Overview of Direct Digital Controls

For Building Automation Systems

Part 2 – Inputs/Sensors

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).

For a DDC controller (which is really just a computer dedicated to a particular kind of task) to accomplish something real and useful in a very real world it must have more than RAM, ROM, a microprocessor, and a program to tell it what to do. It must have very real, physical connections to the things which one is attempting to control.

It needs information about the current status and condition of the equipment to be controlled. Pressures, temperatures, flow rates, voltages, electrical current draw, position of mechanical parts, and so forth. These are called **INPUTS**. And the DDC controller needs the information from these in order to decide what to do, what changes to operating conditions need to be made, what actions to take (if any are needed), and so forth.

In order for a DDC to controller to actually accomplish it's function of controlling a piece of equipment (or several pieces of equipment), it also needs **OUTPUTS.** A way to turn something on or off, or to reposition a mechanical part, or whatever is needed to accomplish the control function(s).

Inputs and outputs on a DDC controller are the microprocessor's connection to the real world around it. Without inputs and outputs, that DDC controller is pretty much a useless toy; all the programming inside it, no matter how skillfully done, accomplishes nothing, does nothing of usefulness or relevance to the real, physical world.

INPUTS

Inputs are information providers to the "brain" of the DDC controller. Just as your sense of touch, smell, vision, hearing, and taste are information providers to your own brain.

An input might be of the type where a sensor of some sort, placed in some appropriate spot on or in the equipment to be controlled is hard-wired back to a physical, electrical connection (terminals) on the controller. With the sensor outputting a varying resistance, voltage, current flow, or modulated pulse width or frequency that is proportional to the current value of whatever the sensor was designed to sense. The sensor might be sensing temperature, pressure, force, flow rate, position, voltage or electrical current levels, or whatever.

Or there are “intelligent”, networked sensors which also act as inputs for a DDC controller. In this case the sensor itself has its own microprocessor, RAM, ROM, network driver circuitry, and built in programming. As well as the usual needed circuitry to sense whatever is being measured. In this case the sensor “reads” the current value of whatever it was designed to sense, converts that information from an analog signal into digital information. Then adds whatever information it needs to add to conform to the networking protocol being used; such as identification data that will tell the recipient (a DDC controller) what input the sensor is, data about which controller the information is intended for, data error detection and checking information, and so forth. Then this “packet” of information is sent out onto the network and routed to the intended recipient, or in the case of some control network protocols, it is simply sent to ALL devices on the network. This is all done in much the same way as with the regular computer networks with which most of you are familiar.   The protocol may be different, but the idea is the same. So what the recipient DDC controller receives is not a varying and proportional direct current voltage signal (for instance) that represents a pressure being measured. Rather, it receives a packet of information over a computer network that identifies the sender, gives a definite value in hexadecimal numbers or in ASCII, and usually includes miscellaneous data such as the data type (binary, integer, floating point, how many decimal places, etc), and more.

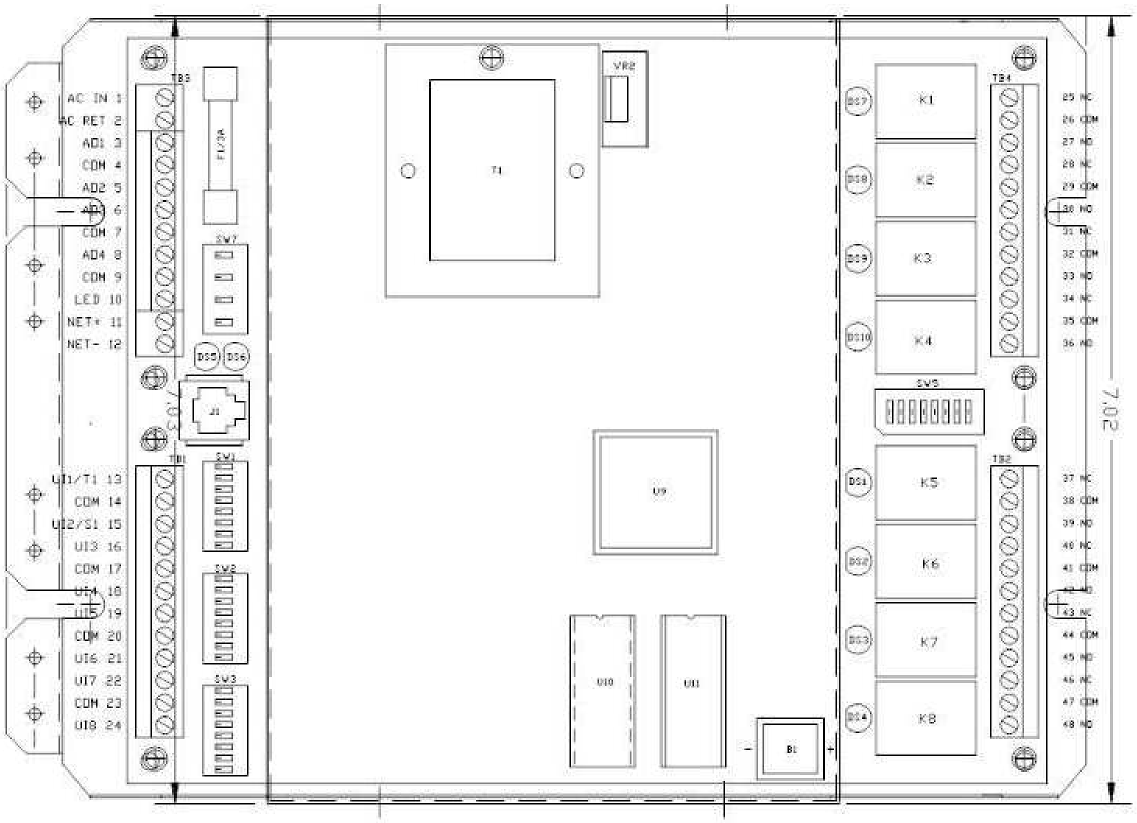
Networked sensors have the advantage that there does not need to be separate, physical connection terminals on the controller for each, separate sensor being used in the system. Nor do you need to run 2, 3, or 4 wires each (depending on sensor type) all the way from the controller, to each and every sensor in the system. Which can be time and labor and material consuming. Not to mention that long runs of wiring can cause problems due to voltage drops, picking up electromagnetic interference, and so forth. You can simply run one network cable (1 to 8 wires depending on network type and protocol) from a single port on the DDC controller, on to each sensor. Daisy chained (from one device to the next), or in a Star/Free Topology configuration, or in a combination of the two. Again, dependent on the network type and protocol being used. (If you have any interest in really learning about DDC controllers and automation, you should probably read up on computer/data network theory and basic principles.)

