

## **WebXL Control Operation**

The WebXL system is an object orientated control system. The system is configured using functional blocks. Each block takes one or more inputs and has one or more outputs. Inputs and outputs can be either analogue or binary. In addition to the physical analogue inputs (AI) , analogue outputs (AO), binary inputs (BI) and binary outputs (BO) there are internal analogue variables (AV) and binary variables (BV).

Programming the system to do a specific task means using one or more functional blocks. These can be cascaded if required. A lot of control operation can be done by just using a single block. Inputs and outputs also have a limited set of functions which provide additional control. For example, binary outputs have the capability of incorporating minimum off or on time control. This is useful where compressors must be turned off for a minimum period of time before being allowed to restart.

## **WebXL functional blocks - overview**

### **Analogue input - Analogue output blocks**

These blocks take one or more analogue inputs and use them to generate a single analogue output.

#### **Continuous loops**

These are the main control loops and are used to control the position of things like dampers in response to changes in an input. They provide PID control.

#### **Heating/cooling loops**

These use 2 continuous loops to provide both a heating and cooling function. They provide a shortcut method of otherwise configuring 2 separate loops. If a control needs both heating and cooling controlled from a single input then it is easier to use a heating/cooling pair rather than configure 2 2 separate continuous functions as the heating/cooling pair shares the same input and sets the loop polarities automatically.

#### **Arithmetic functions**

These functions provide a variety of commonly used functions involving one or more analogue inputs. They can be used to do things like derive an average temperature from a number of sensors, provide a slope conversion on an input to adjust scaling factors, multiply or divide an input by another input, generate the enthalpy from temperature and humidity.

### **Analogue input - Binary output**

These functions allow the conversion of an analogue input to a binary output.

#### **Discrete functions**

Discrete functions provide a discriminator or switch type of control as you might find in a thermostat. If the analogue input is above a threshold it turns the output on. If it is below the threshold it turns the output off. The function provides more complex control than this because you can set up both a switch type function or a window type of function. For further details refer to the Discrete function description.

### **Binary input - Binary output**

These functions take one or more binary inputs and use them to generate single binary output.

#### **Delay functions**

These functions are used to provide time delays. For example, they can be used to take an input and delay its appearance at the output or they can take an input and stretch the input in time so that the output remains on for a specified time after the input turns off. Delay functions can also be used to provide a variable delay depending on the values of analogue points. This allows the implementation of optimum start/stop operation.

#### **Logic functions**

Logic functions take a number of inputs and use the logical combination of these to provide a binary output to the function. They are extremely useful in creating an output which depends on the state of a number of inputs. They can also be used to generate state machines which allow the sequencing of operations.

#### **Lead/lag functions**

These functions are used to provide the rotation of use of pieces of equipment so as to equalise their use. Fault control inputs allow the control to bypass faulty units.

## Time control functions

These provide for time control of the system.

### **Time schedules**

Time schedules allow the control of operation depending on the time of day and day of the week. Individual schedules can be set up to turn a binary variable on at a particular time of day and off at a particular time of day. The schedule can be made to operate only on certain days of the weeks or holiday exceptions. More complex time schedules can be created by grouping individual time schedules. Each schedule (single or group) drives a binary variable so the schedules can be used in all functions which accept a binary input.

### **Holiday schedules**

Holiday schedules come in two flavours. The simplest of these is just setting a particular day from midnight until midnight as a holiday. A more complex mode of configuration allows a holiday schedule which is true from a particular time of day on a particular day until a specified time and date. Like time schedules, holiday schedules drive digital variables and as such can be used in any other function which accepts a binary input. In addition the holiday schedules are linked to the time schedules so that they can be used to enable or disable time schedules.

## Alarm functions

These functions are used to provide exception reporting. They let the operator know when something isn't working as expected. Each alarm function can also be programmed to send a notification by email to one or more recipients.

## Message functions

These functions serve no purpose in an application where only one controller is used. They are meant for use in providing and receiving information from other controllers in a system.

## **WebXL Continuous loops & Heating/Cooling Loops**

These functions are used where proportional control is required. Heating/Cooling loops are just a pair of continuous loops which share the same input. They provide a convenient shortcut for doing both heating and cooling control. Everything discussed below applies to both heating/cooling loops and continuous loops.

