Overview of Direct Digital Controls

For Building Automation Systems

Part 3 – Outputs

This document is meant to be a continuation of the brief introduction to the ideas behind Direct Digital Control (DDC) of Building Automation Systems (BAS).

In part 2 of this series I discussed controller inputs. Which allow the controller, and its program, to "sense" various operating parameters. The same as your body senses things through touch/feel, smell, taste, hearing, and sight. Feeding your mind information to consider, and perhaps act upon. But for your mind to process those "inputs", and then take any appropriate action, you need "outputs". In the case of your body, your outputs are things like your feet, legs, arms, hands, fingers, and voice. Through which you can take an action; if such is needed.

In the case of DDC controllers as used for BAS control and management, the microprocessor and its program need some means in order to act upon the outside (of it) physical world and cause things to happen. In order to control whatever it is meant to control.

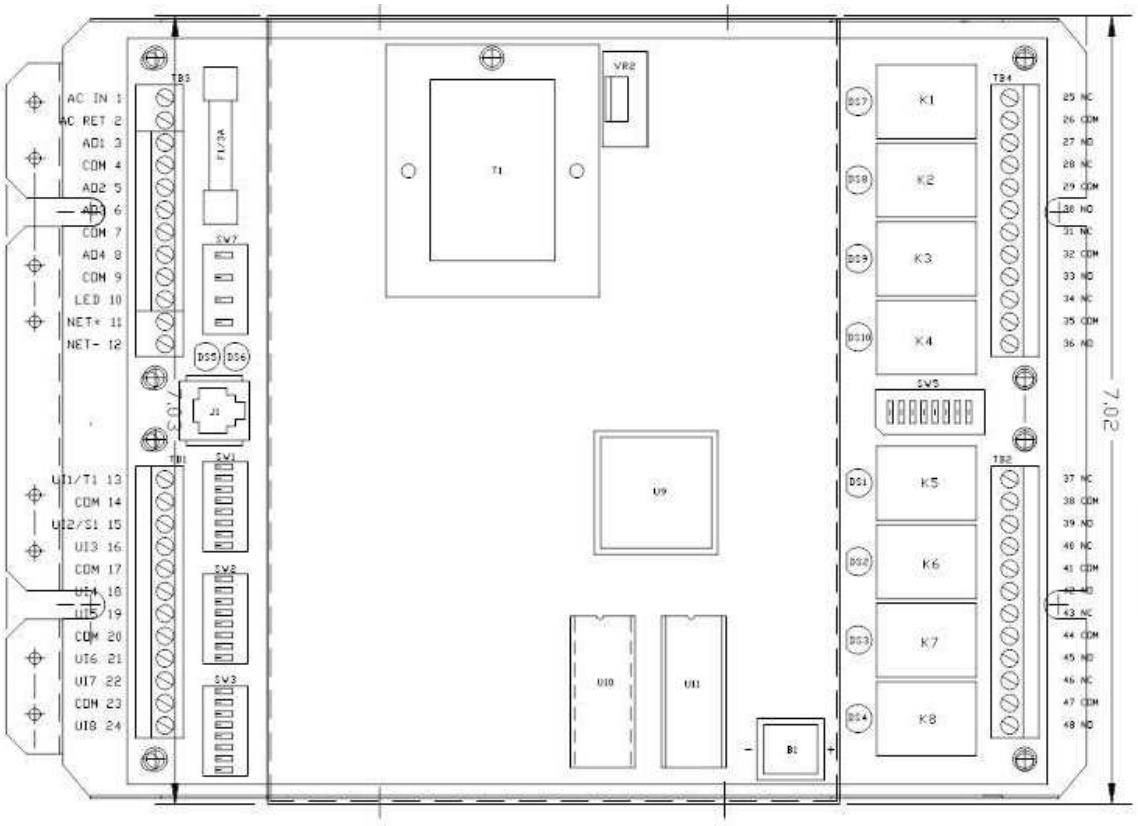
Outputs, for a DDC controller, are generally one of two types; analog or digital.

A Digital Output is similar to a switch that can be turned ON or OFF. With the switch being controlled by the controller and its program. Generally speaking, in the case of the type controllers about which I have been discussing, these digital outputs are limited to handling relatively small electrical current flows. Often, only one amp, or a few, at 24 volts AC. Most often the digital outputs (also called BINARY outputs) of such controllers will be used like pilot relays. That is, a small relay which will in turn energize a larger relay capable of handling the electrical current loading of a motor, an electrical branch circuit, and so forth. Or, a digital output might be used to sound an audible horn, a signal light, or whatever.

An Analog Output, in the case of DDC controllers used for BAS applications, is most often of a type that can produce a variable signal output of a DC voltage or milli-amp current. Typical output ranges are 0 to 5 volts DC, 0 to 10 volts DC, 0 to 20 milli-amps, or 4 to 20 milli-amps. Other ranges and types of analog outputs are possible and available, but the specific ones I gave are considered pretty standard in the industry. Analog outputs are typically used to provide a positioning or speed control signal. But may be used for other purposes.

OUTPUTS

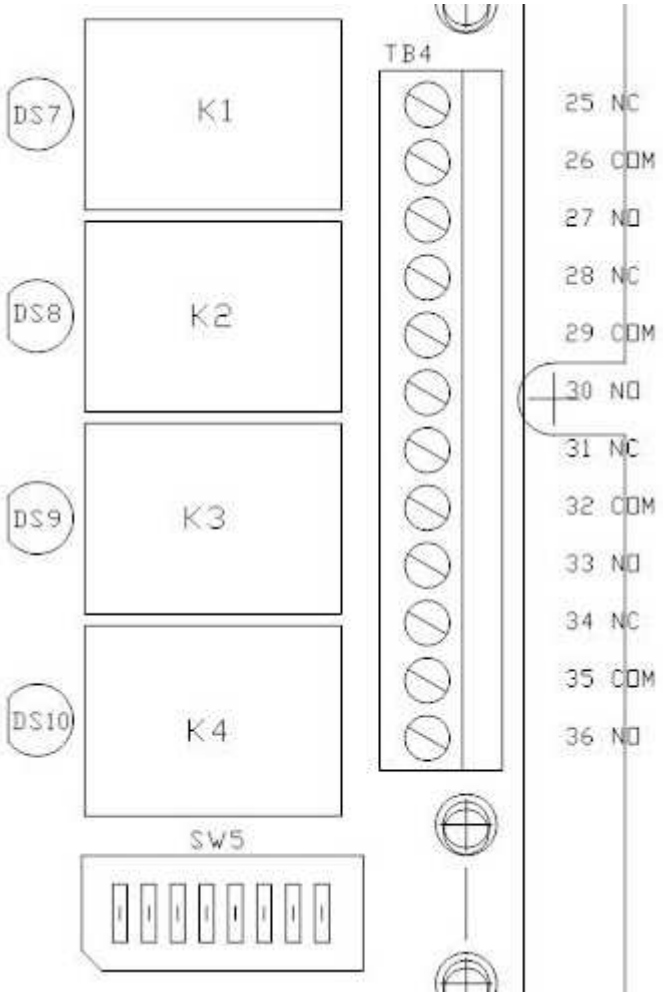
Now let's take another look at that CAD drawing of the general purpose controller which I showed in Part 2 of this series of articles.



On the top, left hand side of this controller is a terminal board which represents the physical connections for the 4 analog outputs this particular controller has. They're marked AO1 through AO4. On the right hand side, top to bottom, are shown 8 single pole, double throw relays; marked K1 through K8. Which are the digital (or binary) outputs of this particular controller. To the right of them are two terminal boards which are the connection points for wiring the contacts of these relays to something you might wish to turn off or on.