On the other hand, networked sensors have their drawbacks. Price per sensor is generally much higher than is the case with regular sensors. Depending on the network type and protocol used, the network cable itself might cost considerably more per foot than ordinary wiring used with conventional sensors … and it may be more easily damaged than standard 18 gage, stranded, twisted and shielded cabling that is commonly used with conventional sensors. Meaning that network type wiring MIGHT be not only more costly per foot, it might cost more to install since the installers may have to be more careful (and slow) during the installation. Also consider the following. With conventional sensors, each wired independently to separate and discrete terminals on the DDC controller, interference induced in any one set of cables (for one sensor to input connection), or the cable itself being cut, means that only THAT sensor’s information is lost. (Usually) Whereas in a networked sensor system; induced interference, faulty data transmission due to bad connections or damaged wires, or cut wiring usually means the loss of information from many or all of the sensors. Computer data networks with faults in them can also be devilishly hard to troubleshoot, in order to determine the cause and location of the fault. And may require expensive troubleshooting instruments and/or software. Conventional sensors, with conventional wiring usually only require a standard DMM (digital multi-meter) for the troubleshooting of the sensor itself and the wiring from it to the controller. Lastly, networked sensors have more parts and circuitry … which means there are more possible points of failure as compared to conventional sensors.

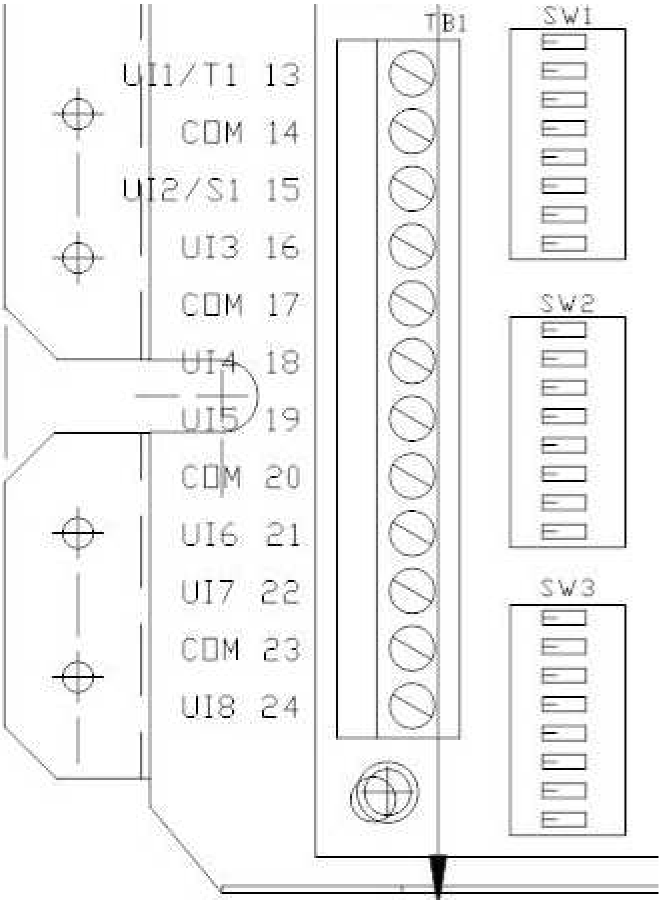
All the above that was mentioned about wired, networked sensors, also holds true for wireless, networked sensors. Which are available. But which are even more expensive per sensor device. And have even more problems in their usage. There are many things which can interfere with a radio signal from a wireless sensor to a DDC controller. Distance, signal strength loss due to walls and structural metal, EMI (electro-magnetic interference) from any number of sources, and so on. All of which can require some detailed thought and study before electing to use such devices, and which may necessitate the further installation of repeaters, signal boosters, etc. With the end result that the wireless sensor, which looked so good at first glance, ends up costing FAR more in initial cost of installation, and maintenance over the years, than what it would have cost to use a wired sensor.

