Index

Programmable Logic Controller ................................................................. 5

Preferences ................................................................................................. 6
Preferences - Editor .................................................................................... 6
Preferences - I/O devices ........................................................................... 7
Preferences - Monitor .................................................................................. 8
Preferences - Plc ........................................................................................... 9
Preferences - Set Defaults and Reset Views .............................................. 10

Project Settings ......................................................................................... 11
I/O Devices .................................................................................................. 12
Add Objects ................................................................................................ 14
Object Properties ....................................................................................... 16
Display ......................................................................................................... 17
Simulation .................................................................................................... 18

Builder ........................................................................................................ 19
Error List ..................................................................................................... 20
Resource List ............................................................................................... 21

Running the Plc ......................................................................................... 22
Objects ......................................................................................................... 23

Common - MARKERTO ................................................................. 24
Common - MARKERFROM .............................................................. 25
Common - NUMN ..................................................................................... 26
Common - BIN0 ......................................................................................... 27
Common - BIN1 .......................................................................................... 28
Common - TOD ........................................................................................... 29
Common - TSM .......................................................................................... 30
Common - TMR ........................................................................................... 32
Common - TMRP ......................................................................................... 33
Common - SECPULS .................................................................................. 34
Common - ONESHOT .................................................................................. 35
Common - UPDOWN ................................................................................... 36
Common - CNTRPULS ............................................................................... 37
Common - CNTRHOUR ................................................................................ 38
Common - MEM ........................................................................................... 39
Common - MEMREGA ............................................................................... 40
Common - MEMREGD ............................................................................... 41
Common - SETPOINTA .............................................................................. 42
Common - SETPOINTD .............................................................................. 43
Common - WRITESETPOINTA ............................................................... 44
Common - WRITESETPOINTD ................................................................. 45
Common - HOLDA ....................................................................................... 46
Common - HOLDD ................................................................. 47
Common - TREND ............................................................... 48
Common - REMARK ............................................................. 49

I/O - MBINCS ................................................................. 50
I/O - MBINIS ................................................................. 51
I/O - MBINHR ................................................................. 52
I/O - MBINHRD ............................................................... 53
I/O - MBINIR ................................................................. 54
I/O - MBINIRD ............................................................... 55
I/O - MBOUTCS .............................................................. 56
I/O - MBOUTCSX ............................................................ 57
I/O - MBOUTH ................................................................ 58
I/O - MBOUTH ................................................................ 59
I/O - MBOUTH ............................................................... 60
I/O - MBOUTHDX .......................................................... 61

Logic - NOT ....................................................................... 62
Logic - AND ...................................................................... 63
Logic - QUADAND ........................................................... 64
Logic - NAND ................................................................. 65
Logic - OR ........................................................................ 66
Logic - QUADOR ............................................................. 67
Logic - NOR ...................................................................... 68
Logic - EXOR ................................................................... 69
Logic - PULS .................................................................... 70
Logic - SETRESETR ........................................................ 71
Logic - SETRESETS ........................................................ 72
Logic - BINCOUNT .......................................................... 73
Logic - SWITCHA ............................................................. 74
Logic - SWITCHD ............................................................. 75
Logic - BCDIN ................................................................. 76
Logic - BCDOUT .............................................................. 77

Compare - COMPEQ ........................................................ 78
Compare - COMPGEQ ...................................................... 79
Compare - COMPGEQ ...................................................... 80
Compare - COMPLT ........................................................ 81
Compare - COMPEQLT ..................................................... 82
Compare - COMPNEQ ...................................................... 83
Compare - COMPHL ........................................................ 84
Compare - LIMITMAX ...................................................... 85
Compare - LIMITMIN ...................................................... 86