Keep in mind, I am only showing this controller as a generic example. Exact configurations, components used, placement, and numbers of analog and digital outputs (or inputs, for that matter) vary considerably from model to model and manufacturer to manufacturer. For instance, from the same manufacturer that produces the general purpose controller that is shown above, you could buy similar controllers with either more, or fewer, inputs and outputs.

Now let's take a closer look at those digital outputs ….



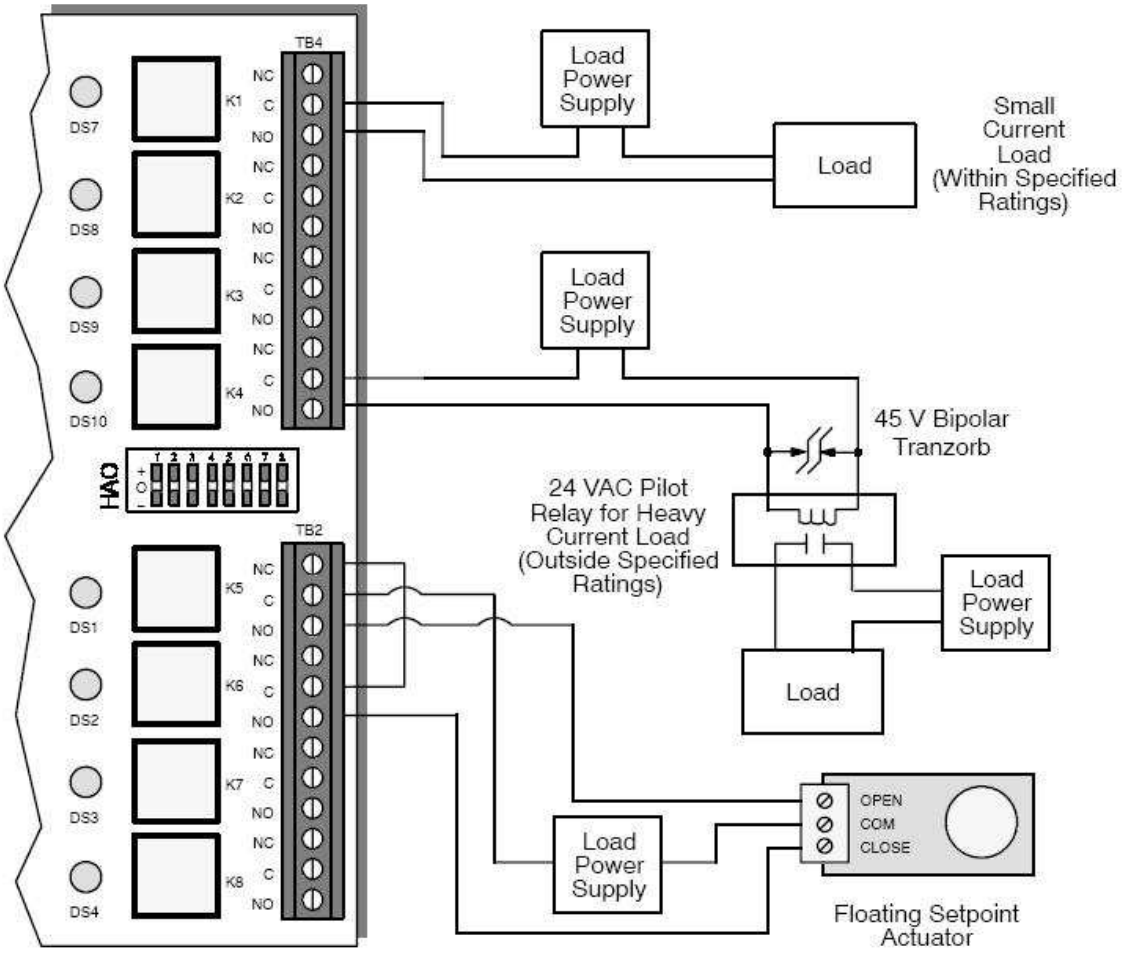
I'm only going to show digital outputs 1 through 4 here. Arrangements for outputs 5 through 8 are the same, with only the markings of terminals and components being different.

K1, for instance, is a single pole, double throw relay. The terminal marked 26 would be the common pole connection for this relay. Terminal 25 would be the NC (normally closed) contact. While terminal 27 would be the NO (normally open) contact. The rest of the connections follow a similar pattern.

DS7 is a LED which lights up whenever K1 is energized.

SW5 is a DIP switch block, with 8 dip switches. This is the HAO control. That's Hand – Automatic – Off. If the dip switch on the farthest left is moved to the down position, relay K1 will energize. Regardless of what the program logic is doing, or trying to do. If the same dip switch is positioned all the way to the top, relay K1 will de-energize. Again, despite whatever the program logic is trying to do. In the middle position, the energizing or de-energizing of relay K1 is under the control of the program running within the controller. Maybe … via software and direct electronic communications with the controller you can also over-ride program control and command the relay on or off.

These relays are rated by the controller manufacturer at 1/8 HP @ 120 volts AC, 10 amps with a resistive only load @ 120 volts AC, or 2.0 amps @ 240 volts AC general purpose loading. So they're beefier than what is found on many other controllers, but you may still have to use them as pilot relays to energize or de-energize heavy duty contactors for larger electrical loads. The relays are designed to last at least 10 million cycles.

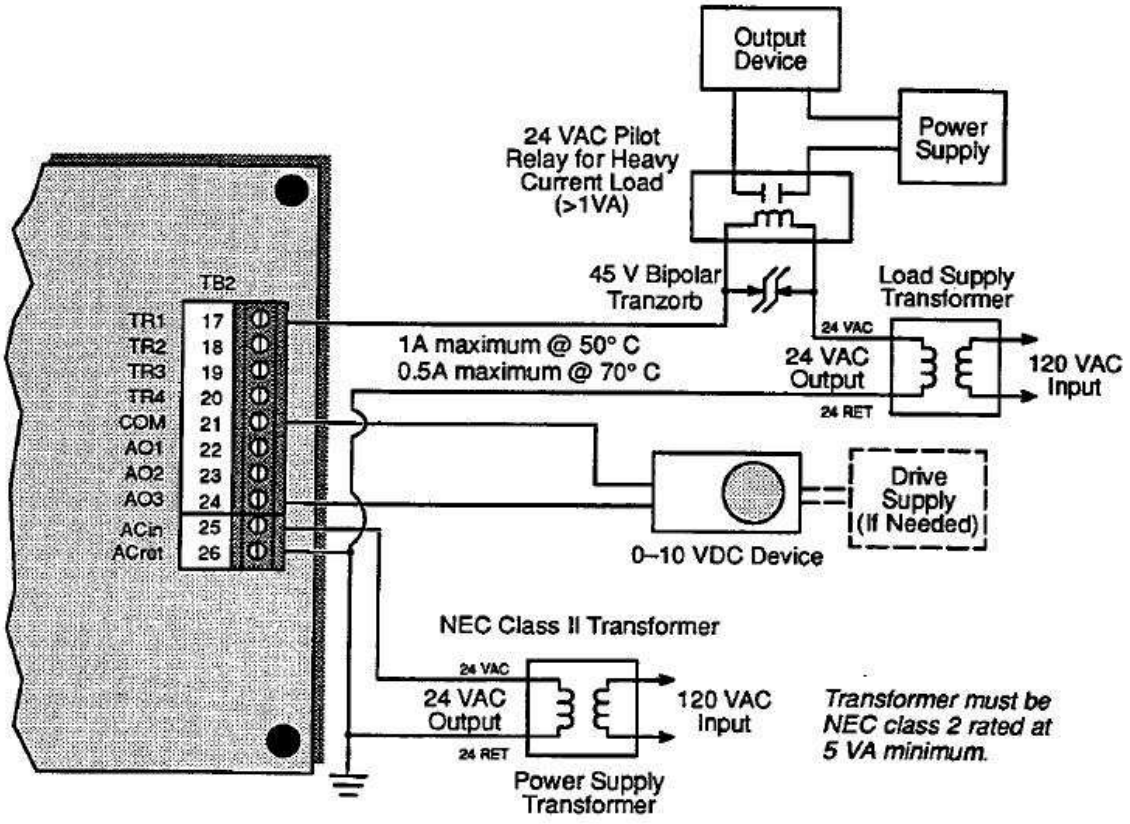


Here, we can see some typical wiring arrangements for wiring the outputs of this particular controller. At the top you can see an example of the K1 relay being wired so as to energize or de-energize a small electrical load.

