- tables 26

BBMD 208

- settings 17

bias in PID controller 71

binary value objects 62

BIND 116

boolean logic 107

broadcast 207

buttons in Netsensor 30

## C

cables to connect a controller to a computer 9

calendar object 76

calibration

- input objects 46
- Calibration
  - of NetSensor 29
- Cascade 95
- clear
  - Control Basic editor 99
- CLEAR 117
- CLOSE 117
- Close all 95
- close Control Basic editor 99
- Close On Group Link 156
- Closing BACstage 20
- Closing CP-5050 20
- cold start 36
- comma (,) in dialing options 15
- comparison operators 107
- compile 99
- configuring a controller 9
- conformance class 208
- connect
  - to a controller 9
  - to a system 14
  - to device 24
- CONST 117
- constants in Control Basic 117
- consumption programs for accumulators 53
- Control Basic 97, 105, 130
  - BAS file location 180
  - clear programs from editor 99
  - compile 99
  - copy 100
  - editor window 97
  - errors, warnings and information 100
  - expressions 102
  - keywords 113
  - line numbers 97
  - load 99
  - mnemonics 110
  - Next Generation 189
  - open file 99
  - paste 100
  - renumbering line numbers 99
  - save to file 99
  - scans 100
  - select all 100
  - send 99
  - undo 100
  - variables 108
- controlled point in PID loops 70
- copy 100
- COS 118
- COS-1 118
- custom
  - alarm messages 172
  - analog units 156
  - binary units 156
  - input device type 47
  - output device type 55
  - transferring custom units to new systems 156
- cut 100
- D**
- dash (-) in dialing options 15
- date
  - in controllers 44
  - setting date and time 169
- day
  - of month 120
  - of week 120
  - of year 120
- Day of week
  - Programming in NetSensor 32
- DEC 118
- declaring
  - constants with CONST 117
  - variables with LOCALS 130
- delete operator 153
- demand programs for accumulators 53
- derivative in PID loops 71
- device
  - address with static binding 34
  - disable 36
  - enable 36
  - List 24
  - object 42
  - tables 25
- device instance 208
  - in BACstage 16
- device menu 23

- DEW-POINT 119
- DEWPOINT 118
- DEWPOINTS 119
- dial-up 20
- dialing options 15
- DISABLE 119
- disable/enable device 36
- DOM 120
- DOW 120
- DOY 120
- E**
- ELSE 125
- ENABLE 121
- END 121
- end of line 209
- ENDIF 125
- engineering units 177-178
- enthalpy 209
- ENTHALPY 121
- ENTHALPY-SI 122
- ENTHALPYSI 122
- errors in Control Basic 100
- event
  - acknowledgement 78
  - enrollment object 79
  - processing 77
- events 170
- example programs 113
- Exception schedules 74
- Exit 20
- expressions 102
- F**
- file formats
  - alarm log 179
  - AVI 19
  - BAC 38, 167, 179
  - BAS 180
  - group display graphics 158
  - LOG 37
  - MP3 19
  - NG 180
  - WAV 19
- file locations 179
  - alarm log 179
  - BAC configuration files 179
  - Control Basic (BAS and NG) files 180
  - group displays 180
  - trend log 180
  - user sign-on log 180
- filtering inputs 51
- find text in Control Basic 99
- firewall 209
- FLUSH 122
- Flush File Cache 37
- FOR TO NEXT 122
- foreign device 17
- frame 212
- functions 101
- G**
- Gated Event State Transitions 44
- GEST 44
- GOSUB 123
- GOTO 124
- group displays 157
  - animated graphics 165
  - automatically close 156
  - background graphics 165
  - changing a present value 165
  - errors 156
  - file location 180
  - links to other group displays 163
  - scaling animated graphics 163
  - selecting and viewing 157
  - supported graphic file types 158
- H**
- half-router 210
- HALT 124
- help
  - menu 175
- help examples 113
- hierarchy of operators 105
- holiday schedule 76
- HSEL 125
- hub 210
- humidity programming in Netsensor 32