Calculate - CALCD .......................................................... 87
Calculate - CALCSUB ...................................................... 88
Calculate - CALCMUL ...................................................... 89
<table>
<thead>
<tr>
<th>Calculate - CALCDIV</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate - CALCAVG</td>
<td>91</td>
</tr>
<tr>
<td>Calculate - CALCABS</td>
<td>92</td>
</tr>
<tr>
<td>Calculate - CALCINT</td>
<td>93</td>
</tr>
<tr>
<td>Calculate - CALCROUND</td>
<td>94</td>
</tr>
<tr>
<td>Calculate - CALCAD</td>
<td>95</td>
</tr>
<tr>
<td>Calculate - CALCDA</td>
<td>96</td>
</tr>
<tr>
<td>Calculate - CALCHS</td>
<td>97</td>
</tr>
<tr>
<td>Calculate - CALCSU</td>
<td>98</td>
</tr>
<tr>
<td>Calculate - CALCIEEEIN</td>
<td>99</td>
</tr>
<tr>
<td>Calculate - CALCIEEEOUT</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Controller - REGGAIN</th>
<th>101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller - REGONOFF</td>
<td>102</td>
</tr>
<tr>
<td>Controller - REGOFFON</td>
<td>103</td>
</tr>
<tr>
<td>Controller - REGONOFFDIFF</td>
<td>104</td>
</tr>
<tr>
<td>Controller - REGOFFONDIFF</td>
<td>105</td>
</tr>
<tr>
<td>Controller - REGCURVE</td>
<td>106</td>
</tr>
<tr>
<td>Controller - REGSEGMENT</td>
<td>107</td>
</tr>
<tr>
<td>Controller - REGPWMFIX</td>
<td>108</td>
</tr>
<tr>
<td>Controller - REGPWMVAR</td>
<td>109</td>
</tr>
<tr>
<td>Controller - REGP</td>
<td>110</td>
</tr>
</tbody>
</table>

Appendix A ................................................................. 112
Using a serial device for RS232 or RS485 via TCP/IP ........................................ 112
Check an IP-address on the network .................................................................. 115

Appendix B ................................................................. 116
Modbus information ...................................................................................... 116
Programmable Logic Controller

oPlc is a powerful program that turns your Mac into a reliable Programmable Logic Controller. There are more than 80 functional objects available.

- thousands of universal Modbus Input and Output devices available to connect...
- use oPlc at home to control electric, heating, ventilation, cooling, access security, etc...
- control your model train track or other hobby equipment...
- use oPlc for education; learn the basic functions of a logic controller...
- contains basic HVAC components...
- use oPlc on a dedicated Mac mini for professional purposes...
Preferences

You can configure oPlc for personal use.

The new settings will become active after the **Save** button is pressed.

Preferences - Editor

The workspace **Width** and **Height** are pixels and cannot exceed the range of 3000 x 2000. The default 1040 x 650 is used for the Print function A3 landscape format.

When **Start with Last Saved Plc** is checked, oPlc will automatically load the latest used Plc source after you start the program.

If you have also set the checkbox for **Start Plc in Runmode**, oPlc will load, build and run automatically. This option is used for Plc programs on a dedicated system.

**Show Object Location x y** enables the mousepointer location in pixels on the main view. It shows the location with respect of the upper left corner of the object frame.

When **Snap to grid at 5 pixels** is checked, all objects and wires are magnetic snap to a (non visible) grid.
Preferences - I/O devices

oPlc can handle I/O devices using the standard **Modbus TCP** protocol or **Modbus RTU over TCP**. For RTU over TCP you can use an universal serial to ethernet converter.

For more information about connecting network devices and use of (serial) I/O please refer to **Appendix A** of this reference manual.

![Preferences Window](image)

The **I/O Scan interval (msec)** is used to select the interval time for oPlc to read or write data from or to the ModBus interface.

The **Connect Time-out (msec)** is the maximum time oPlc tries to connect an IP-address.

The **Protocol Time-out (msec)** is the maximum time for a Modbus device to respond to a request.

If a time-out occurs, a message is displayed in the workspace.

**Outputs after Initialization** can be set to off in case of an initialization page where outputs are written.

If you like to use **Modbus Base 0** (also known as **J-bus** protocol), select this check box. In this case the first register, input or coil count starts with 0 instead of 1.
Preferences - Monitor

For monitoring (running oPlc), you can set the following settings:

**Digits (max)** - minimum 1 and maximum 10 characters displayed.

**Decimal Points** - minimum 0 and maximum 5.

**Background White** can be set if you have a lot of data fields crossing the wires.

**Display Sort by Remark** can be used if you want the datapoints on the display in an alfanumeric order, using the remark fields.
Preferences - Plc

The **Scaninterval (msec)** is the sequence time the Plc executes code. A short time allows the Plc to run faster. However, it might slow down the Mac for other applications.

When you use an Initialize page (see Project Settings) you can select the **Plc Initialization Time (msec)**. This initialization function can be used to preset memory registers to its default during startup. This value should be at least 10 times the value of the scaninterval to be sure that the Plc has a minimal of 10 scans before output processing is started.

The **Trend interval (sec)** is used for the TREND object. The default value is 600 sec (10 minutes).

If **Save Registers at Runtime** is selected, oPlc will check for data changes every minute. When data is changed, the registers are saved to disk, using the Plc filename with the extension **plist**.

