Seeing The Light With BACnet®

By David Fisher, Member ASHRAE

The international standard ISO 16484-5, also known as "BACnet" began as an ASHRAE (then ANSI) standard in 1995. From the beginning, BACnet was intended to embrace all types of systems found in building automation including lighting and lighting applications. BACnet's vision has never been limited to simply integrating lighting with other building automation. The intent, and actual practice, has always been to provide a technology that could serve as the primary means of communication between any type of automation device, and lighting systems certainly figure into that vision.

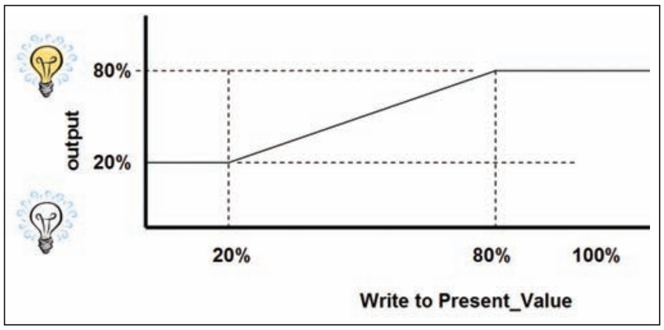
This article describes some features of BACnet for lighting applications that, as of the time this article was written, are not yet part of the standard, but are expected to be published soon.

BACnet provides standardized mechanisms for many types of building automation system (BAS) interoperability based on an *object-oriented model*. Individual automation system devices can perform any type of function, from lighting controllers, to dimmers, to wall stations, as well as HVAC, security, fire and life safety, and access control. Each device that participates in the BACnet network has its own *device instance* that uniquely identifies that device. Devices organize their information into *objects* identified by their *object identifiers*. Some of the most basic lighting applications for BACnet can be easily based on combinations of the nine basic object types: analog, binary and multistate input, output, and value.

When lighting applications get more complex, these object types are more limiting. While a Binary Output object (BO) can be used for simple on/off lighting outputs, there are common lighting applications that require additional features that do not map so obviously onto the BO object's standard properties. For example, a commonly used feature in on/off lighting outputs is called *blink warning*. When the output is turned off, as a result of writing OFF or INACTIVE to

About the Author

David Fisher is president of PolarSoft® Inc. in Pittsburgh. He has been active in the development of the BACnet standard since its inception.



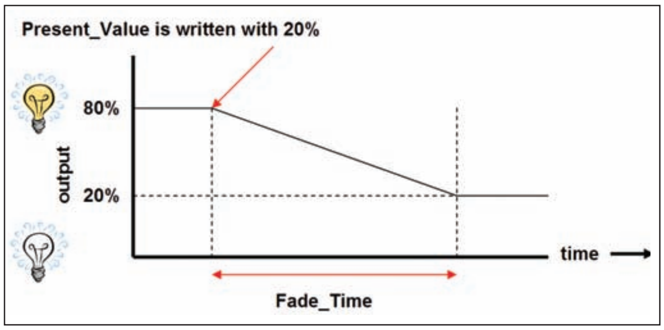


Figure 1 (left): Clamping is the procedure of restricting the lighting level to a range of some specified minimum to some specified maximum as long as the light is on. Figure 2 (right): Fading is the procedure of changing from one level to another over a fixed period of time.

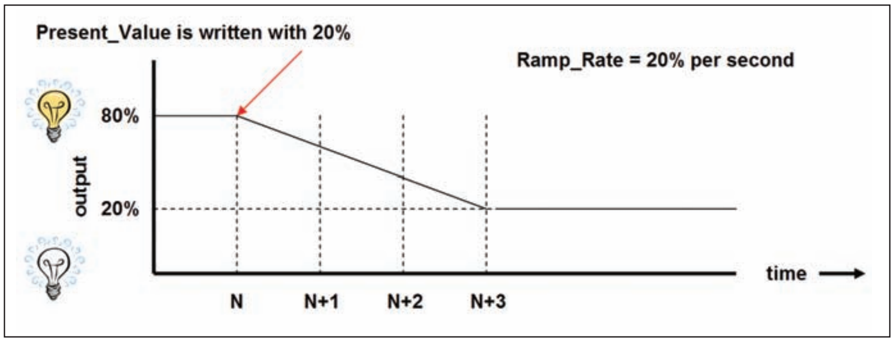


Figure 3: Ramping is the procedure of changing from one level to another at a fixed rate of change of level per second.

the Present_Value property of the output object, blink warning would briefly turn the light off, then back on again and it would remain on for a short period of time. This short warning alerts occupants that the lights are due to turn off soon unless they take some action that extends the scheduled occupancy time. After this brief delay, the lights go off automatically unless this action is taken. To implement this feature, and to make it *configurable* through BACnet, some additional properties are required that specify:

- Whether blink warning should take place at all;
- If so, how long should the output blink off;
- How long should the light remain on again after blinking until automatic turn off; and
- Other specifications.