### **What is a Continuous loop?**

A continuous loop takes an analogue value as an input and compares it to a setpoint. The difference, or error value, is used to drive the output which then drives the controlling device in such a way as to minimise the error. The simplest form of loop just multiplies the error by a constant value. The value of the multiplier is called the **proportional gain (Pg)**. The equation for this is:

$$\text{Output} = \text{error} * \text{proportional gain}$$

The disadvantage of this type of loop is that there must always be an error for the output to be nonzero.

One can add a fixed value to the output to set the error to be zero at one value. This makes the loop become a **Reset** loop. The equation for this is:

$$\text{Output} = (\text{error} * \text{Pg}) + \text{Reset value}$$

The reset value is normally set to the expected required output under normal conditions.

### **Integral gain**

The error can be removed if we add an **Integral gain (Pi)** to the loop. Integral gain effectively sums the error over time and multiplies this sum by the integral gain and adds the result to the output. This has the effect of gradually driving the output higher or lower as time progresses which results in the error eventually reaching zero. The integral gain has to be kept fairly low otherwise the loop can become unstable and the output will switch between fully on to fully off. Integral gain is extremely useful in providing an adaptive control loop.

### **Differential gain**

The problem with using integral gain is that it takes time for the loop to reach equilibrium. While this works perfectly well in normal building control it has problems with fast responding control such as steam control. We can add another type of control called **Differential gain (Pd)** to a control loop to handle this sort of control. Differential control looks at the rate of change of the input and multiplies this by the differential gain and adds the result to the output. Take the case of steam where the control valve opens a bit too far. You will get a very rapid increase in the output which will give a large differential error resulting in a very rapid loop response. Differential gains have to be kept fairly low to keep the loop stable.

### **PID loop gains**

The use of the three types of gain gives what is called a PID loop. In a building the typical sort of proportional gain used is about xxx. If integral gain is used a typical starting point is about one hundredth of the proportional gain. The differential gain is likely to be similar in value to the integral gain. Always start at a conservative value and only increase the gains slowly if required. If the loop becomes unstable dramatically reduce the gain values.

### **Deadband**

To prevent actuator wear it is common practice to use a deadband in a continuous loop. The deadband is the amount the input can change without causing any change in the output. For example, if you have a setpoint of 21°C with a deadband of 0.5°C then the temperature can vary from the setpoint by ±0.5°C before the output will change.

### **Heat/Cool loops**

Heating/Cooling loops use 2 loops to generate a heating and a cooling input. They share a common input and otherwise are identical to using 2 separate loops, one for heating and one for cooling. The advantage of using a Heating/Cooling loop compared to using 2 separate loops is that there is less data to enter when configuring the loop and the polarity for the heating and cooling outputs is automatically selected.

## **WebXL Arithmetic functions**

The arithmetic functions in the WebXL system accept up to 16 inputs, which have to be analogue points or fixed values. The output of an arithmetic function is an analogue output or variable. A variety of arithmetic functions are implemented as detailed below.

Note that all inputs to a function must use the same units and scale factor otherwise the output is probably meaningless. All calculations are done using the internal 12 bit value for a point. The outputs and calculations are protected from returning an invalid result so that numbers greater than 4095 are limited to 4095 and negative numbers are set to 0.

### **Maximum value**

This functions looks at each input and returns the maximum value found.

### **Minimum value**

This functions looks at each input and returns the minimum value found.

### **Average value**

This functions looks at each input and returns the average value of all the values.

### **Median value**

This functions looks at each input and returns the median value of the inputs. The median is the midpoint between the minimum and the maximum.

### **Sum**

This function returns the sum of all the inputs. Note that this can result in a value which is greater than the internal 12 bit representation. The result is limited at the maximum internal value.

### **Enthalpy**

This function takes a temperature as the first entry and the humidity as the second entry and returns the enthalpy. The calculations are based on a temperature range of 0-100°C and a humidity between 0 and 100%.

### **Difference**

This function only uses 2 inputs and computes the difference between the first and the second entries as:

$$\text{Output} = \text{first input} - \text{second input.}$$

If the result is less than zero the result is set to 0.

### **Multiply**

This function multiplies the first entry by the second entry and divides the result by the scale factor. The scale factor can be 1, 10 or 100. The result is limited to 4095 which is the maximum value permitted in 12 bit arithmetic.

$$\text{Output} = (\text{first input} - \text{second input}) / \text{scale factor}$$

### **Divide**

This function multiplies the first entry by the scale factor and then divides the result by the second entry. The result is limited to 4095.

$$\text{Output} = (\text{first input} * \text{scale factor}) / \text{second input}$$

### **Slope and Offset**

This function implements the function:

$$\text{Output} = ((\text{first input} * \text{second input}) / \text{scale factor}) + \text{third input}$$

Which represents the equation:  $y = ax + b$  where a is the slope of the line and b is the offset.