When the **Tcp command Listener** is selected, oPlc can receive commands to control the Plc. This is used by the Mac App **oWatchdog**, a unique program that continuously checks network devices like routers, servers, websites and oPlc.
Preferences - Set Defaults and Reset Views

You can restore the factory defaults or reset all View settings.

After you accepts the default settings, you need to save the settings.

When you reset the views, you have to restart oPlc.
Project Settings

Used to set the Plc version number, the number of pages and the initialization page.

The **Plc Version** can be used for updates of a released project.

The **Number of Pages** can set from 1 to 10.

The **Initialization Page** can be set to any page in the range of the **Number of Pages** above. The Plc code on the initialization page is executed only during the Initialization Time (see **Preference Plc**).
I/O Devices

oPlc can handle Modbus communication with devices that are directly connected via TCP or serial RTU devices that are connected via TCP, using a serial to ethernet converter.

The number of I/O devices cannot exceed 99.

From the Object Properties the device Number is selected to connect an I/O object to an I/O device.

In this example object P28 is using DEV1, register 5.

Refer to the documentation of your device for more information about the Slave-address, IP-address, Port number and available data points.

See Appendix B for more information about the Modbus protocol.
For testing the Plc program - when there are no Modbus devices connected - you can select I/O enabled or disabled.

When I/O is set to disabled, no data is read or written from or to all Modbus devices.

You can use **Simulation** for testing the Plc program with simulation data for Modbus Inputs.
Add Objects

There are many different objects divided into the following groups:

- **Common** - marker, register, timer, counter, time schedule, setpoint
- **I/O** - ModBus read or write coil, input and holding register
- **Logic** - digital port (N)AND, (N)OR, EXOR, flipflop, switch, BCD converter
- **Compare** - greater-, equal-, less, high and low limiter
- **Calculate** - add, subtract, divide, multiply, data conversion
- **Controller** - on/off, off/on, segmentation, PWM, P-regulator

More information about the objects is described later in this manual.
You can **Drag & Drop** or **Double-Click** an object from the Object list to place it on the workspace.

Drawing a wire is simple by **Dragging** from one **Connect-point** to another.

A wire can be moved, depending on the position.
Object Properties

For each object the **Remark** property can be set.

More information about **Object Properties** are described later in this manual.
Display

The display can be used for monitoring setpoints and I/O values. You can also use the display to adjust values.

Data can be changed by double clicking on the data area. When the Plc is running, the Run led will blink.
Simulation

Simulation can be used when input data is not available or when an input module is in error.

When you set an input to **Hold**, the simulation Data will be used. This data is saved as memory register data and displayed in a blue color.

When setting an input to **Auto**, the actual data read from the Modbus device will be used.

You can set all inputs to Hold or Auto by using the buttons or you can set a single input, just by typing H, Hold, HOLD or A, Auto, AUTO in the **Status** column.
**Builder**

The builder is used to check and compile the Plc source. The builder will start automatically after you click the **Save, Build and Start** button:

The builder generates an object file with the same name as the source file, but with the .obj extension.

There are two message types: errors and warnings. An error is fatal and will not allow you to run the Plc program. A warning is only for information and the Plc program will start.

When an error occurs, you can click on the **Error list** button to get more information about the error.
Error List

This view shows you a list of errors and warnings about the object, type and pagenumber involved.

When you click an error line, the object or wire is shown in a red color.

The builder button indicates that an error occurs.
Resource List

The resource list shows information about the amount of objects, connect points and wires used.

![Resource List Window]

**Maximum allowed numbers:**

Workspace pages: 10
Objects: 400
Timers: 20
Running the Plc

When you click the **Run/Stop** button, oPlc will **Save**, **Build** and **Start** executing the program.

When the Plc is running, you can click again on the **Run/Stop** button to stop executing. It might take a few seconds before the application actually stops.
Objects

oPlc is not only a digital Plc, but also a small HVAC controller. The objects are divided into two categories: **Analog** and **Digital**.

You cannot connect an analog point to a digital point. However, there is a conversion object, called CALCAD and CALCDA, to handle this.

All inputs are called **Xn** and all outputs are called **Yn** (where n stands for the sequence number). You can connect wires from any output to an input, but you cannot connect an input to another input.

**Analog** objects are white colored. **Digital** objects are gray colored.
Common - MARKERTO

Digital or analog input X1

A special marker to connect a wire to a MARKERFROM on another page.