No standard properties for BO objects exist that represent these lighting-centric ideas. Some systems use additional Binary Value (BV) and Analog Value (AV) objects to convey these parameters, and some systems use nonstandard properties of their BO objects to accomplish the same goal. Obviously, it would be better if there was a standard BACnet object that already had properties to represent these concepts, and that was one of the motivations behind the creation of the Lighting Output (LO object). Dimmer outputs are even more complicated, but in their simplest form can be modeled using Analog Output (AO) or AV objects to represent the dimmer output, usually in units of 0% to 100% intensity. However, there are a number of applications that again require more sophistication.

Although the total range for dimmable lights is 0% to 100%, often there is a need to distinguish between *off (0%)* and some minimum allowable level. This minimum level varies with application and may also vary based on scene presets. Similarly, the maximum level in a particular room under particular conditions may need to be restricted to something less than 100%.

Clamping (*Figure 1*) is the procedure of restricting the lighting level to a range of some specified minimum to some specified maximum as long as the light is on.

In architectural lighting, it is common to require not only control over the lighting level, but also control over the speed at which the level changes from one setting to another, for example, dimming from 100% to 50% over a 10-second period. There are three common types of lighting operations for level-based dimmers:

- Fading (*Figure 2*) is the procedure of changing from one level to another over a fixed period of time.
- Ramping (*Figure 3*) is the procedure of changing from one level to another at a fixed rate of change of level per second.
- Stepping (*Figure 4*) is the procedure of incrementally increasing or decreasing lighting level in fixed steps (for example when a wall panel ▲ or ▼ button is pressed).

These ideas are much more complex than the simple level-oriented model in AO and AV objects. Again, the LO object provides additional properties that allow configuration of these types of behaviors for analog (dimmer) types of lighting outputs.

The LO object (*Figure 5*) packages all of these ideas into one lighting-centric object model and so is better suited to lighting applications that have these requirements. The LO object is similar superficially to an AO in the sense that its Present_Value property represents a 0-100% lighting intensity output. The Present_Value is *commandable*, and has a range of 0% to 100% luminance where:

- 0% = Off;

- 1% = Dimmest; and
- 100% = Brightest.

When two or more controllers attempt to write to the same property of the same object, there is the potential for conflict from a control perspective. BACnet provides a *prioritization mechanism* that allows you to assign relative importance to each potential writer so that the system can decide who is most important when multiple writes occur to the same output. In these cases, the written value includes a priority level from one (most important) to 16 (least important). The object remembers each written value at each priority level and writers are also allowed to relinquish their control of an output, which causes the next most important value to take effect. This kind of output is called *commandable*, which means that it can be changed and the changes can be coordinated among multiple sources.

Because it's commandable, the usual prioritizing behavior can be expected for writes to an LO Present_Value. Some optional properties allow the Present_Value to be clamped within a range when written with a *nonzero* value (*Figure 6*). For example, if the property Min_Pres_Value = 15, and Max_Pres_Value = 80 then when Present_Value is written with 1, the Present_Value and output are clamped to 15% instead. When Present_Value is written with 100, Present_Value and output are clamped to 80%.

Blink warning is intrinsically supported in the LO object. When Present_Value is written with a 0 value, which intuitively means "turn off", the output turns off for Blink_Time (usually brief) if the Blink_Time property has a nonzero time value, indicating that blink warning is desired.

Normally, the output is turned on again after the Blink_Time and remains on for a while before turning off again. This period of time is called Off_Delay (*Figure 7*). In some cases, even though a given output usually wants to support blink warning, it may be desirable to force the output off immediately and ensure that it stays off, in effect trumping the blink warning feature. For example, in a conference room or auditorium when a presentation is being made, we may want to turn the lights off immediately and no warning is required at this moment. For those applications, the priority used when writing to the Present_Value can be configured so that blink warning does not occur.