## Reverse Slope and Offset

This function implements the function:

$$\text{Output} = \text{third input} - ((\text{first input} * \text{second input}) / \text{scale factor})$$

Which represents the equation:  $y = b - ax$  where  $a$  is the slope of the line and  $b$  is the offset.

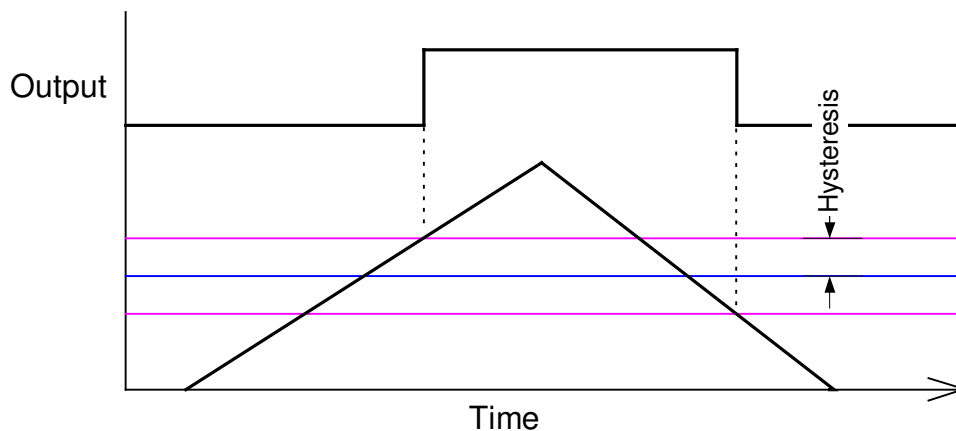
## WebXL Discrete loops

Discrete loops provide a discriminator type of function. They take an analogue value as an input and compare this to a setpoint. The output of the loop is a binary value. If the input is above the setpoint then the output will be ON, if the input is below the setpoint the output will be OFF. This is a basic **Switch** function. The output of the loop can be inverted which means that the ON becomes OFF while the OFF becomes ON.

A **Window** function uses an upper and a lower setpoint. If the input is between the 2 setpoints the output is ON otherwise it is OFF. Again the loop output can be inverted making the output ON when the input is less than the lower setpoint or higher then the upper setpoint.

If the input is not stable then one can get repeated transitions from ON to OFF and vice versa as the input oscillates around the setpoint value. To overcome this problem **hysteresis** can be added to the function. With hysteresis the output changes from OFF to ON when the input is above the setpoint PLUS the hysteresis. It turns OFF when the input falls below the setpoint MINUS the hysteresis. A similar operation applies to a window function at both the upper and lower trigger points. If the input now oscillates around the setpoint the output remains stable.

The hysteresis value is chosen to accommodate expected small scale fluctuations in the input.



## **WebXL Delay functions**

Delay functions are used when you need something to happen for a longer, or possibly shorter, time than a binary input is present for. An example of this is when a light should remain on for a few seconds after the switch controlling it is turned off. This is used in modern vehicles to allow the cabin light to remain on while the person is locking the car. The WebXL system allows the generation of a number of commonly used types of delay. In addition, a delay where the time delay can be controlled by an analogue point is very useful in implementing energy saving optimised start/stop operation of plant.

### **Retriggerable fixed width delay**

This form of delay drives the output ON whenever the input turns ON. The output stays on for a fixed time irrespective of how long the input is ON for. If the input turns OFF and then ON the delay gets restarted (re-triggered) so that the output will be ON for the delay time after the last input transition from OFF to ON.

### **Non-retriggerable fixed width delay**

This form of delay drives the output ON whenever the input turns ON. After that time the input is ignored until the delay finishes. Thus this form of function ignores anything on the input while the delay time is active.

### **Input lengthened by delay time**

This delay type is used to delay the turnon of the output. The output stays OFF after the input turns ON and stays OFF until the delay time after which the output follows the input. Multiple transitions of the input while the delay is active reset the delay.

The diagram below shows how these various delays work..