It is also recommended - if you use the same input on different pages - to use markers and avoid unnecessary I/O communication.
Common - MARKERFROM

Digital or analog output Y1

Marker to connect a wire from a MARKERTO on another page. You can select the associated marker in the **Object Properties** list.

Before you can connect a MARKERFROM you first have to place the associated MARKERTO object.
Common - NUMN

Analog output Y1

Generates a fixed analog value. To use variable analog values, use an **Analog Setpoint**.
Common - BIN0

Digital output Y1

Generates a fixed digital output 0. To use variable digital values, use a Digital Setpoint.

You can use a NUM0 to connect unused digital inputs.
Common - BIN1

Digital output Y1

Generates a fixed digital output 1. To use variable digital values, use a Digital Setpoint.

You can use a NUM1 to connect unused digital inputs.
Common - TOD

Analog output Y1

Generates a time or date in several formats.

The following functions can be set:

TD - Time of day 0 to 2359
TH - Time in hours 0 to 23
TM - Time in minutes 0 to 59
TS - Time in seconds 0 to 59
DS - Day of week (Sunday = 1)
DW - Day of week (Monday = 1)
DM - Day of month 1 to 31
MY - Month of year 1 to 12
YS - Year short 12 to 99
YL - Year long 2012 to 2099
Common - TSM

Analog output Y1
Analog output Y2 (clock value Monday)
Analog output Y3 (clock value Tuesday)
Analog output Y4 (clock value Wednesday)
Analog output Y5 (clock value Thursday)
Analog output Y6 (clock value Friday)
Analog output Y7 (clock value Saturday)
Analog output Y8 (clock value Sunday)
Analog output Y9 (clock value when inactive)

Time Schedules can be used to generate a setpoint on a certain time. The **day begin**, **day end** and the **setpoint** must be separated by a space character.
If you need a clock only to generate a single status, use the CALCAD object at the output and an inactive setpoint value of 0.
Common - TMR

Digital input X1
Digital output Y1

Second timer. Using a fixed time set by the Object Properties. When digital input X1 changes to 1 the timer starts. After the set time is elapsed, the digital output Y1 will become 1. When the digital input changes to 0 the output Y1 is set to 0 and the timer is reset.
Common - TMRP

Digital input X1
Analog input X2
Digital output Y1

Second timer. Using an external time.
When digital input X1 changes to 1 the timer starts. After the set time is elapsed, the digital output Y1 will become 1. When the digital input changes to 0 the output Y1 is set to 0 and the timer is reset.
Digital output Y1

Second pulse.
The digital output Y1 is set 500 msec to 1 and 500 msec to 0. This object can for example be used to generate pulses for counters and integrators.
Common - ONESHOT

Digital input X1
Digital output Y1

One shot (or monostable multivibrator) to generate a single pulse. The pulse length can be set by the Object Properties. The timer starts when digital input X1 is set to 1. At the same time digital output Y1 is set to 1. When the time is elapsed, the digital output Y1 will become 0 again.
Common - UPDOWN

Digital input X1
Digital input X2 (up or down)
Digital input X3 (reset)
Digital output Y1

Up or Down counter with external reset. When digital input X1 changes from 0 to 1 the counter will increment \((X2 = 0)\) or decrement \((X2 = 1)\).

When digital input X3 is set to 1 the Up Down counter is reset.

In this example the Up Down counter will count till 5. Then the One Shot is active for 5 seconds. This results in a count down for 5 seconds. Then it starts all over again.
Common - CNTRPULS

Digital input X1
Digital input X2 (reset)
Analog output Y1

Puls counter with external reset, using a memory register. An internal register is used to prevent data being lost when oPlc is turned off

When digital input X2 is set to 1 the puls counter is reset.

The memory register can be preset either via Object Properties or during runtime via the Display.
Digital input X1
Digital input X2 (reset)
Analog output Y1

Hour counter with external reset, using a memory register. An internal register is used to prevent data being lost when oPlc is turned off. The data on analog output Y1 = hhhh:mm:ss (hours:minutes:seconds).

When digital input X2 is set to 1 the hour counter is reset.

The memory register can be preset either via Object Properties or during runtime via the Display.

To convert clock data to a single value in seconds, the CALCHS object can be used.
Common - MEM

Analog input X1
Digital input X2 (clock data)
Analog output Y1

Memory function. The analog input X1 is clocked when the digital input X2 changes from 0 to 1. The data is not saved when oPlc is turned off.