The K4 relay, however, is wired to energize or de-energize a load that is greater than the contacts of K4 are rated to handle. So, in this case, K4 is wired to energize or de-energize a heavy duty contactor that is rated as being adequate to handle the electrical loading necessary for whatever is being turned on or off. The Tranzorb shown is a semiconductor device, wired in parallel with the connections of the coil of that heavy duty contactor. Which is meant to "absorb" what could be a rather large voltage spike created when the magnet field of the heavy duty contactor's coil collapses. Look up tranzorbs and their uses if you're not familiar with the issue.

In the above example, relays K5 and K6 are both wired to a floating point type motor/actuator. More about this will be discussed later.

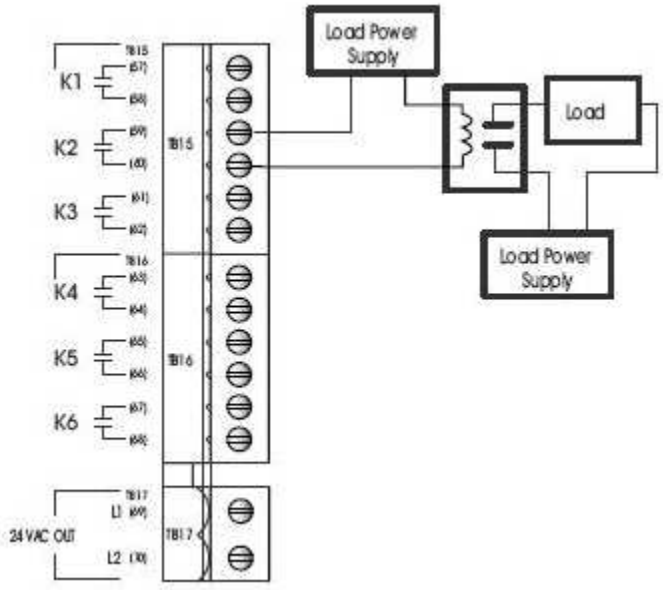
Many DDC controllers have actual electrical-mechanical relays as digital outputs. But others use a different technology. Commonly, triacs are used as digital output devices. A triac is a semiconductor device, simply think of one as a solid state switch which can be electronically commanded to conduct, or not conduct, an electrical current flow. If you really want to know how they work, look it up. Semiconductor devices are an entire field of study, all by themselves. In the case of DDC controllers used for BAS applications, triacs are handy because they're small, relatively cheap, and if not subjected to abuse and excessive voltages or currents can be switched on or off many millions of times before failure. NOTE … I said "if not subjected to abuse and excessive voltages or currents". If you don't wire and use them correctly, within their limits, you can burn a triac out pretty durned quick and easy. And since these are most commonly soldered onto a printed circuit board with lots and lots of very small electronic components on them, any of which can be damaged by an incautious or unskilled use of a soldering iron … you could find yourself having to purchase a new controller, for from $150 to a couple thousand dollars, just because you burned out a 15 cent triac.



Above is a picture of a portion of a different controller. Which has the triac style digital outputs. Marked TR1 through TR4. Notice there is only a single terminal for each digital output. Only one is needed in this case. Here, with this controller, each triac has one side of it soldered directly to the Common electrical reference plane of the controller.

When wiring a device to be energized by such a digital output, the **HOT** wire is what is connected to the triac. In the "Off" condition, the triac does not conduct an electrical current. So, in the case as shown above, even though the coil of the relay that we want to be able to energize or de-energize does have 24 VAC applied to it, there is no complete path for the flow of electricity, when the triac is Off. When the triac is switch On, there would now be a path back to Common, electrical current would flow, and the relay would energize.

Another example ….



Here is a picture of a portion of yet another controller. In this case, this particular controller uses triacs, but each side of each triac has been soldered to individual terminals. And the triacs operate just as would be the case if they were NO contacts on a standard relay.

Typically, the triacs used by most of the manufacturers of DDC controllers used for BAS applications with which I am familiar, have been rated for 24 volts AC or DC, and a maximum of 1 ampere current flow.

Digital outputs aren't used solely for simply switching something Off or On.

In many cases digital outputs are used for the purposes of providing an "Enable" or "Permission to Operate" signal to some separate control system, that operates and controls an item, or several items that are otherwise not under the direct control of the DDC controller itself. For instance, a digital output of a DDC controller might simply provide an "Enable" signal to a separate boiler control system. Which, once it has permission to operate, continues on its own independent of that original controller. Until that "Enable" signal is removed. This is a pretty common practice. The thinking being that the designers of the boiler control system are specialists in that field, have received

approval for their design and the components used from relevant regulating agencies, have already gotten final approval and certification of that actual installation by the local AHJ (Authority Having Jurisdiction), have the appropriate amount of money set aside to satisfy any bonding and insurance obligations to cover their responsibilities in the case of catastrophic failure/accident, and … hey, it's working just fine. All the customer wants is for his BAS system to be able to Enable, or Disable the operation of the boiler(s) according to some schedule, or schedules, he wants to set up. And/or IAW some other criteria such as the current outdoor temperature.

For instance, maybe the building owner wants his boilers, used for heating the building, to be disabled for operation in April. Normally. That's a scheduling action, easily set up within a DDC controller. BUT … the owner also wants the BAS to monitor outdoor air conditions and IF the outdoor temperature drops below 35 degrees, give the boilers permission to run, even if it is April. BUT … he wants his BAS to also monitor individual room temperatures and if any one of them drops to 60 degrees during the normal working hours of his company, turn the durned boilers on, even if it is May. More … even if it is after normal working hours, or a holiday, if any of those rooms drops in temperature to 55 degrees. Turn the boilers on, and the air handlers or whatever else, and let them operate until no room temperature is below 65 degrees. Regardless of the time of day or time of year. Then shut them down again.

All of which is easily accomplished with most any BAS system.

Get the idea? In the given example, owner has whoever designs and installs his BAS system keep his boilers controls, as is, but he's still able to exercise a flexible control over when they'll actually operate.

In some installations I've done, that's all the owner wants. We install a bunch of "Enable" and "Disable" points on boilers, hot water heaters, pumps, air handlers, furnaces, lighting, and whatever. Then have the points enable or disable according to such things as schedules, occupancy status as sensed by occupancy sensors, the amount of daylight outside, some established low or high limit temperature for any one room (or any 3 rooms, or any 5% of rooms, or whatever), or whatever criteria the customer wants. But once an Enable signal is sent, whichever equipment then operates off its own control system.