### **Reverse operation**

Triggering of the delay can be either on the OFF to ON (rising) edge of an input or on the ON to OFF transition (falling edge). The diagrams indicate what happens on the rising edge. Reverse operation inverts the Input.

### **Variable delay**

Variable delay uses a number of other analogue parameters which control the delay time of the function. The delay time varies as a function of the source, setpoint, low and high limits at the time the input to the function becomes true (ON for normal operation or OFF for reverse action). If the source is below the setpoint then the delay is:

$$\text{Delay} = \text{delay time} * (1 - ((\text{setpoint} - \text{source}) / (\text{setpoint} - \text{low limit})))$$

While if the source is above the setpoint the delay becomes:

$$\text{Delay} = \text{delay time} * (1 - ((\text{source} - \text{setpoint}) / (\text{high limit} - \text{setpoint})))$$

Reverse action on the **Variable delay** changes these formulae to:

Source < setpoint:

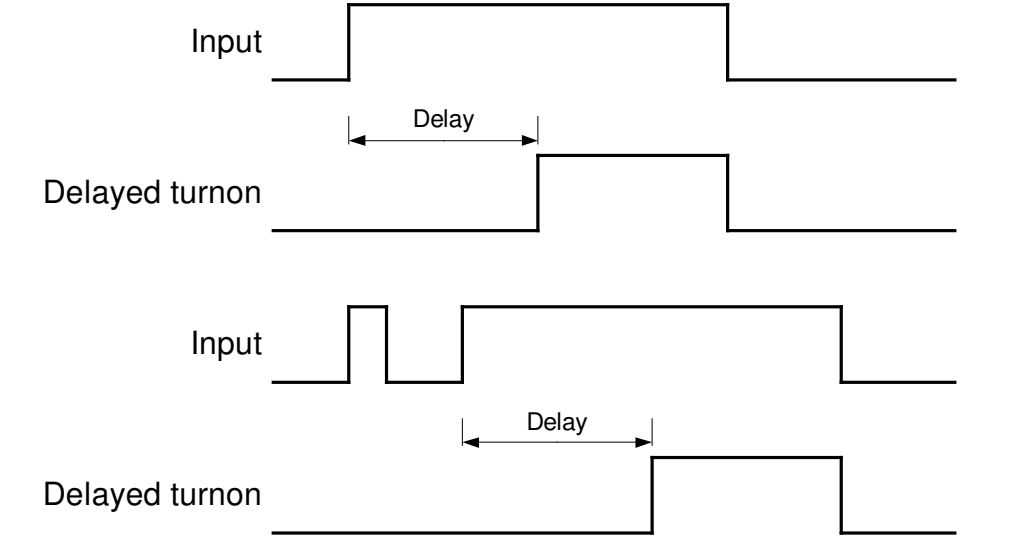
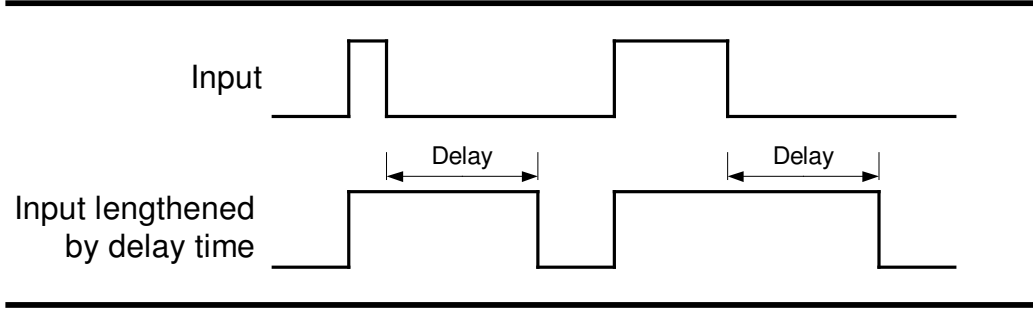
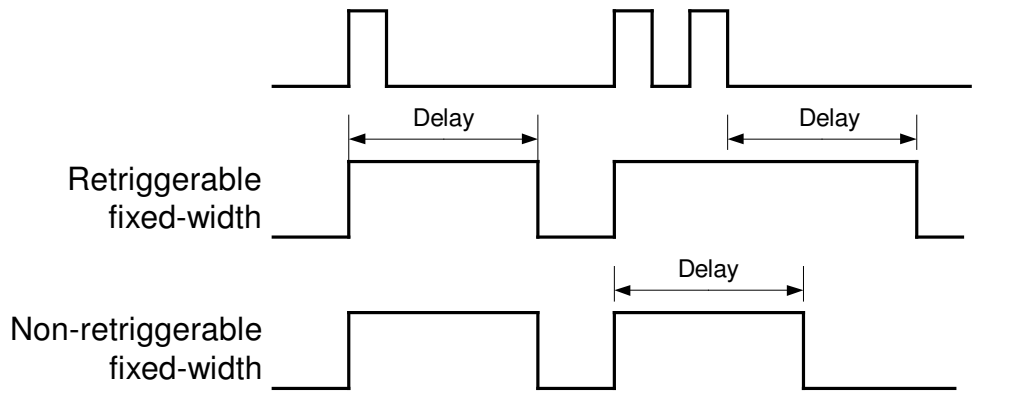
$$\text{Delay} = \text{delay time} * (\text{setpoint} - \text{source}) / (\text{setpoint} - \text{low limit})$$