In this example the analog output Y1 still have the previous clocked data. When the digital input X2 changes from 1 to 0 and then to 1 again, the new input data X1 is written to Y1.
Common - MEMREGA

Analog input X1
Digital input X2 (clock data)
Analog output Y1

Memory register with clock function for analog data. An internal register is used to prevent data being lost when oPlc is turned off.

The memory register can be preset either via Object Properties or during runtime via the Display.
Common - MEMREGD

Digital input X1
Digital input X2 (clock data)
Digital output Y1

Memory register with clock function for a digital status. An internal register is used to prevent data being lost when oPlc is turned off.

The memory register can be set to 1 or 0 either via Object Properties or during runtime via the Display.
Common - SETPOIN TA

Analog output Y1

Analog setpoint. An internal register is used to prevent data being lost when oPlc is turned off.

The memory register can be preset either via **Object Properties** or during runtime via the **Display**. A memory register uses the same number as the object number.
Digital output Y1

Digital setpoint. An internal register is used to prevent data being lost when oPlc is turned off.

The memory register can be set to 1 or 0 either via Object Properties or during runtime via the Display. A memory register uses the same number as the object number.
Common - WRITESETPOINTA

Analog input X1

Write analog setpoint. This object can also be used on the initialization page to preset a memory register during start-up.

Via the **Object Properties** you can select the default MRP (self) register or an existing memory register.

In this example an initialization page is used. After the initialization page is finished executing, you can change the value of the memory registers via the Display.
Common - WRITESETPOINTD

Digital input X1

Write digital setpoint. This object can also be used on the initialization page to preset a memory register to 0 or 1 during start-up.

Via the **Object Properties** you can select the default MRP (self) register or an existing memory register.

In this example an initialization page is used. After the initialization page is finished executing, you can change the status of the memory registers to 0 or 1 via the Display.
Common - HOLDA

Analog input X1
Digital input X2 (hold auto)
Analog output Y1
Analog output Y2 (simulation data)

Hold Auto for an analog value, using a memory register for simulation data. This object can be used to enable manual processing (override).

In this example the outside temperature is overridden by the value 18 when the digital setpoint MRP3 is set to 1 (hold mode).
Common - HOLDD

Digital input X1
Digital input X2 (hold auto)
Digital output Y1
Digital output Y2 (simulation data)

Hold Auto for a digital status, using a memory register for simulation data. This object can be used to enable manual processing (override).

In this example the door release input is in error. By changing the digital setpoint MRP3 to 1 (hold mode) the digital output Y1 from the HOLDAUTO object is set to 1 (the preset value of digital output Y2).
Digital or analog input X1

This object can be used for trending (data logging). You can set the trend interval via **Preferences - Plc.**

When new TREND objects are added, you may want to **Initialize new Trendfile** to set the object names to the header of the file.

The trend file is stored in the default folder of oPlc with same name as the project sourcefile but with the extension **tsv** (Tab Separated Values).

You can only open the trendfile when oPlc is not running!

When the trendfile is opened, be aware that new trends are not saved.

If **Open Trendfile** does not work, open the tsv file using the Finder to set the appropriate app to open the tsv file.
Common - REMARK

The remark field can be used to add information on a page.
I/O - MBINCS

Digital output Y1

Modbus Input Coil Status. Using Modbus function code 01.

See Appendix B for more information about the Modbus protocol.
Digital output Y1

Modbus Input Status. Using Modbus function code 02.

See Appendix B for more information about the Modbus protocol.
I/O - MBINHR

Analog output Y1

Modbus Input Holding Register 16 bits integer.
Using Modbus function code 03.

See Appendix B for more information about the Modbus protocol.
I/O - MBINHRD

Analog output Y1

Modbus Input Holding Registers 16 bits integer divided by 10. Using Modbus function code 03.

See Appendix B for more information about the Modbus protocol.
I/O - MBINIR

Analog output Y1

Modbus Input Registers 16 bits integer. Using Modbus function code 04.

See Appendix B for more information about the Modbus protocol.
I/O - MBINIRD

Analog output Y1

Modbus Input Registers 16 bits integer divided by 10. Using Modbus function code 04.

See Appendix B for more information about the Modbus protocol.
I/O - MBOUTCs

Digital input X1

Modbus Output Coil Status.
Using Modbus function code 05.

See Appendix B for more information about the Modbus protocol.
I/O - MBOOUTCSX

Digital input X1

Modbus Output Coil Status. Using Modbus function code 15 (Force Multiple Coils).

See Appendix B for more information about the Modbus protocol.
I/O - MBOUTHHR

Analog input X1

Modbus Output Holding register 16 bits integer. Using Modbus function code 06.