The concepts of fading, ramping, and stepping are special for two reasons. First, these ideas involve some *operation* that is initiated by writing to a property, but is carried out over a period of time. Second, each operation requires additional parameters to completely specify the desired effect and the parameters are different for each kind of operation.

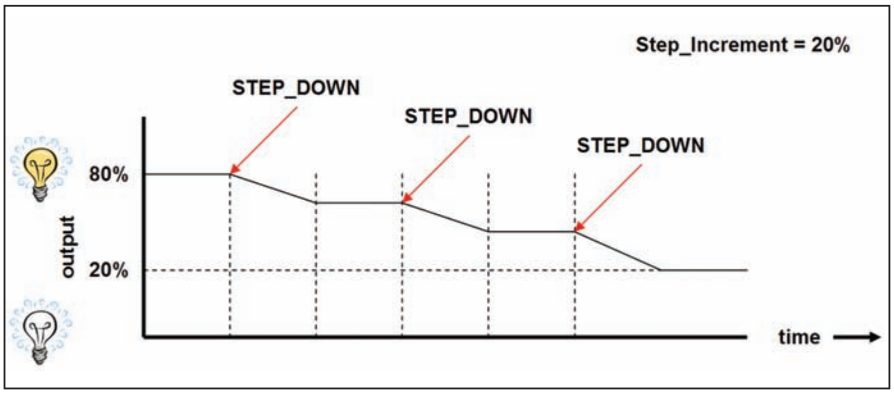


Figure 4: Stepping is the procedure of incrementally increasing or decreasing lighting level in fixed steps (for example when a wall panel ▲ or ▼ button is pressed).

The special property Lighting_Command may be written with a compound value that specifies a unique lighting operation and initiates that operation. When Lighting_Command is written, any in-progress operation is canceled first. Lighting_Command is not commandable and no attempt is made to synchronize multiple writers, so the most recent write wins. However, when lighting operations occur, they affect the Present_Value at the priority level specified in the Lighting_Command_Priority property, and the Present_Value *is* commandable.

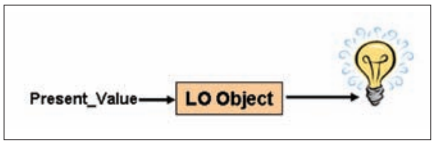


Figure 5: The LO object is similar to an AO representing a 0% to 100% lighting intensity output.

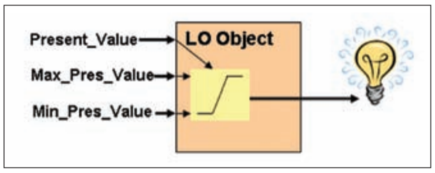


Figure 6: Present_Value can be clamped within a range when written with a nonzero value.

The simplest operation is called STOP. Any previously initiated operation must stop regardless of whether it has completed yet. For example, we could initiate a fade from 100% to 50% over 10 seconds, but four seconds into the execution of the fade we could receive a STOP that would cancel further fading leaving the output at 80%.

The next simple operation is called GOTO_LEVEL, which has a single parameter for the desired target level. This has the same effect as writing the desired target level directly to Present_Value.

There are two operations for fading. FADE_TO specifies a target level to fade to and the period of time over which the fade occurs uses a default period of time specified by the Fade_Time property. FADE_TO_OVER includes the target level as well as the fade time as parameters to the operation.

There are two types of ramping operation. RAMP_TO specifies a target level and the Ramp_Rate property specifies a default rate of change for ramping. When the level is achieved, the ramping stops. RAMP_TO_AT_RATE specifies both the target level and the rate.

There are four types of stepping operation. STEP_UP means to increase the output by an amount specified in the Step_Increment property. This has the same effect as if you wrote to Present_Value with Present_Value + Step_Increment. STEP_DOWN means to decrease the output by an amount specified in the Step_Increment property. This has the same effect as if you wrote to Present_Value with Present_Value – Step_Increment. STEP_UP_BY increases Present_Value and specifies the increment amount as a parameter in the operation. STEP_DOWN_BY decreases Present_Value and specifies the increment amount as a parameter in the operation.