This is common in situations where a facility owner/operator wants a BAS system which can give him/her some control over the operation of the building's mechanical and electrical systems, and which can save them energy (and money) by not operating things which are not currently needed … AND where facility owner/operator wants to keep the cost of the BAS installation as low as possible.

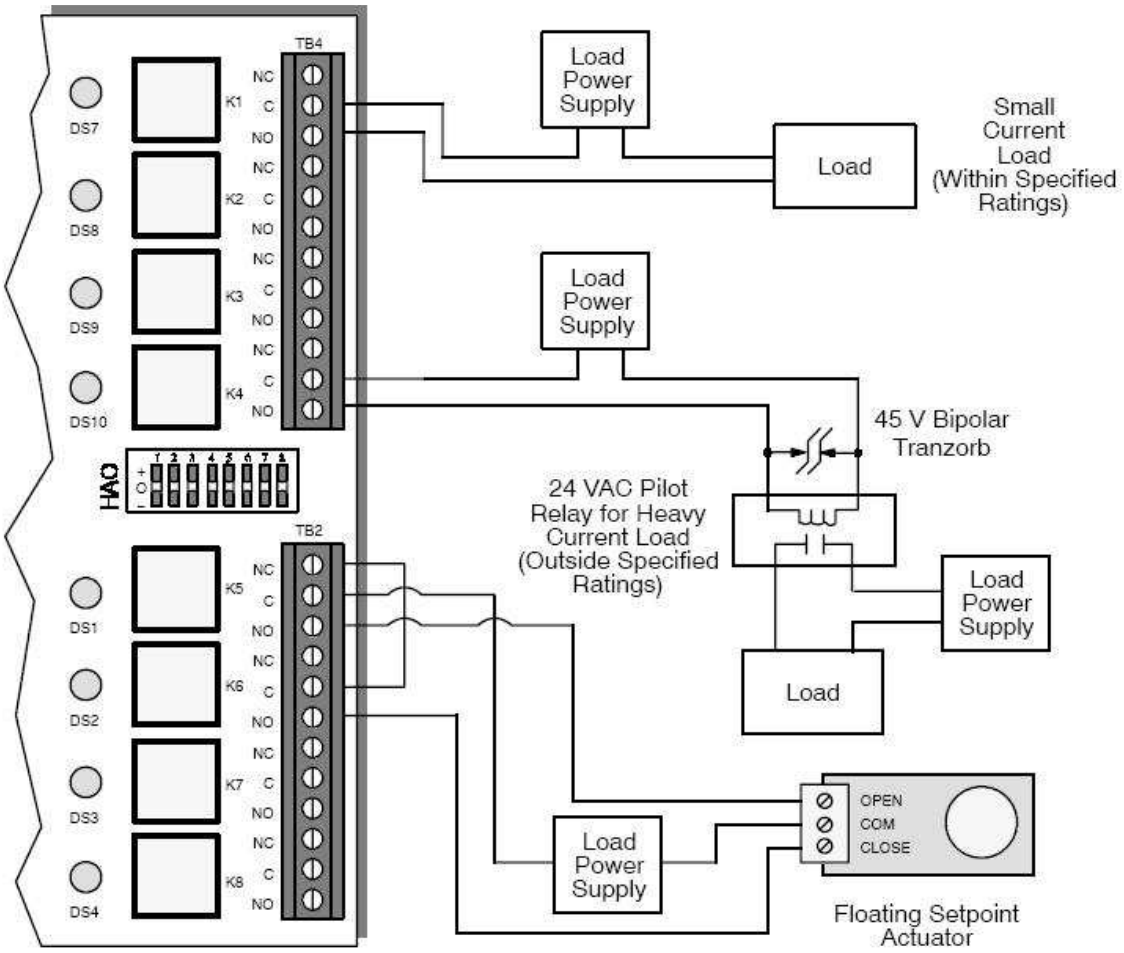
Taking over a more complete and full control of equipment can almost always save even more energy (money), but the payback period (the time it takes for the savings in energy costs to equal or exceed the cost of the control system) is longer. And if it's an older

building, or a smaller one, it simply may not be worthwhile financially to implement a more complete control system.

One needs to always remember that just because something CAN be done, this does not also mean that it should, or that it is worthwhile in terms of energy or money saved.

Digital outputs may also be used to position some motor or actuator at some intermediate position, neither fully open nor fully closed for instance. In other words; a proportional, variable, or stepped control.

How? Let's take another look now at that thing I previously skipped over … the floating point motor.



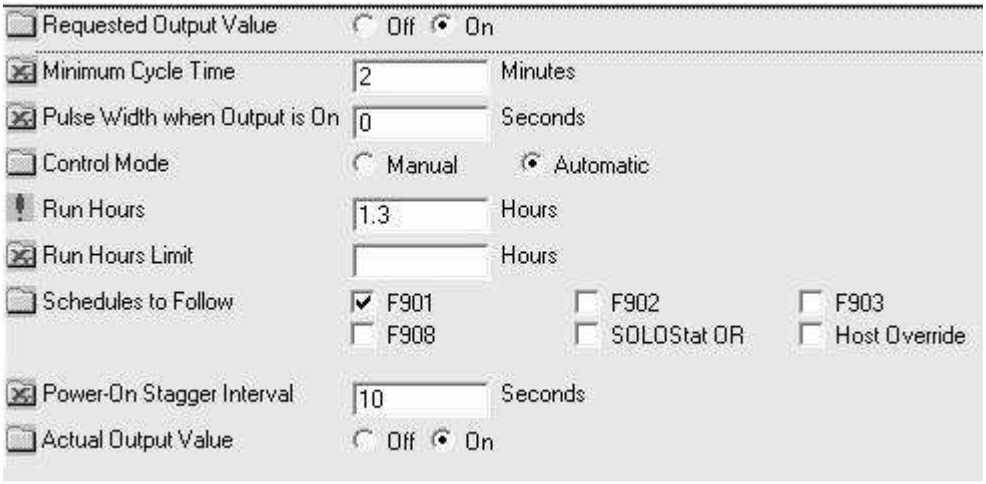
Notice above how that "Floating Setpoint Actuator" is wired. (Floating Setpoint Actuator is just another name for Floating Point Motor. Which may also be called a Tri-state Motor.)

If relay K5 on the controller energizes then the motor (actuator) will start to rotate in the Open direction. To open a valve, for instance. If K5 were left energized long enough,

we could presume the valve would open up to 100%. But, suppose we didn't leave K5 energized that long? Let's consider a floating point motor and valve made by Belimo which is listed as having a 90 second total travel time, in order to move from the fully closed position to the fully open position. So assume that we're starting with the valve fully closed. What happens if we energize K5 for only 45 seconds, and then de-energize that relay? The valve will have only moved to 50% of fully open, and will stay right were it is until we do something else. If, at some point, we then energize K6 for 22.5 seconds, the valve will close down until it's only 25% open.

Get the idea? With floating point motor control we can use two digital outputs to position that valve most anywhere we want it, not simply fully open or fully closed. The same concept holds for floating point actuators that might be used on dampers or other devices.
I'll give more information on floating point motor control a little later.

As when I discussed inputs on controllers, there is more to it than simply wiring up something to a controller's output. Let's take a look at an example.



Here, once again I've made a screen shot of a display while I'm using a manufacturer's software for configuring a DDC controller. In this case I'm looking at a tab for Digital Output #1.