Source > setpoint:

$$\text{Delay} = \text{delay time} * (\text{source} - \text{setpoint}) / (\text{high limit} - \text{setpoint})$$

Normal operation of a variable delay sets the delay to 0 when the source equals the appropriate limit and the delay is equal to the set delay when the source is at the setpoint.

The output of a variable delay turns ON when the delay time has expired after the input turns ON. Reverse operation on the input allows the selection of whether the rising or the falling edge on the input is used as a trigger.



## WebXL Logic functions

Logic functions are implemented as an array of AND/OR functions. Each function takes up to 8 inputs and combines these as a set of 8 AND functions which are all OR'd together. If one AND function is TRUE then the output of the function is TRUE. The output of the function can also be inverted if required. The inputs must be binary inputs, variables or outputs and the output can be a binary variable or output.

To define a function you have to select the inputs for the function. There can be any number of inputs from 1 to 8. The AND functions are defined in 8 columns where an input can be selected as True, False or not used. As an example look at the following function:

Inputs	AND 1	AND 2	AND 3	AND 4	AND 5	AND 6	AND 7	AND 8
A	T			F				
B	T	F						
C		T						
D	F	T						
E		T						
F								
G								
H								

This defines a logic function of the form:

Output = (A & B & /D) **OR** (/B & C & D & E) **OR** /A

Where & represents the logical AND and /X represents the complement value of X

Entries with white backgrounds are ignored. You only have to assign physical points to inputs which are used (ie where the background colour is red or green).

Logic functions can be cascaded and are evaluated from 1 to 32 so that if you feed back an output from say function 27 to function 25 any changes in the output of LF25 due to change in the output of LF27 will not be processed until the next iteration. Logic functions are evaluated once/second. In general feeding an output from a later function to an earlier one will introduce a delay while feeding forward won't. There could be cases where this introduced delay is desirable.

## WebXL Lead/Lag functions

A Lead/Lag function uses a number of inputs and drives a number of outputs. It is used to cycle the operation of plant such as chillers. Each device in the list has a control input, a fault input and an output.

If the **Fault** input is true then that piece of plant is assumed to be unavailable and is taken out of the algorithm.

The various assigned outputs are put into a list in the order entered into the list of points. When an input is turned ON the algorithm turns ON the first output in the list. When another input is turned ON the algorithm will turn ON the next output in the list. Note that there is no correlation between which input is turned ON and which output is turned ON. For example take the case of three devices, say chillers:

Input	Fault	Output
IA	FA	OA
IB	FB	OB
IC	FC	OC

A red background indicates that the point is OFF while a green background indicates that it is ON. If Input IB goes true, ie it is turned ON, then one output turns ON. Initially this will be OA as:

Input	Fault	Output
IA	FA	OA
IB	FB	OB
IC	FC	OC

Now if another input, say IC, turns on then the second output turns on as:

Input	Fault	Output
IA	FA	OA
IB	FB	OB
IC	FC	OC

If input IA also turned ON then all outputs would turn ON. If device OB is faulty then it gets taken out of the system operation and we have:

Input	Fault	Output
IA	FA	OA
IB	FB	OB
IC	FC	OC

Note that output OC has turned ON instead of output OB and that input IB is still active. If input IB now turns off we get:

Input	Fault	Output
IA	FA	OA
IB	FB	OB
IC	FC	OC

Even though input IB initially turned ON output OA turning it off turns OFF the last output. Now if all inputs turn OFF and the fault on device 2 goes away we revert back to the initial condition:

Input	Fault	Output
IA	FA	OA
IB	FB	OB
IC	FC	OC

If the input causing plant rotation now gets triggered and then input IB turns ON again we get the following situation:

Input	Fault	Output
IA	FA	OA
IB	FB	OB
IC	FC	OC

Note that now output OB has turned ON whereas before output OA turned ON. We have thus rotated which piece of plant gets turned ON as a result of an input turning ON. Each time the input causing rotation changes from OFF to ON the starting point in the list is rotated one place down. Note that it is possible to cause a rotation even with some outputs turned ON. The individual short cycle timers on binary outputs can be used to control physical plant minimum ON or OFF times.

## ***WebXL Time and holiday control***

The WebXL system provides a comprehensive set of time and holiday controls. Time and holiday schedules drive internal binary variables. This means that they can be used just like any other binary point. The great advantage of this becomes obvious when trying to do some otherwise difficult operations such as running plant over the midnight boundary. With WebXL this becomes very easy. All you need do is set a time schedule to turn on at the desired timer. Drive the output of the time schedule into a delay function which is set as a fixed delay corresponding to the time the plant should turn on. You then use the output of the delay function to control the plant.

### **Time schedules**

A typical time schedule contains a start time and a stop time plus the days of the week when the schedule should be active. The times are set in 24 hour time mode. Public holidays and other date driven controls use the Public Holiday day.

### **Time group schedules**

These allow a number of schedules to be grouped together to provide a more complex set of On and Off times during a day.

### **Holiday schedules**

Holiday schedules drive the Public Holiday day of the week in a time schedule as well as setting their own binary variable. A public holiday is a single day starting at midnight and ending at midnight.

### **Holiday From-To schedules**

These schedules provide a range of holiday dates and times. They can be set to start at a particular time on a particular day and operate until a particular time and date. There a number of variations which can be employed for further flexibility. Besides being set for a range of dates they can be set to start immediately and end on a particular time and day or they can be set to start on a particular day and then run indefinitely.

## **WebXL Alarms**

Alarms in the WebXL system can be generated by any point. Alarms send notifications by email and also can send a message to an optional headend. This allows the consolidation of viewing alarms at a system wide level. In addition, an alarm can also be configured to drive a binary output or binary variable. This could be useful for providing a local user alert or for establishing fault-tolerant control.

Each alarm has an "input" which if it is a binary value can be set to trigger an alarm either when the "input" becomes true or false. Analogue points generate an alarm by comparing the value of the point with that of another point or fixed value. Hysteresis can be added to eliminate spurious alarm states when the 2 analogue values are similar.

A binary point can also be used to **Inhibit** an alarm. For example, you could generate an alarm if the temperature in a room gets too cold but then inhibit the alarm from occurring based on a time schedule so that it doesn't fire at night or on weekends when the plant is turned off.

An alarm state has to persist for a user programmable time before an actual alarm is raised. If during this delay time the alarm goes away then the delay is reset. This can be used to prevent transitory conditions from causing an alarm. During the delay time the alarm is put into a **Pending** state.

# Sphere Systems WebXL DDC Control System

## **Overview**

The Sphere Systems WebXL represents the next generation in ddc control technology. It uses the current technologies of modern electronics, the web and XML protocols to deliver true state-of-the-art performance with power of operation, simplicity of use and ease of installation.

Unlike BACnet and LONworks systems which require complex protocols in order to configure and operate a system, the WebXL system follows the well-proven software concept of encapsulation. All functionality is contained in the WebXL controller. The controller is configured using a web interface and does not require frontend software of any sort. Controllers are connected to the standard IP network which is already present in most buildings thus doing away with special wiring and the need for special purpose devices like BACnet gateways and routers. Communication between controllers is made using XML protocol messages.

By adopting the concept of encapsulation it is possible to build controllers with very different functionality and freely interconnect and configure these on a network. This makes true interoperability a practical reality.

The concept of encapsulation is further enhanced by having sufficient physical points (ie inputs and outputs) in the controller and also by having a sufficient set of inbuilt control algorithms. This leads to greatly reduced network traffic between controllers thus limiting the use of network bandwidth.

Simplicity of system design is an important consideration and as the WebXL controller contains a large variety of inbuilt functions it is very versatile. The majority of a complete building ddc system can be configured using just 2 basic building blocks, viz the WebXL controller and WebXL I/O modules. As the I/O modules can be located a considerable distance from the controller module, wiring costs can be minimised by locating I/O modules next to the plant they control.