- I**
- IF- THEN 127
- IF THEN 125
- IF THEN ELSE 127
- IF+ THEN 126
- INC 127
- information in Control Basic 100
- input
  - filter 51
  - objects 45
- instance 208
- INT 127
- integral in PID loops 71
- Internet Protocol network settings 17
- internetwork 210
- INTERVAL 128
- INVLN 128
- IP address 210
- ISNAN 128
  
- J**
- job folders 179
  
- K**
- keywords 113
- KMC controllers 181
- KMC Controls
  - technical support 8
  - web site 8
  
- L**
- LAN type 14
- LAN, See Local Area Network 210
- leap year 131
- LET 129
- line numbers 97, 101
- LN 130
- LN-1 129
- Load in Control Basic 99
- local
  - date 44
  - time 44
- local area network 210
- local variables 108
  
- LOCALS 130
- Location of BACDOC.INI file 16
- loop objects 69
- LSEL 130
  
- M**
- MAC address 211
  - assigning to BACstage 17
  - assigning to controllers 43
- manipulated in PID loops 70
- manual change
  - analog value 60
  - binary value 63
  - output 54
- Manual Override Priority permission 153
- master administrator
  - changing simulator mode connection 20
  - in Password Manager 152
- MAX 131
- menus
  - Access 13
  - Device 23
  - Help 175
  - object 41
  - system 151
  - Window 95
- MIN 131
- Minutes Of Inactivity 156
- mnemonics 110
- MOD 131
- MODELNUMBER 132
- modem connection 20
- MONTH 132
- motion sensing 33
- MS/TP 211
  - settings in BACstage 17
- multistate value objects 64
  
- N**
- NAN 132
- NETSENSORSTATUS 133
- NetSensor
  - buttons 30
  - calibration 29
  - changing time 33

- configuring 27
- Day of week 32
- display blanking 30
- displaying
  - humidity 32
  - temperature 31
- motion sensing 33
- programming 31
- verifying status 33
- NETSENSOR-STATUS 133
- Network Backup 167
- Network Restore 168
- network segment 211
- network settings
  - BBMD 208
  - Internet Protocol 17
  - MS/TP 17
- new features 8
- node 212
- nomad controller 211
- NOT 133
- notification class object 77
- Number Of Object Names to Read At A Time 156
- Number Of Objects to Read At A Time 156
- Number Of Properties to Read At A Time 156

**O**

- object 212
- objects
  - about 91
  - accumulator 45
  - analog output 53
  - analog value 59
  - basic tables 26
  - binary input 45
  - binary output 53
  - binary value 62
  - calendar 76
  - device 42
  - device tables 25
  - event enrollment 79
  - input 45
  - list 166
  - loop 69
  - multistate value 64

- notification class 77
  - program 67
  - schedule 72
  - trend 88
- Objects menu 41
- occupancy sensing 33
- off-panel points
  - reading 109
  - writing 109
- ON-ERROR 136
- ON GOSUB 133
- ON GOTO 134
- ONERROR 135
- OPEN 136
- open file in Control Basic 99
- operator
  - adding an operator name 153
  - deleting 153
  - names 153
  - open group display 153
- operators
  - arithmetic 106
  - boolean 107
  - comparison 107
  - precedence 105
  - relational 107
- OR 136
- Our Device Instance 16
- Our Peer Name 16
- OUTPUT-OVERRIDE 137
- OUTPUTOVERRIDE 137

**P**

- packet 212
- PAD router 212
- PANEL-ADDRESS 137
- PANELADDRESS 137
- password
  - assigning 152
  - in simulator mode 20
  - reinitialize device 36
- Password Manager 152
- Paste 100
- period(.) in dialing options 15