As can be seen, I've set a minimum cycle time for the output at 2 minutes. What this means is that the output will energize for a minimum of two minutes, even if the command changes and tells it to turn off before that amount of time elapses. Likewise, if the output is commanded off, once it goes off it will not again energize until a minimum of two minutes has elapsed. Regardless of the "Requested" value or command. This is common practice and is utilized whenever and wherever needed to prevent what is called "short cycling". If this digital output were used to start a refrigerant compressor or a sizeable electrical motor, turning either on and off again too quickly could wreck them, or blow the circuit breakers, or both.

I left Pulse Width as zero, as I'm not using pulse width modulation in this example. And won't be discussing it in this overview.

Control mode I set to Automatic. In this case, since I have no other control loops enabled, the controller knows it is simply to follow whatever schedules I set up in the checked schedule block. I could check, and then fill in, multiple schedules if I needed to do so to have different "Run" times on different days and at different times of day.

The Run Hours and Run Hours Limit block are similar to the case I discussed back in the section about Inputs. Run hours simply keeps a tally of the total number of hours the output has been energized. Cumulative. And Run Hours Limit tells the controller to send out an alarm (notification) if that many run hours has been exceeded.
Power On Stagger Interval is set to 10 seconds. What this does is that if power is interrupted to the controller, and is then restored, the controller will recommence doing its thing, operating whatever it has been connected to. But, when restarting (re-energizing) each of its digital outputs, it will wait N seconds (whatever number is in the appropriate block) before energizing each output. This way, you don't get a power loss and then a restoration, and have the controller try to turn everything on all at the same time. Which could create one heck of a new power surge, and result in yet another power failure if that controller were turning on, all simultaneously, a lot of very large electrical motors.

So, in my simple example, I've shown how to wire something up to a digital output of a certain make and model of controller, and have even configured a basic type of "control loop" which would control something. Since the digital output will follow a schedule.

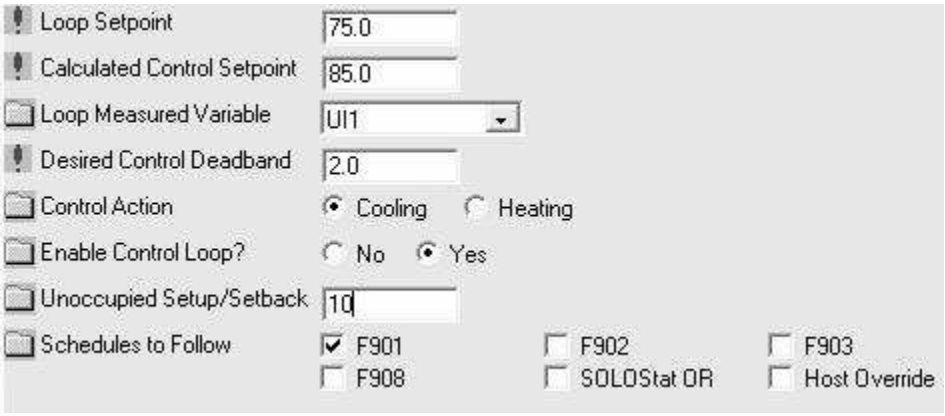
Above is a screen shot of a schedule screen. In this case, suppose that it's Schedule #1. In my previous example setup, the digital output would energize at any time this schedule was "Occupied". As can be seen, it will go "Unoccupied" at 1700 hrs (5 p.m.) and stay that way until 7 a.m. the next day. IF … it's a weekday. For this particular controller, there are 8 schedules available, built-in to its firmware. I could use schedule # 2 and set up things so that schedule #2 would have an "Occupied" period from 9 a.m. until 3 p.m. on Saturdays and Sundays. As an example. Then go back to the digital output screen and click to enable it to follow schedule 2 as well as schedule 1. Eight schedules aren't flexible enough? That's okay, I can also click to enable the controller to ALSO follow a broadcasted schedule that originates from another controller or from a building supervisory controller. Or, as this controller allows for it, I could write my own scheduling and control program, in a programming language that is similar to Basic,

compile it and load it into the memory of this controller, and have the output, or outputs follow the commands of the program and ignore the internal firmware completely.

Ah, choices … they're a good thing.

But a digital output can be controlled by other strategies, other than simple scheduling. The possibilities are almost limitless, given that you can make a custom program for this controller, and create your own possibilities. But this document is NOT gonna cover the custom programming of DDC controllers.

However, in the example controller I am basing this document on, it has built-in firmware routines, ready to use, IF you so wish, that can serve as examples of typical digital output control.



Above I show a screen shot of the screen for setting up what is called a Thermostatic Control Loop, for digital output #1. Let us suppose that I have the actual output wired to start an exhaust fan. And that I've installed a temperature sensor in a room that the fan serves, then connected the sensor to UI1 of this controller. I've already told the digital output that it is in automatic mode. Set up a schedule, and now I'm going to add more to the control process for this fan. Here I give the control loop a setpoint, a deadband, have told it that this is a cooling sort of action, enabled the control loop, and told it that during unoccupied periods, IAW the schedule indicated, it should SET UP the temperature setpoint by 10 degrees.

During normal working hours, any time the room temperature increases to past 75 degrees, that exhaust fan will start. Drawing away the heated air. Hopefully. As the room cools down to the setpoint, minus the deadband, the fan shuts off.

As can be seen in the sample above, evidently the controller's internal clock is saying that it is after normal working hours, so the CALCULATED setpoint, which is what really controls at what temperature the fan turns on, has changed from 75 to 85 degrees, because the controller knows to add that 10 degrees to the setpoint during unoccupied times.

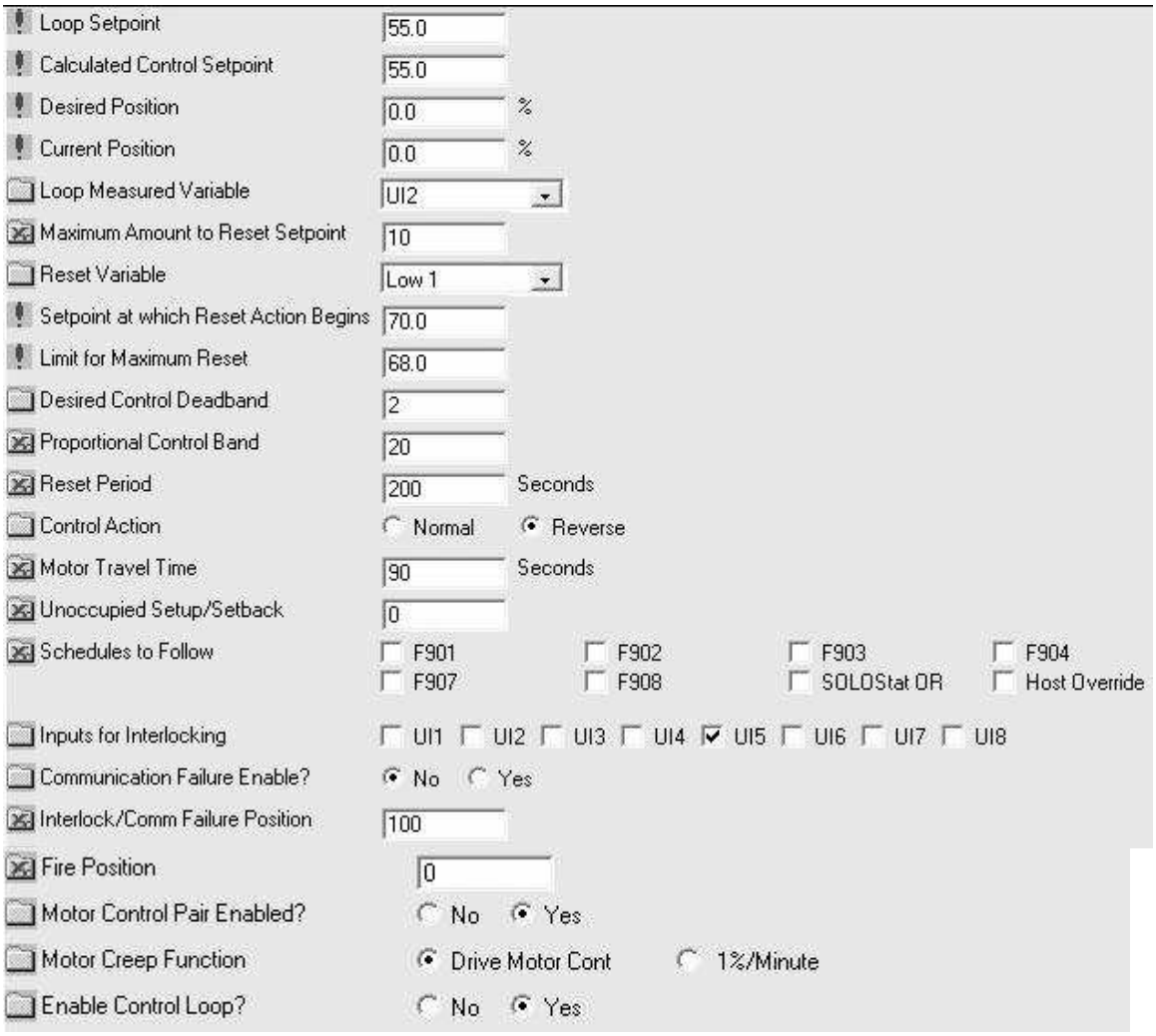
If I'd hooked an electric baseboard heater to the same digital output instead of a fan, and changed the Control Action to "Heating" instead of "Cooling". Then during occupied hours the controller would turn on the heat if the room temperature fell below 75 degrees. But after hours, it wouldn't do so until the temperature dropped below 65 degrees.