DDC controllers may also get input data from other DDC controllers. i.e. A common practice for a building with an automation system in it is for there to be an outdoor temperature sensor wired to one of the controllers. And that controller, in turn, not only uses that information itself, but broadcasts the current outdoor temperature over the network to all other controllers in the building. In some installations I've done, a single controller, located in one building, reads the outdoor temperature from one of its sensors and then broadcasts that value not only to other controllers in that building, but also broadcasts that value to controllers in other buildings owned by the same customer. The data flows from the originating controller, out onto their LAN, out over the Internet, and then to LANs in their other buildings, and onto other DDC controllers. Not necessarily a wise idea, BTW. As anyone who is much of an outdoorsman can tell you, or anyone who has been in the HVAC business for long, local temperatures can vary several degrees in a matter of just a few miles.

Okay, let's look at a DDC controller that has discrete, wired inputs. I'll confine myself to talking about such for the rest of this section that concerns inputs.



The above is extracted from a CAD drawing and shows one type of general purpose controller (I'll explain what that means in a later discussion), from one, particular manufacturer. There are both many kinds of DDC controllers, and many different manufacturers. I show this one, as it's nicely generic. The view is with the cover removed. The lower left hand side shows a terminal board. For this controller, that is the terminal board for the inputs. Let's take a closer view there.

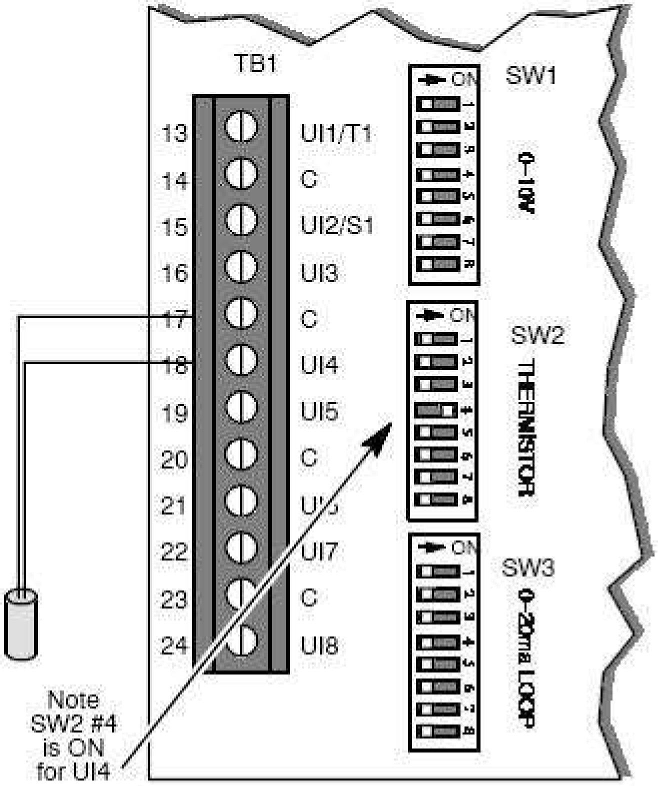


You can now see that the terminal board is labeled TB1. And each terminal is labeled.

UI1, UI2, etc means “Universal Input” number 1, 2, etc. Some makes and models of controllers have dedicated inputs. That is to say that the input might accept ONLY a resistance input, or only a voltage input, or whatever. But common in the DDC world is the usage of universal inputs. Each universal input on this controller, for instance, can handle resistance, dry contacts (like an open or closed switch), voltage, or current. Next to the terminal board, on the right, are shown 3 DIP switch blocks. Marked SW1, SW2, and SW3. Each DIP switch block has eight individual dip switches. One dipswitch each, for UI1 thru UI8. Each switch on SW1 selects whether or not the related UI is configured electronically to accept a voltage type input device. SW2 is for selecting a resistance type device. SW3 is for selecting a milli-amp signaling device.