- permissions
    - operator 153
    - view/modify 153
  - PI 138
  - PIC statement 213
  - PID Controller 212
  - PID loop See Loop objects 69
  - point-to-point 213
    - settings 17
  - port 212
  - POWER-LOSS 138
  - POWERLOSS 138
  - precedence of operators 105
  - printing help 8
  - priority
    - array 213
    - BACnet standard levels 94
  - Process identifier 79
  - program objects 67
  - programming
    - alarms, See Working with alarms 83
    - an accumulator object 52
    - consumption 52
    - demand 52
    - the NetSensor 31
    - with names 100
  - programming notation 105
  - properties 213
    - about 91
  - protocol 213
  - protocol implementation conformance
    - statement 213
  - PTP 213
    - connecting for 20
  - pulse inputs 48
  - pulse rate 52
- R**
- radians 138
  - random numbers 139
  - Read/Write Property 173
  - reading properties from other devices 109
  - Recipient list 78
  - Reinitialize Device 35
  - relational operators 107
  - REM 138
  - renumber 99
  - repeater 214
  - replace text in Control Basic 99
  - restart 35
    - cold start 36
    - warm start 36
  - restore
    - all devices 168
    - BACnet files 40
    - device 38
  - RETURN 139
  - RLQ 139
  - RND 139
  - router
    - tunnel 215
  - routers 214
- S**
- save
    - to file in Control Basic 99
  - scale 163
  - scans 100
  - SCANS 140
  - SCHED-OFF 141
  - SCHED-ON 142
  - SCHEDOFF 140
  - SCHEDON 141
  - schedules
    - annual, See calendar object 76
    - exception 74
    - holiday 76
    - object 72
    - override 74
    - SCHED-OFF 141
    - SCHEDON 141
    - weekly 74
  - segment 211
  - Select all 100
  - send
    - in Control Basic 99
  - Send Update Notification 36
  - SENSOR-OFF 143
  - SENSOR-ON 144

- SENSOROFF 142
  - SENSORON 143
  - serial number 175
  - services 91, 214
  - setpoint
    - NetSensor button 30
    - programming in NetSensor 32
  - settings
    - Bacstage 155
    - objects to discover 156
    - objects to read 156
    - properties to read 156
    - system 155
    - time and date 169
  - Show
    - Group List in Groups 159
    - Object List in Groups 159
  - Sign Off 19
  - silence mode
    - See Disable/Enable 36
  - simulator
    - mode 20
    - passwords 20
  - SIN 145
  - SIN-1 144
  - slave device 214
  - sorting the device list 25
  - Sound On Alarm 18
  - SQR 145
  - START 145
  - static binding 34
  - STOP 146
  - subnet 214
  - subnet mask 214
  - switch 214
  - switches
    - end of line 209
  - Symbols
    - Dialing options 15
  - synchronize time 169
  - system 14
    - automatic connection 19
    - groups displays 157
    - List 14
      - number 14
      - settings 155
      - System Menu 151
      - system time in NetSensor 33
      - time 169
- T**
- tables
    - Basic 26
    - device 25
    - TBL in Control Basic 146
  - TAN 146
  - TAN-1 146
  - TBL 146
  - telephone number for KMC Controls 8
  - temperature display with NetSensor 31
  - termination
    - end of line 209
  - This Station 202
  - Tile 95
  - time
    - changing from NetSensor 33
    - in controllers 44
    - programming in NetSensor 32
    - setting time and date 169
    - synchronization 156, 169
    - system time master 185
    - timekeeping in systems 185
    - UTC 185
  - TIME 147
  - TIMEOFF 147
  - TIMEON 148
  - token 215
  - toolbar 99
  - trend logs
    - creating with trend object 88
    - file location 180
    - of accumulator inputs 53
    - viewing 90
  - trend object 88
    - buffer size 88
  - tunnel router 215
- U**
- UDP/IP 215

Undo 100

units

    custom analog 156

    custom binary 156

    standard analog units 177

    standard binary units 178

universal serial bus 215

USB 215

Use Local Names 100

Use Object Names 100

## V

value objects

    analog 59

    as variables 108

    binary 62

    multistate 64

variables 108

    declaring with LOCALS 130

vendor ID 16

version

    of BACstage 175

    of firmware in controllers 44

view/modify permissions 153

viewing

    a summary of alarms and events 37

    group displays 157

    the alarm and event file 37

    the alarm and event list 170

    trend logs 90

## W

WAIT 148

warm start 36

warnings in Control Basic 100

web site for KMC Controls 8

weekly schedules 74

Window menu 95

writing

    programs 101

    properties to other devices 109

## X

XOR 149

## Y

YEAR 149