As far as selecting Loop Measured Variables goes, I could elect to have this control loop watching any of the controller's inputs. Or could elect to have the measured variable be the Low Value, High Value, or Average Value from a Math Control, that's also built into the firmware of this controller. Then set up the Math loop to watch whichever inputs I desired and produce that Low, High, and Average value. Or I could point this control at some one or other data that would periodically be broadcasted over the control network, and use that for its measured variable. Or …. Or …. Or …..

There are a number of other possibilities, but I'm not trying to teach you to become a controls technician. I am simply trying to introduce the most basic concepts behind what direct digital control of BAS is all about.

FWIW, on that last screen I showed, just because I said that control loop was a kind called a Thermostatic Control Loop, and the Control Action block says "Cooling" and "Heating", don't take that to mean that the only thing you can do with it is to heat or cool something. Just think of "Cooling Action" as meaning energize the digital output if the value of the measured variable RISES above the setpoint. And "Heating Action" means it should energize the output if the value of the measured variable FALLS below the setpoint. The control loop works just as well if you're measuring the amount of carbon monoxide building up in a garage, and want a damper to open and an exhaust fan to start if the gas concentration in the garage exceeds 30 parts per million.

Now, let's look at that floating point motor thing. How can we control accurate positioning of a valve or damper motor, at some needed intermediate position that is between fully open and fully closed, with digital (binary) outputs?



On this screen, I'm working with what's called a motor control loop, by this manufacturer. Specifically, what is meant is "Floating Point Motor" control.

Let us suppose that I have a floating point motor wired to digital outputs 5 and 6 of my controller. And that the above screen is for what would be called Motor Control Loop #3. Loop #3, when enabled, would control digital outputs 5 and 6. As the digital outputs are assigned to motor control loops in set pairs. Let us further imagine that this valve is to control the heating valve of an air handler, which I want to modulate in order to maintain a 55 degree discharge air temperature, and that said discharge air temperature sensor is connected to universal input #2.

As can be seen, I've set the control setpoint to be 55 degrees. Calculated setpoint is also 55 degrees at this time. I don't set that, the controller calculates it. Desired position and current position are both zero. Let's assume discharge air temperature is currently a bit over setpoint. I've got the Measured Variable part of the control looking at the right input.

Now, we see something new. I've entered 10 in the Maximum Reset block. What's that? Look down a bit further and see that I have Reset Variable set to something called LV1. This value can be found in a firmware math loop in this controller. In that math loop I've told it to fetch temperatures from several inputs that represent several rooms served by this air handler, locate the lowest value and place that value in a location called LV1.

Then I set the Reset Action Start at 70 degrees, with a reset limit at 68 degrees.

What does all this do? Well, as the coldest room has its temperature drop to below 70 degrees, the controller will start adding increments to the setpoint, resulting in a new, higher, calculated setpoint. The calculated setpoint is really what controls the valve. By the time the coldest room hits 68 degrees, the calculated setpoint will be 65 degrees, (55 plus the 10 set in Maximum Reset). And the discharge air should go up to 65 degrees in order to help warm the rooms. (The presumption, in this scenario, is that the rooms served have some other heat adding source, and normally simply need cool, fresh air to maintain comfortable temperatures. Perhaps they're work areas with a lot of heat generating equipment. But, under some conditions, perhaps some machines are turned off, 55 degree air from the air handler is just a bit too cool and needs to be boosted.)

In the following blocks I filled in the needed information to implement a PI control scheme (Proportional plus Integral control action)

Moving on, Control Action is set to Reverse, which is the same as "Heating". In Motor Travel Time I've told the controller that the valve takes 90 second to move from fully open to fully closed. Now the controller knows, and can calculate how long it needs to energize a digital output in order to move the valve a desired amount of travel.

I've set to setback or setup. And I've not linked the control to any schedules. So it'll operate the same around the clock.

At Inputs for Interlocking, I've checked input 5. In this imaginary scenario, I've connected a Fan Status sensor to input 5. Then I set fail position to 100%. What this means is that if the air handler supply fans shuts down for any reason, the heating valve will open wide. Doing so should keep the air handler's interior warm, a desirable thing if it's a little nippy outside, ie maybe negative 20 'F, or some other chilly value.

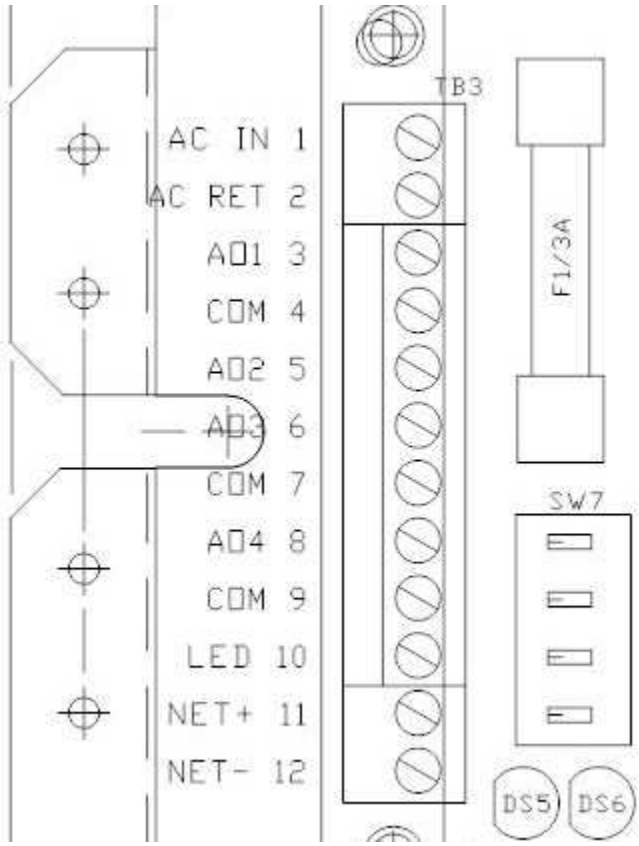
I've enabled the Motor Pair (in this case, linked digital outputs 5 and 6 to this control loop). I set the motor creep function to a value that means, in the case of THIS controller, that whenever the valve's position is 0%, or 100% … or is supposed to be that … the controller will continue to drive it closed or open as appropriate continuously to ensure that it is positively all the way closed, or all the way open. Enabling this function means that periodically any positioning errors caused by the fact that the valve motor does not travel at a rate of EXACTLY 90 seconds for a full travel time, is corrected for.