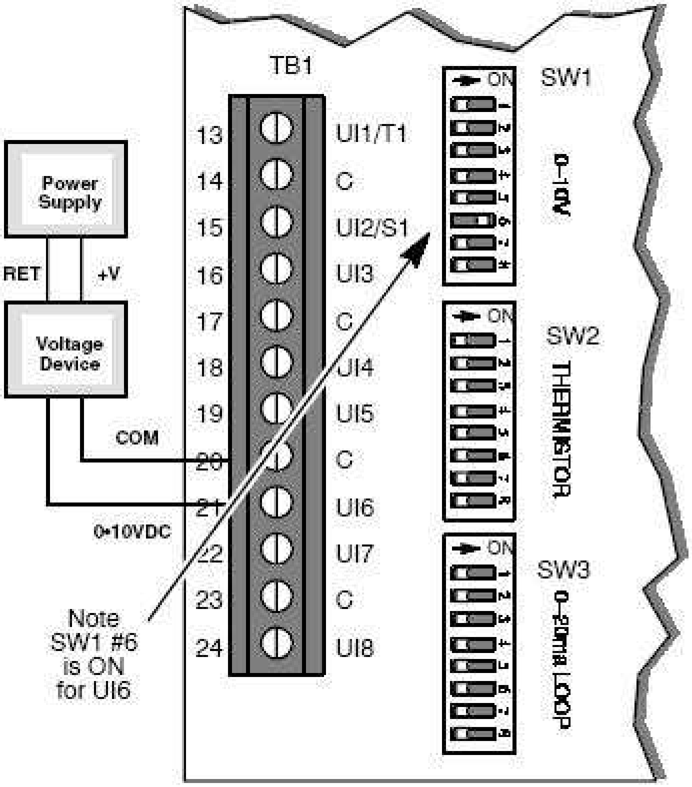
For instance, if UI1 (input one) was going to have a temperature sensing, varying resistance type sensor attached to it. The first switch of SW2 would be flipped to the “On” position, and the first switches of SW1 and SW3 would be moved to the “Off” position. This actually cuts in and out certain electronic components that terminal 13 is connected to.

If one were then going to connect a sensor which gives a 0 to 10 volt DC output signal to UI2, then switch 2 of SW1 would be placed in the “On” position, and switch 2 of SW2 and SW3 would be moved to the “Off” position. And so forth. For input 1, if a 0-5 volt device were to be connected, switch 1 of SW1, SW2, and SW3 would all be turned off. And if we want to sense the opening or closing of a dry contact (like a switch), then the appropriate input would be configured for a resistance device. In this case what happens is that, with this particular make and model of controller (it varies from manufacturer to manufacturer and model to model), every 100 milli-seconds the controller itself outputs a 5 volt DC signal at the input terminal. Then it compares the voltage read between that terminal and a common (ground) terminal. If it senses 5 volts DC, the switch must be open. If it senses 2 volts or less, the switch must be closed. Simple, isn’t it? This isn’t rocket science folks. If I can learn to understand this stuff, anybody can.

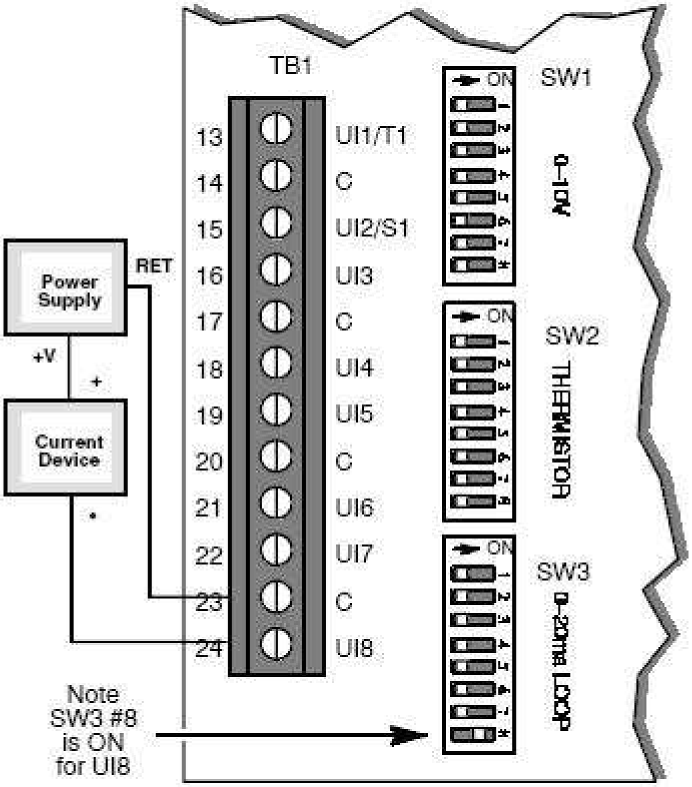


Note
SW2 #4
is ON
for UI4

A thermistor (a type of temperature sensor whose electrical resistance varies according to the temperature it is sensing) connected to input #4.



A sensor that outputs a proportional 0 to 10 volt DC signal, connected to input #6.