See Appendix B for more information about the Modbus protocol.
Analog input X1

Modbus Output Holding register 16 bits integer divided by 10. Using Modbus function code 06.

See Appendix B for more information about the Modbus protocol.
**I/O - MBOUTHXR**

Analog input X1

Modbus Output Holding register 16 bits integer. Using Modbus function code 16 (Preset Multiple Registers).

*See Appendix B for more information about the Modbus protocol.*
I/O - MBOUTHRDX

Analog input X1

Modbus Output Holding register 16 bits integer divided by 10. Using Modbus function code 16 (Preset Multiple Registers).

See Appendix B for more information about the Modbus protocol.
Logic - NOT

Digital input X1
Digital output Y1

When input X1 = 0, output Y1 = 1 else output Y1 = 0.

<table>
<thead>
<tr>
<th>X1</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Logic - AND

Digital input X1
Digital input X2
Digital output Y1

When input X1 and X2 are both 1, output Y1 = 1 else output Y1 = 0.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Logic - QUADAND

Digital input X1
Digital input X2
Digital input X3
Digital input X4
Digital output Y1

When all the inputs X1, X2, X3 and X4 are 1, output Y1 = 1 else output Y1 = 0.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Logic - NAND

Digital input X1
Digital input X2
Digital output Y1

When input X1 and X2 are both 1, output Y1 = 0 else output Y1 = 1.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Logic - OR

Digital input X1
Digital input X2
Digital output Y1

When input X1 or X2 = 1, output Y1 = 1 else output Y1 = 0.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Logic - QUADOR

Digital input X1
Digital input X2
Digital input X3
Digital input X4
Digital output Y1

When input X1, X2, X3 or X2 = 1, output Y1 = 1 else output Y1 = 0.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Logic - NOR

Digital input X1
Digital input X2
Digital output Y1

When input X1 or X2 = 1, output Y1 = 0 else output Y1 = 1.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Logic - EXOR

Digital input X1
Digital input X2
Digital output Y1

When both inputs X1 and X2 are 0 or both inputs X1 and X2 are 1, output Y1 = 0 else output Y1 = 1.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Logic - PULS

Digital input X1
Digital output Y1

When input X1 changes from 0 to 1, output Y1 changes 1 <-> 0 or 0 <-> 1.

<table>
<thead>
<tr>
<th>X1</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>^1</td>
<td>1</td>
</tr>
<tr>
<td>^0</td>
<td>1</td>
</tr>
<tr>
<td>^1</td>
<td>0</td>
</tr>
<tr>
<td>^0</td>
<td>0</td>
</tr>
</tbody>
</table>

X1 input pulse
Y1 divided by 2
Logic - SETRESETR

Digital input X1 (set)
Digital input X2 (reset)
Digital output Y1
Digital output Y2

Set Reset flipflop with reset dominant (ruling).

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y1</th>
<th>Y2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Rd</td>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>^1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>^0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Logic - SETRESETS

Digital input X1 (set)
Digital input X2 (reset)
Digital output Y1
Digital output Y2

Set Reset flipflop with set dominant (ruling).

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y1</th>
<th>Y2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sd</td>
<td>R</td>
<td>Q</td>
<td>_Q</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>^1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>^0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Logic - BINCOUNT

Digital input X1
Digital output Y1
Digital output Y2

Binary count.

<table>
<thead>
<tr>
<th>count</th>
<th>X1</th>
<th>Y1</th>
<th>Y2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>^=1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>^=0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>^=1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>^=0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>^=1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>^=0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>^=1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>^=0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Logic - SWITCHA

Analog input X1
Analog input X2
Digital input X3
Analog output Y1

Analog switch.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>a2</td>
<td>0</td>
<td>a1</td>
</tr>
<tr>
<td>a1</td>
<td>a2</td>
<td>1</td>
<td>a2</td>
</tr>
</tbody>
</table>
Logic - SWITCHD

Digital input X1
Digital input X2
Digital input X3
Digital output Y1

Digital switch.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1</td>
<td>b2</td>
<td>0</td>
<td>b1</td>
</tr>
<tr>
<td>b1</td>
<td>b2</td>
<td>1</td>
<td>b2</td>
</tr>
</tbody>
</table>
Digital input X1
Digital input X2
Digital input X3
Digital input X4
Analog output Y1

BCD to Analog converter. Output data range analog 0 - 15.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>Y1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
</tbody>
</table>

Example of using two BCDIN converters to generate an analog output (data 0 - 255) from 8 digital inputs.
Logic - BCDOUT

Analog input X1
Digital output Y1
Digital output Y2
Digital output Y3
Digital output Y4

Analog to BCD converter.