Lastly, I enable the control loop, and the controller will take over control of the valve.

Easy enough, isn't it?

Types of output devices connected to digital outputs are typically any of a wide, wide variety of relays or contactors, dry contacts on other control systems which represent an "Enable" signal, and floating point type motors.

Now, let's take a look at analog outputs.



Here I am showing the upper left side of the CAD drawing for my example controller. As can be seen, there are 4 analog outputs available. Marked AO1 through AO4.

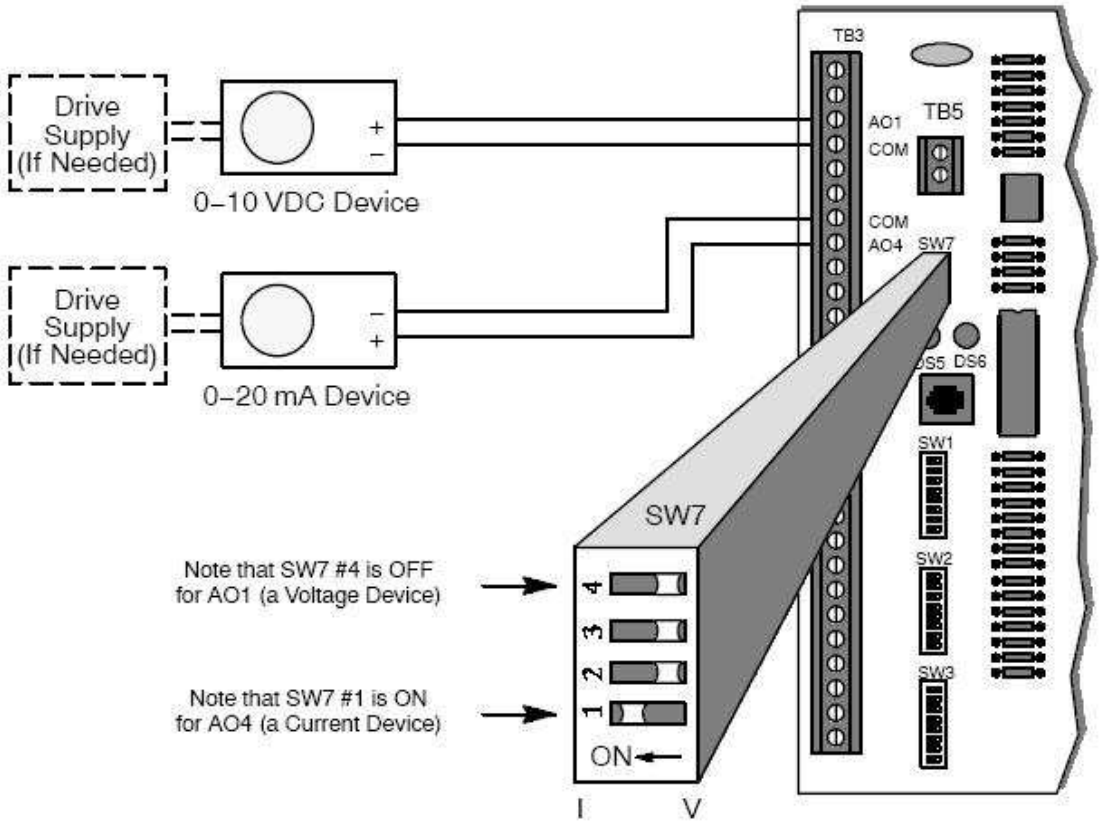
SW7 is a DIP switch block, with 4 dip switches. You use it to select whether the analog output signal for the associated analog output will be a 0 to 10 volt or a 0 to 20 milli-amp signal.

Analog outputs are used typically to feed an analog signal to devices that require one. For instance, it might be a damper or valve motor that positions itself according to the value of the current signal. If we were talking about a standard 2 to 10 volt DC device, like a Belimo valve or damper motor, it derives it's power continuously from a a wire 24 volt AC connection. And has a third wire for positioning signal. At 2 volts or below, the valve or damper would be in a closed position. At 4 volts it should be 25% open, at 6 volts it should be at 50%, 75% at 8 volts, and wide open at 10 volts.

Remember that AD (analog to digital) converter thing I mentioned back in the section about inputs? Well, for analog outputs driven by a digital controller one needs DA (digital to analog) converters. And on this controller there is one such for each of the analog outputs. The ones on this particular controller are 8 bit resolution devices. In other words, if we continuously stepped up the output from zero to 100 percent, the increase would not be absolutely smooth. Instead, if this analog output we're theorizing about is set up to produce a DC voltage signal, the signal will increment in jumps of 0.04 volts.

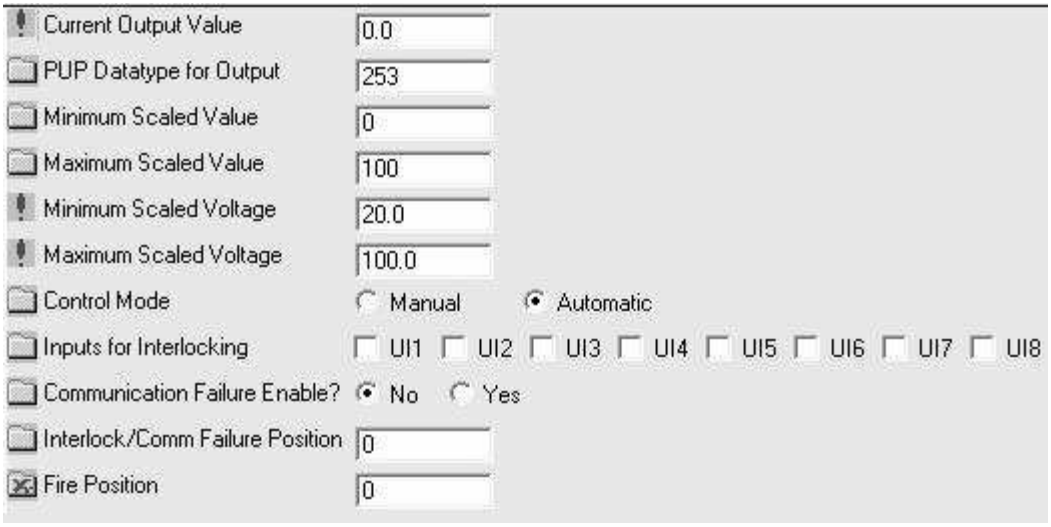
Which, in all reality is a fine enough control for most, if not all BAS applications. Tho there could be times when it is not. i.e For process control in factories and chemical plants. Certain lab equipment applications. And so forth.