Ease of use of the WebXL system is further enhanced by not requiring the use of a programming language. The system is configured using functional blocks and drag and drop techniques.

## **WebXL controller module**

This is the heart of the WebXL system. It contains all of the control logic, logging memory and the web configuration interface. It communicates to the outside world via an IP connection and is configured using a standard browser. Intermodule communication is via XML protocol packets. It communicates to the I/O modules via a 76.8kbit RS485 network. It supports up to 4 I/O modules. Logging memory is battery-backed. This module also supplies regulated power for the I/O modules.

## **WebXL I/O modules**

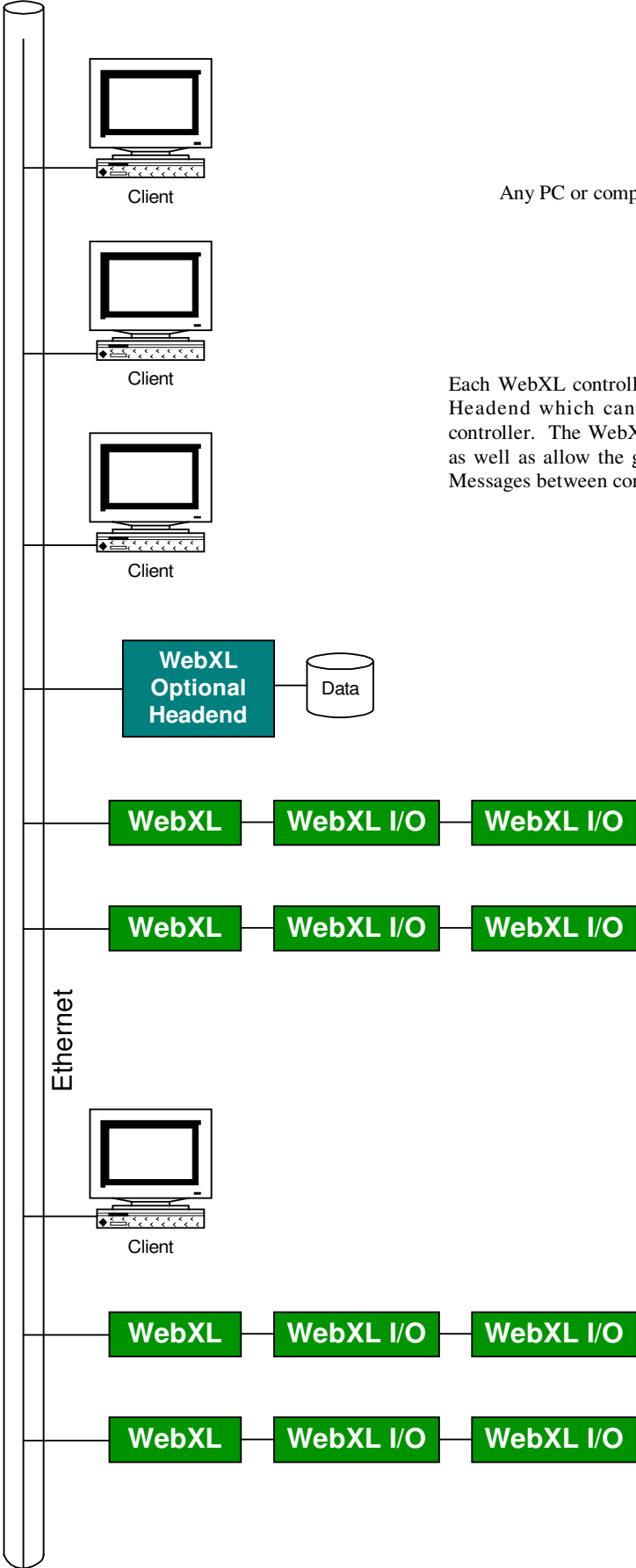
These devices provide input/output capability for the system. Each module has 8 binary inputs accepting a dry contact switching to ground and 8 binary outputs. These outputs are driven by relays rated at 3A. There are 8 analogue inputs accepting 0-10V signals with each input having a resolution of 12 bits. There are 8 analogue outputs each providing a voltage source from 0-10V with a current limit of about 20mA. Each output also has a resolution of 12 bits.