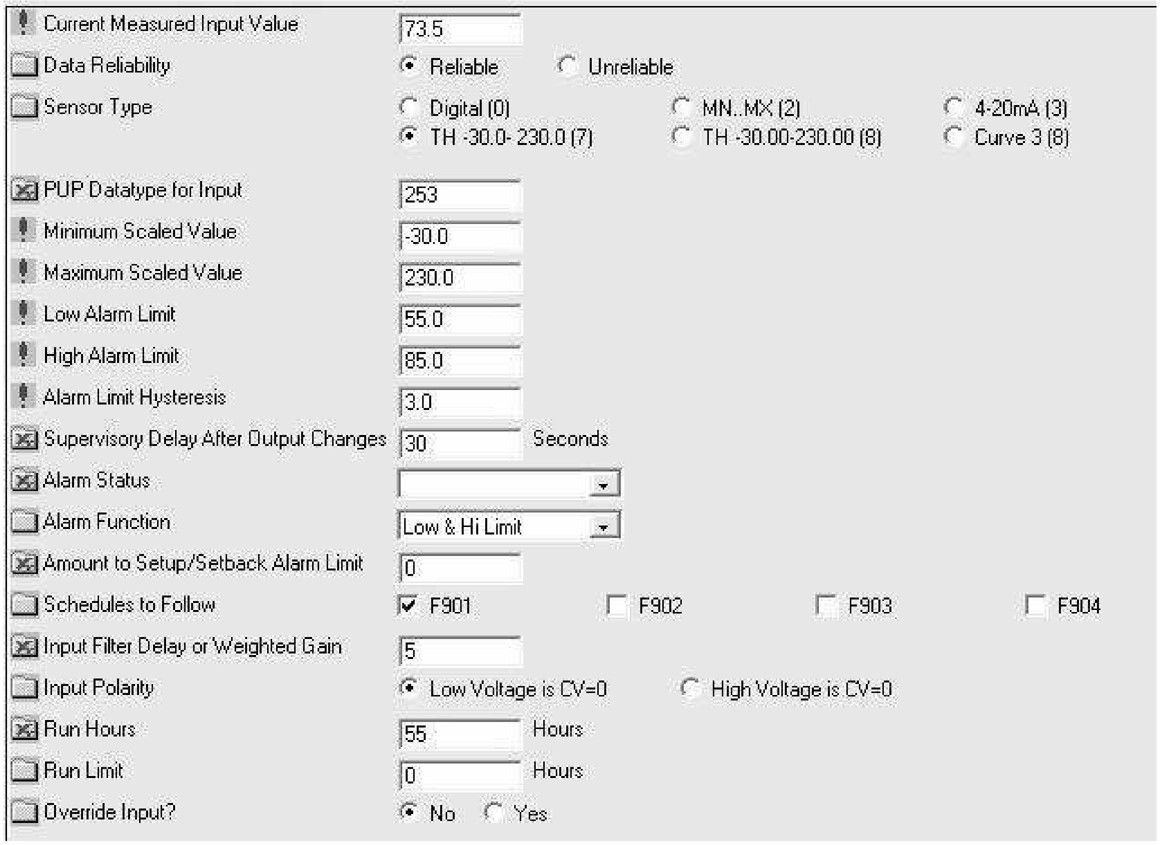


A sensor that outputs a 0 to 20, or a 4 to 20 milliamp signal connected to input #8.

Now, just connecting a sensor to a controller’s input, is only a part of the task. Remember, I stated that this particular controller is a type that is called “general purpose”. In other words, the very same controller could be used to control all sorts of things. AND … all sorts of sensors might be attached to its inputs.

The “brain” of the controller has no way of knowing what you attached to where, what information it is supposed to gather from an input (or what to do with it but that’s another part to this series), and so forth. So JUST wiring sensors to it is only part of the job.

Following the wiring of an input, you need to configure the software inside the controller in order to give it some information to work with. So let’s take a look at that process, for this particular controller.



Current Measured Input Value	73.5
Data Reliability	● Reliable ○ Unreliable
Sensor Type	○ Digital (0) ○ MN..MX (2) ○ 4-20mA (3) ● TH -30.0- 230.0 (7) ○ TH -30.00-230.00 (8) ○ Curve 3 (8)
PUP Datatype for Input	253
Minimum Scaled Value	-30.0
Maximum Scaled Value	230.0
Low Alarm Limit	55.0
High Alarm Limit	85.0
Alarm Limit Hysteresis	3.0
Supervisory Delay After Output Changes	30 Seconds
Alarm Status	
Alarm Function	Low & Hi Limit
Amount to Setup/Setback Alarm Limit	0
Schedules to Follow	☑ F901 ☐ F902 ☐ F903 ☐ F904
Input Filter Delay or Weighted Gain	5
Input Polarity	● Low Voltage is CV=0 ○ High Voltage is CV=0
Run Hours	55 Hours
Run Limit	0 Hours
Override Input?	● No ○ Yes

Here we have a view of a screen for configuring an input for our example controller, as seen using some software from the manufacturer that is used for configuring, monitoring, troubleshooting, and programming such devices.

In this example I am configuring, setting up, an input to which a temperature sensor has been attached.