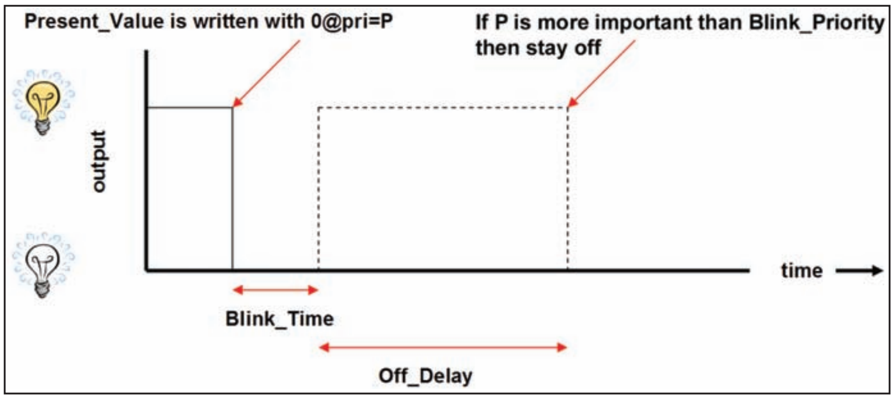


Figure 7: Blink warning timing.

An important lighting command is RELINQUISH. This has the effect of writing a NULL value (relinquish) to the Present_Value at the priority specified by Lighting_Command_Priority, in effect releasing the control of the Present_Value at that priority level.

Often there are multiple dimmer outputs that, while they are separate and discrete outputs that could be individually addressed, under some circumstances want to behave as if they were one logical output (*Figure 8*). In BACnet, this application is modeled as a logical *channel* using a channel object (CH). A channel object has a Present_Value that can feed forward to multiple individual object property destinations. The idea is that any time the CH instance Present_Value property is written, that value is rewritten to a specific collection of other object properties. These objects can live in the same device that contains the CH object, or in other BACnet devices, allowing the channel object to use an intermediary device to manage the forwarding operation. Because of latency issues, CH objects most often restrict their use to redistributing the value to objects in the same device.

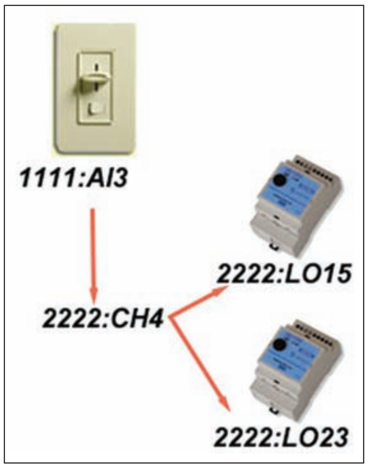


Figure 8: Logical channel using a channel object (CH) feeding forward to multiple LO objects.

Although there are various applications for CH objects, often we want to be able to send a single write request (aimed at a specific channel for example) that multiple devices can receive and act upon at the same time. BACnet lighting applications use the special *WriteGroup* service to send these kinds of requests.

Unlike regular WriteProperty requests that only change a property of an object in one device; the WriteGroup can change properties in objects in multiple devices at the same time. To do this, the WriteGroup is *broadcast* in a way that allows many devices to hear the same request simultaneously. For this to make sense and not cause chaos, all of the listening devices must use the same assignment of channel numbers to mean the same logical thing.

The range of channel numbers is restricted from 1 to 65535. In a large facility with other types of applications besides lighting, sharing the same range of channel numbers could be limiting to some BACnet applications. For that reason, the CH object and WriteGroup have an additional concept called *control groups*. Control Groups allow for the dynamic reassignment of channels to specific groupings, which is useful for *dynamic room configuration* applications such as ballrooms and conference suites.

These concepts allow WriteGroup to provide a compact and efficient representation for transmitting change requests. While not quite as lean as the DMX protocol that is popular with theatrical lighting, WriteGroup can provide high speed transmission of complex lighting commands while making only modest demands on BACnet network bandwidth.

Although the original BACnet standard objects have been used successfully for lighting applications since 1995, these new developments in extending the BACnet standard with some lighting-centric features are a welcome addition.

After a successful vetting through several public reviews, we expect to see products emerging in the marketplace that take advantage of new BACnet lighting features. Lighting applications are highly dynamic and are taking on an increasingly important role in energy savings, green initiatives, and building environment and ergonomics. The Lighting Application Working Group is tackling additional new territory in architectural and theatrical lighting, including color and motion control. With the shift from incandescent to LED-based lighting in the near future, the issue of how to control and interoperate across lighting systems and building automation systems is timely and critical. What better way to help them along than using the international standard BACnet?

Author's Note

This article is an abbreviated version of a white paper published by the SSPC135 Lighting Applications Working Group.●