## **Typical system configuration**

The diagram below illustrates just how simple a WebXL system can be. While a small system can consist of just a few WebXL controllers, when a system gets larger it is convenient to have some way to have the web browser automatically jump to a controller without having to remember it's IP or name. The optional headend fulfils this requirement. It contains a list of all devices on the network and the drawings (dynamic graphics) on each controller. It seamlessly redirects browsers to the relevant controller. It can also have drawings of it's own which are made up of points from a number of WebXL controllers. It also acts as a central repository for all current alarms in the system.

# Sphere Systems WebXL

## System Component and Network Diagram



### The Clients

Any PC or computer which runs a web browser can be used as a client.

### The Server

Each WebXL controller acts as a web server. WebXL also supports an optional Headend which can be used to automatically redirect clients to a WebXL controller. The WebXL can also be used to store a list of all alarms in the system as well as allow the generation of drawings using points on different controllers. Messages between controllers use XML protocols.

## **WebXL controller Technical Data**

### **Power**

Operates from 18V to 26V AC or 16V to 35V DC @ 20VA depending on the number of I/O modules connected. Also can be operated from a nominal 12V DC supply.

Uses a half-wave rectifier to allow multiple controllers to be driven from the same transformer. One leg of the supply connects to earth panel ground.

Provides 12V DC power at up to 1.2A for the I/O modules.

Has an inbuilt battery charger for 12V 7AH leadacid batteries.

### **Processor and Memory**

High performance processor with flash memory for program storage, battery backed ram and non-volatile serial memory for storage of configuration and uploaded graphic files.

### **Communications**

76.8kbit RS485 link to I/O modules

10Mbit Ethernet port for configuration and communication

9600bit RS232 port for local communication and diagnostics

### **Inbuilt functions**

*Logic functions* use 8 inputs to provide an eight way AND/OR configuration.

*Delay functions* provide a variety of delays.

*Continuous (pid) loops* provide for proportional control.

*Discrete loops* provide switching of binary points using analogue values.

*Arithmetic functions* allow a variety of calculations such as maximum, average, multiply, divide and slope conversion.

*Lead/Lag functions* allow cyclic rotation of outputs with fault control.

*Time schedules* allow control of operation depending on times.

*Holiday schedules* allow definition of special conditions based on either a date or range of dates and times.

*Internal variables* are used to store outputs of functions. All time and holiday schedules drive internal binary variables so that they can be used like physical points.

*Messaging* allows the sending of information between controllers.

*Alarms (Events)* are notified by email to either a single address or a group of addresses.

*Drawings* provide graphic screens for controlling the system. They support uploaded background images and dynamic animations.

### **Logging**

All binary inputs and outputs are logged for a change of state

All analogue inputs and outputs are logged at user defined intervals

Counters on the binary inputs are logged at user defined intervals

Alarms are logged when they are generated and when they are cleared

### **Dimensions and mounting**

DIN rail mounting

### **Environmental**

Designed to operate from 0 to 60°C at 5 - 95% humidity, non-condensing.

## **WebXL I/O modules Technical Data**

Some of the functionality for the points is derived from software algorithms in the WebXL controller.  
All analogue points feature 12 bit resolution for smoother control.

### **Analogue inputs**

8 inputs each with 12 bit resolution  
Individual software trim adjustments to compensate for transducer errors  
Fully transient protected  
0-10V range.  
Can also be used as binary inputs.  
Manual override capability

### **Analogue outputs**

8 outputs each with 12 bit resolution  
0-10V range sourcing up to 20mA of current.  
Short circuit protected.  
Protected against transients.  
Slew rate control of outputs allows use for things like lighting dimmers.  
5 priority levels

### **Binary inputs**

8 inputs accept a dry contact switch closure to ground.  
Fault protected against transients  
Number of starts counter on each input  
Runtime logging on each input  
Manual override capability

### **Binary outputs**

8 outputs each having voltage-free type A relay contacts  
Relays rated to 3A resistive 30V DC or 250V AC  
Short cycle timer on every output  
Adjustable power up delay on each output  
Number of starts counter on each output  
Runtime logging on each output  
LED indicator on each output  
5 priority levels

### **Power supply**

Operates of 12V DC normally supplied by the WebXL controller.  
Power drain depends on what is connected to the outputs. With all relays activated and low power analogue outputs current drain is typically around 150mA.

### **Dimensions and mounting**

DIN rail mounting

### **Environmental**

Designed to operate in temperatures from 0 to 60°C at 5 -95% humidity, non-condensing.