<table>
<thead>
<tr>
<th>X1</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The input data X1 is converted to INT and then logical AND with 15 (0x0F).

Example of using two BCDOUT converters to generate 8 digital outputs from an analog input (data 0 - 255).
**Compare - COMPEQ**

Analog input X1 (A)
Analog input X2 (B)
Digital output Y1

Compare analog data X1 (A) with analog data X2 (B).
If A is equal to B output Y1 = 1 else output Y1 = 0.
Compare - COMPGT

Analog input X1 (A)
Analog input X2 (B)
Digital output Y1

Compare analog data X1 (A) with analog data X2 (B).
If A is greater than B output Y1 = 1 else output Y1 = 0.
Compare - COMPGTEQ

Analog input X1 (A)
Analog input X2 (B)
Digital output Y1

Compare analog data X1 (A) with analog data X2 (B).
If A is greater than or equal to B output Y1 = 1 else output Y1 = 0.
Compare - COMPLT

Analog input X1 (A)
Analog input X2 (B)
Digital output Y1

Compare analog data X1 (A) with analog data X2 (B).
If A is less than B output Y1 = 1 else output Y1 = 0.
Compare - COMPEQLT

Analog input X1 (A)
Analog input X2 (B)
Digital output Y1

Compare analog data X1 (A) with analog data X2 (B).
If A is equal or less than B output Y1 = 1 else output Y1 = 0.
Compare - COMPNEQ

Analog input X1 (A)
Analog input X2 (B)
Digital output Y1

Compare analog data X1 (A) with analog data X2 (B).
If A is not equal to B output Y1 = 1 else output Y1 = 0.
### Compare - COMPHL

Analog input X1 (A)
Analog input X2 (B)
Analog input X3 (C)
Analog output Y1 (H)
Analog output Y2 (L)

Compare analog data X1 (A), analog data X2 (B) and analog data X3 (B) and set the Y1 to the highest and Y2 to the lowest value.
Compare - LIMITMAX

Analog input X1
Analog output Y1

The output data on Y1 is high (max) limited to the property setting.
Compare - LIMITMIN

Analog input X1
Analog output Y1

The output data on Y1 is low (minimal) limited to the property setting.
Calculate - CALCADD

Analog input X1 (A)
Analog input X2 (B)
Analog output Y1

Add.
The output data on Y1 is the analog data X1 (A) + X2 (B).
Calculate - CALCSUB

Analog input X1 (A)
Analog input X2 (B)
Analog output Y1

Subtract.
The output data on Y1 is the analog data X1 (A) - X2 (B).
Calculate - CALCMUL

Analog input X1 (A)
Analog input X2 (B)
Analog output Y1

Multiply.
The output data on Y1 is the analog data X1 (A) x X2 (B).
Calculate - CALCDIV

Analog input X1 (A)
Analog input X2 (B)
Analog output Y1

Divide.
The output data on Y1 is the analog data X1 (A) / X2 (B).
Calculate - CALCAVG

Analog input X1 (A)
Analog input X2 (B)
Analog output Y1

Average value.
The output data on Y1 is the average value of data X1 (A) and X2 (B).

Example:
If X1 = 15.8 and X2 = 35.2 then Y1 = 25.5.
Calculate - CALCABS

Analog input X1
Analog output Y1

Absolute value.
The output data on Y1 is the absolute value of data X1.

Example:
If X1 = -23 then Y1 = 23.
Calculate - CALCINT

Analog input X1
Analog output Y1

Integer value.
The output data on Y1 is the integer value of data X1.

Example:
If X1 = 32.6 then Y1 = 32.
Calculate - CALCROUND

Analog input X1
Analog output Y1

Rounded value.
The output data on Y1 is the rounded value of data X1.

Example:
If X1 = 12.5 then Y1 = 13.
Calculate - CALCAD

Analog input X1
Digital output Y1

Conversion analog to digital.
If the data on X1 is greater than or equal to 1.0 then the output Y1 = 1 else Y1 = 0.
Calculate - CALCDA

Digital input X1
Analog output Y1

Conversion digital to analog.
If the data on X1 = 1 the output Y1 = 1.0 else Y1 = 0.0.
Calculate - CALCHS

Analog input X1
Analog output Y1

Conversion from clock data in hh:mm:ss to seconds.

Example:
Calculate - CALCSU

Analog input X1
Analog output Y1

Conversion from 16 bits signed integer to 16 bits unsigned integer.