Or analog outputs (signals) could represent some value being fed to some other device. Perhaps it is being used to provide a signal that represents a desired temperature setpoint for a heating boiler control panel. Or sometimes it's used to tell a chiller, which has its own control system, what the desired chiller leaving water temperature is supposed to be. Or an analog output from our BAS controller might be feeding a desired speed signal to a VFD (variable frequency drive, also called a variable speed drive). To control how fast a fan or pump is rotating.



Above is an example of how devices needing an analog position would be wired for this particular kind of controller.

Now let's take a look at the analog output configuration screen.



Here I've configured an analog output port. Minimum and Maximum Scaled values, just control that numbers will be displayed up in the Current Value block. Most times they're left at 0 and 100, respectively. To represent some percentage of full signal. But you could, for instance set min to 0 and max to 2000. So that 50% output showed as 1000 somethings; gallons, cubic feet, or whatever. Scaled value is for display purposes only, and changes nothing about the value of the actual output signal.

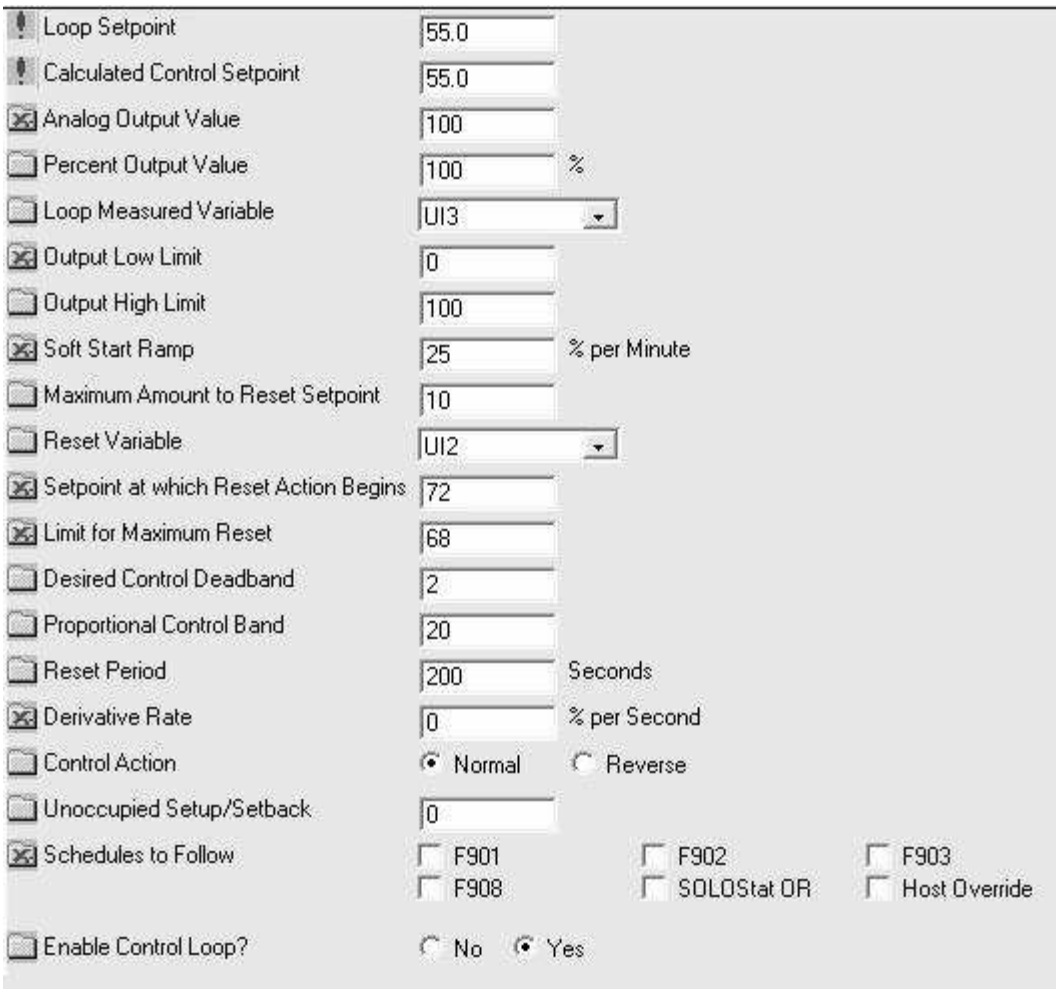
Minimum Scaled Voltage and maximum Scaled Voltage DO have a direct effect on the output. In this case, I've told the controller that the minimum scaled voltage is to be 20% of full voltage scale. In this case, that means 2 volts DC. So at an output CV (current value) you'd be able to actually measure 2 volts DC present at the associated analog output terminal.

The reason I've done that here, is that in this imaginary application, I'm setting up this analog port to control the position of a Belimo damper actuator. For the selected actuator, 2 volts = all the way closed. Zero position.

This is commonly done because it is common for stray, induced voltages to be present in DC voltage signaling wires. But such stray voltages usually are under 2 volts.

I set maximum scaled voltage to 100%, which would be 10 volts in this example. If I had a reason to limit the maximum output voltage to eight volts, for example, I could do this simply by setting max scaled voltage to 80.

I set control mode to automatic. And did not enable interlocking or fire positioning.



Okay, here is the display for the analog output control loop. In this application I'm picturing control of economizer dampers on an air handler and the use of cool outdoor air (it is winter in Minnesota as I type this) to maintain a discharge air temperature of 55 degrees. Loop measured variable is set to UI3 (mixed air temperature). Output Low Limit and Output High Limit would each limit the respective analog output. If low limit were set to 20, then the actual analog output would never go below 20 % of full scale. This might be used to set things up so that the outside air damper was always slightly open. High limit could be used to ensure it never, ever opened more than 60%, if that was what you wished.

Soft ramping controls how fast the output goes from a lower value to a higher value as a result of calculations by the controller indicating such action is needed. One uses soft ramping to ensure the dampers don't open TOO fast, for instance.

Max Reset. Reset Variable, point at which reset begins, limit for maximum reset … were all explained in the discussion of digital outputs. And those things work the same here. In this case I've set up the control to normally maintain a 55 degree mixed air temp. But if UI2 (in this case, we'll pretend it's return air temp) drops below 72 degrees, increment the calculated setpoint to a higher value. If return air temperatures drops all the way to

68 degrees, calculated setpoint will be 65 degrees. If we ‘re getting return air back from the air conditioned space which is that durned cool, discharge air temperature of 55 degrees is too durned cool.

Next the values for the PID control loop are set up. But in this case I left the derivative set to zero. Since what I really want is a PI control, not a PID control. Most times, a full PID control loop is not needed in HVAC work, and in fact may cause more problems than it would fix. Control loops that are using the derivative have a tendency to become very unstable under sudden, significant changes in either the measured variable or the setpoint.

I’m not going to discuss the details and math of Proportional; Proportional plus Integral; or Proportional, Integral and Derivative controls here. Read up on them.

The range of devices that might be used and which might need an analog signal is so wide and varied that I’m not going to attempt to display any sample pictures of such devices here. I mentioned several type devices in the discussion. And there are many more which I have not mentioned.

Analog signals are used to feed “desired position” signals to actuators which move valve or dampers open or closed; they feed desired speed signals to variable speed motor controls; they can represent some set point value, or some RESET value to a boiler or chiller controller. And so forth.

Part 4 of this series will cover the different classifications of DDC controllers as used for BAS installations and touch upon the different programming methodologies employed for those controllers which are freely programmable, in case you need one to do something that any built in control routines just can not handle.