Selected for sensor type, was a selection telling the controller that the attached sensor was a Type 3 Precon thermistor. Thermistors don’t have a linear change of their internal resistance in relation to changes in the temperature to which they are exposed. So it is not so simple as to say that each 1 degree of temperature change always equals some set number of ohms change in the resistance. But in this case, there is a math function built into the firmware of this controller so that it knows the temperature versus resistance change characteristics for a Type 3 Precon thermistor, so it’s a simple matter of clicking on that selection.

If I were to use some other type of resistance based temperature sensor, I’d need to select a piece-wise curve as the input type, then go to the table associated with that selection and fill in the data tables so the controller could do the mathematics to interpolate a current reading of temperature. But I’m keeping things simple here.

On the screen shot, above, you see things like the current temperature reading, the data type selected (in this case a floating point number with one decimal place), the minimum and maximum range scaling capability of the sensor (-30 to 230 ‘F), and the fact that the controller is telling me that the input from the sensor is “Reliable”. That means that the controller senses an electrical resistance on this input that is within the expected range for the selected type of sensor. Otherwise it’d be showing “Unreliable”.

And there are other pieces of information I give the controller.

Under “Low Alarm Limit” and “Hi Alarm Limit”, I’ve told it to send out an alarm if the current measured temperature is lower than 55 ‘F, or higher than 85 ‘F. Under Alarm Function, I’ve told it to send out an alarm for EITHER a low or a high temperature condition. But under Supervisory Delay, I’ve also told it to wait until a temperature has reached an alarm point and stayed there for at least 30 seconds before sending out an alarm. That’s to avoid nuisance alarms. Such as short term temperature swings caused when somebody opened a door and a short lived cold blast of air hit the sensor. The Alarm Hysteresis number means that if the controller sends out the alarm because the temperature hit 85 degrees, it won’t clear the alarm (stop sending out alarm alerts) until the temperature drops to 82 degrees. (Or 55 and 58 degrees in the case of a low temperature alarm)

I’m not using Alarm Setup or Setback in this example. But if I did, and put a value of 10 in that block, then the controller would check the schedule whose block is checked. And if the schedule indicated that this was an “Unoccupied” time of day, or a holiday, etc … it wouldn’t alarm until the temperature dropped to below 45 degrees, or raised past 95 degrees. I could check multiple blocks and have it check several, independent schedules that I might have set up.

The input filtering delay whose value I set to 5, just means that the current value shown won’t be the current instantaneous value; it’ll be one that is the average over a certain time frame. I do that since with very accurate and relatively quick measurements which these kinds of thermistors are capable of producing, I don’t want the controller reacting to every minor fluctuation of temperature for every eddy current it feels. And without that filtering, the display of the current value would be constantly changing back and forth by a 10th, or a few 10th’s of a degree, to no useful purpose in a HVAC application.

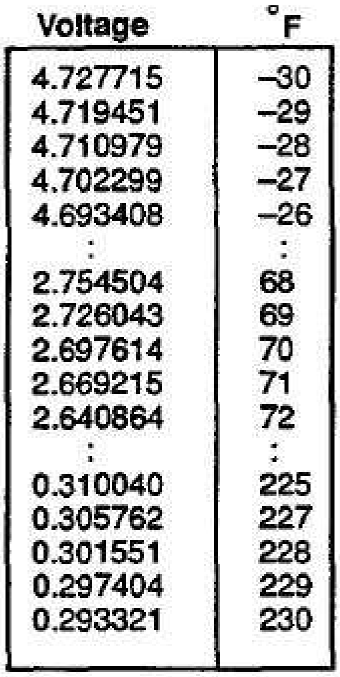
Input polarity, in this case, is irrelevant. Run Hours shows 55 hours. In this case, this really means nothing. The input is simply keeping track of how long it’s had an input at this terminal that’s not zero (or minus 30 degrees in this case). Run hours can be useful for some things. If this input were set up as a digital input, where a zero meant a motor was off, and a 1 meant the motor was running. Then you can keep track of the running hours of a particular motor very easily. Enter a number into the run hours limit block, and if that motor exceeds that many hours of run time, the controller will send out a run hours limit exceeded alarm.

Last, but not least, there is the block for over-riding the input. There are several reasons one might want to do this. The most common reason being to conduct testing. Suppose this controller were set up and programmed to control an air conditioning system. And suppose that this temperature sensor was in a room. And you wanted to know if the controller would start the actions that would cool the room, if this temperature exceeded 75 degrees. But the room isn't really 75 degrees. You could over-ride the current, real temperature and type in whatever you wanted, say 80 degrees, and then watch to see if the cooling came on. Etc. Just make sure you turn the over-ride OFF when you're done testing.

This gives you a glimpse inside the controller. Within it's software. So that you can see that there is more involved in setting up an input, besides just attaching the wires.

For a flow measuring device, for instance, that outputs a 4 to 20 milliamp signal, and which has a range of 0 to 2000 CFM, and produces a linear signal. You need to not only wire it correctly, set the dip switches for the controller's input correctly, but also tell the controller several pieces of information about that input. That the sensor type is 4 to 20 millamps, that the data type is an integer, that the range scale minimum is 0 and the maximum is 2000, etc.

If your flow sensor did not put out a signal that was strictly linear in relation to the flow sensed, then you'd need to set up a piece-wise curve table (also called "curve", "interpolation table" and other terms) and enter the suitable data, usually obtained from the sensor manufacturer. An example of such a table for a temperature sensor which outputs a DC voltage from 0 to 5 volts:



Voltage	°F
4.727715	–30
4.719451	–29
4.710979	–28
4.702299	–27
4.693408	–26
:	:
2.754504	68
2.726043	69
2.697614	70
2.669215	71
2.640864	72
:	:
0.310040	225
0.305762	227
0.301551	228
0.297404	229
0.293321	230

The numbers from the left and right hand columns would be entered in the appropriate places in the piece-wise curve table. And from then on the controller would read the voltage present at that input terminal, and mathematically interpolate a corresponding temperature. Which is what would be displayed in the Current Value block.

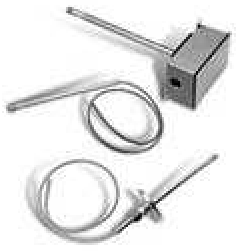
Another thing to think about as concerns inputs for DDC controllers. These ARE digital devices. Not analog devices. Electronically, their inputs may accept an analog signal, but within the electronics of the controller that analog signal is sent through an AD converter. That's an analog to digital converter. Such AD converters have a listed maximum resolution. Which varies from model to model and manufacturer to manufacturer. What does this mean to you?

Let's suppose you have a very high grade sensor that you're attaching to your controller. And it is capable of measuring pressure from 0 to 100 PSI in .001 PSI increments. This does not mean that your controller and its input can necessarily read and display a number like 63.104 PSI. What it can discern, is limited by the resolution of its built-in AD converter. In this example, the sensor is capable of sending up to 100,000 discrete possible values representing what the current pressure being sensed is. But if your controller has only an 8-bit AD converter for its input it is only going to be able to read and display changes in value of approximately 0.4 PSI. If, on the other hand, the input has a 15 bit converter, it will be able to detect increments of .003 PSI. In this example. So input resolution is something to consider when picking a controller for a particular application.

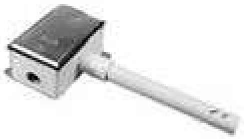
With many controllers, you can also set up inputs that are the results of computations, math functions, and/or Boolean operations. Or which are data which is actually originated elsewhere and sent to this controller over the network.

Now, let's look at some typical sensors that are used in HVAC (Heating, Ventilation, and Air Conditioning) DDC control applications.

The most common information needed for control of HVAC systems is temperature. As one might expect, temperature sensors come in all sorts of styles and designs, specific features that vary according to application and intended place where they are to be installed. Etc. Temperature sensors may be RTD, thermocouple, or thermistor in type. The type is picked IAW the desired application and need. The sensors may output a signal that is a varying electrical resistance, a varying DC voltage, or a varying milli-amp current. Below are some small pictures of a sampling of temperature sensor types and styles.

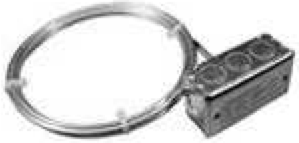


Duct Insertion type temperature sensors

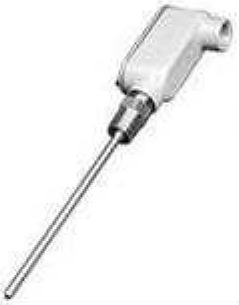


Temperature sensor for mounting outdoors

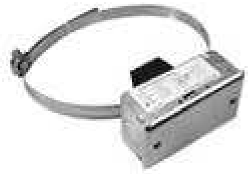
Wall mounted temperature sensor



Duct Insertion type AVERAGING temperature sensor



Immersion type temperature sensor



Strap-on type temperature sensor

Note the coiled tubing of the averaging type temperature sensor. Such sensors typically come with 8, 12, or 24 foot length sensor coils. Which might contain either 4 or 9 thermistors wired in a series-parallel configuration such that the outputs represents a mathematical summing of the temperatures sensed by each of its elements. The coil is unwound and strung up to spread across a wide area, such as the inside of a large air duct. So that one can obtain an average temperature reading for an air stream which might be stratified (layered) such that there is a major difference in the temperature sensed between this location and that.

The immersion type temperature sensor is made to be threaded into a standard female pipe threaded fitting.

Now let’s look at some varying types of pressure sensors.

Fluid pressure differential pressure transmitter

Duct air pressure transmitters

Pressure sensor that would be screwed into a standard pipe threaded fitting

On the left is a type of sensor which has two threaded pressure ports. One is designated high pressure; the other is for the low side pressure. Tubing is run from each to, for instance, the supply and the return sides of a piped water system. This type sensor will determine the difference in pressure between the two points and produce an output called a DIFFERENTIAL PRESSURE. From which, if desired one can mathematically derive a flow rate for whatever is being measured. Or one can simply set up a controller to maintain a pre-determined differential pressure which represents a value calculated to ensure an adequate flow for the application purposes. The duct air pressure sensors serve much the same purposes. The top one is a standard duct static pressure sensor, so that a certain pressure may be maintained by a controller. The bottom type has two pressure sensing ports and is used like the previously discussed fluid differential pressure transmitter. It might be used to sense a pressure drop across a filter or other component. Or might be used to sense a comparative air pressure differential between the air inside a building to that of the outside air. In order to facilitate the control of a buildings internal air pressure, which typically is maintained at some positive value as compared to the outside. The screw-in type pressure sensor is typically used to measure water, oil, refrigerant, or steam pressures in piped systems.

Another item typically sensed in HVAC systems is the relative humidity of air. This is used for various purposes. Read up on HVAC theory if you don’t know why a HVAC system might need to know a relative humidity.

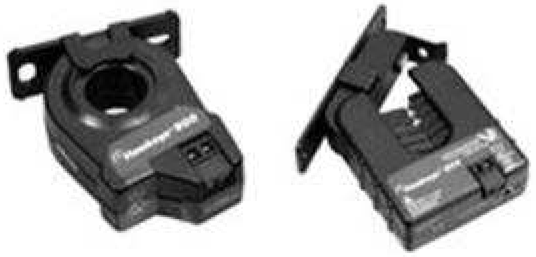
Duct Insertion type humidity sensors

Humidity sensor for mounting outdoors

Wall mounted humidity sensor

As with other sensors previously shown, humidity sensors come in a variety of designs.

In HVAC applications we also commonly want to sense the presence of electrical current, or even a precise, measured value for what amount of current is present at some point. Below are pictures of typical current sensors. With the type on the left, a wire is run through the hole in the sensor “doughnut” and then connected to a motor or whatever. The second style can be opened up and simply clamped around the desired wire. Current sensors may be of a type that is only “Off” and “On”. Meaning they contain an internal switch that opens or closed to indicate the presence of current flow. Such types, which are cheaper than others, are frequently used just to feed the information back to a controller’s input to indicate that an electric motor is either running, or not. The second type can measure and transmit back to the controller an actual amperage value.



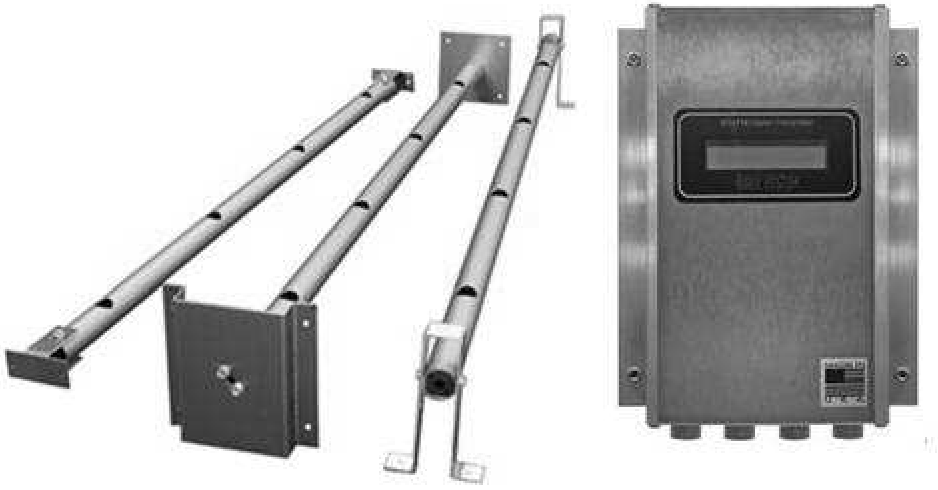
Electrical current sensors

Another type sensor often used in HVAC applications are sensors to monitor and measure the amount of gas concentration of a particular gas in the air of a room, within an air conditioning duct, etc. The gas monitored for and measured could be carbon dioxide, carbon monoxide, any number of flammable or explosive gases, leaking refrigerants, or whatever. Gas concentration sensors are available for hundreds of types of gases.

Duct Insertion type gas concentraion sensor

Wall mounted gas concentration sensor

It is often also desirable to directly measure flow rates of some gas or fluid. Below is a picture of a common type air flow sensor, capable of producing an output that directly represents CFM (cubic feet per minute). On the left are various style sensing probes, on the right is the transmitter. There are such a wide variety of types of flow measuring instruments that I'm not going to bother to show a wide sampling here.



Air Flow type sensors and the sensor transmitter

In addition to all the above, one might measure and monitor actual voltages present at critical points, 3 phase power supply faults such as voltage imbalances and phase reversal, time, speed (linear or rotational), position of an air damper or valve, the amount of light (in lumens) outside (in order to turn off outdoor lighting that is no longer needed), and so forth and so on.

It is not my intention here to discuss everything there is to know about sensors and DDC controller inputs. It is a broad subject, and for design engineers, who design sensors or controller inputs, either is a subject that one can spend a career studying and specializing in.

Part 3 of this series will be about OUTPUTS.