Example:
If X1 = -10 then Y1 = 32777.
Calculate - CALCIEEEEIN

Analog input X1 IEEE 754 Floating point
Analog output Y1 Register n
Analog output Y2 Register n+1

Conversion of IEEE 754 Floating point (32 bits) to 2 integer values (16 bits registers). This object can be used to convert a single precision (floating point) to 2 Modbus registers. The registers must be in sequence n & n+1.

Use the CALCIEEEEOUT to convert 2 Modbus registers to a floating point.

Some Modbus equipment used n+1 & n addressing (registers are swapped).
Calculate - CALCIEEEOUT

Analog input X1 Register n
Analog input X2 Register n+1
Analog output X1 IEEE 754 Floating point

Conversion of 2 integer values (16 bits registers) to IEEE 754 Floating point (32 bits). This object can be used to convert 2 Modbus registers to a single precision (floating point). The registers must be in sequence n & n+1.

Use the CALCIEEEIN to convert a floating point to 2 Modbus registers.

Some Modbus equipment used n+1 & n addressing (registers are swapped).
Controller - REGGAIN

Analog input X1
Analog input X2 (gain)
Analog input X3 (offset)
Analog output Y1

Linear gain control with offset.

Formula:
\[ Y1 = (X1 \times X2) + X3 \]
Controller - REGONOFF

Analog input X1
Analog input X2 (off)
Analog input X3 (on)
Digital output Y1

On Off regulator.
If X1 greater than or equal to X3 the output Y1 is set to 1. If X1 equal to or less than X2 the output Y1 is set to 0.

Example:
Controller - REGOFFON

Analog input X1
Analog input X2 (on)
Analog input X3 (off)
Digital output Y1

Off On regulator. If X1 equal to or less than X2 the output Y1 is set to 1. If X1 greater than or equal to X3 then the output Y1 is set to 0.

Example:
Controller - REGONOFFDIFF

Analog input X1
Analog input X2 (extern setpoint)
Digital output Y1
Analog output Y2 (reference X1 - Y5)
Analog output Y3 (differential)
Analog output Y4 (local setpoint)
Analog output Y5 (working setpoint Y2 + Y4)

On off differential regulator with local and remote setpoint.

If X1 is greater than or equal to Y5 then the output Y1 is set to 1.
If X1 gets a value less than Y5 - Y3, then the output Y1 is set to 0.

Example:

Select a register to change a setting.
Controller - REGOFFONDIF

Analog input X1
Analog input X2 (extern setpoint)
Digital output Y1
Analog output Y2 (reference X1 - Y5)
Analog output Y3 (differential)
Analog output Y4 (local setpoint)
Analog output Y5 (working setpoint X2 + Y4)

Off On differential regulator with local and remote setpoint.

If X1 is greater than or equal to Y5 + Y3 then the output Y1 is set to 0.
If X1 gets a value less than Y5, then the output Y1 is set to 1.

Example:
Controller - REGCURVE

Analog input X1
Analog output Y1
Analog output Y2 (base)
Analog output Y3 (parallel offset)
Analog output Y4 (angle)

Curve. Generates a flow temperature, using +20 as a the minimum flow.

Formula:
\[ Y1 = Y2 + ((20 - X1) \times Y4) + Y3 \]

Example (using defaults \( Y2 = 30 \), \( Y3 = 0 \) and \( Y4 = 2 \)):
If X1 is +20 the analog output Y1 is set to 30.0.
If X1 is -10 the analog output Y1 is set to 90.0.
Controller - REGSEGMENT

Analog input X1
Analog output Y1
Analog output Y2 (Input X1)
Analog output Y3 (Input X2)
Analog output Y4 (Input X3)
Analog output Y5 (Input X4)
Analog output Y6 (Output Y1)
Analog output Y7 (Output Y2)
Analog output Y8 (Output Y3)
Analog output Y9 (Output Y4)

Four segment generator to create a single analog output value.

Example:
If X1 is +20 the analog output Y1 is set to 30.0.
If X1 is +12.5 the analog output Y1 is set to 55.0.
If X1 is -10 the analog output Y1 is set to 90.0.
Controller - REGPWMFIX

Analog input X1
Digital output Y1
Analog output Y2 (cycle time)
Analog output Y3 (pulse width)
Analog output Y4 (reference time)

Puls Width Modulator with a fixed puls length. The minimal pulwidth is 1 second. When the analog input X1 is greater than 50%, the pulses are divided until 100% is reached (digital output Y1 = 1 continuous).

The reference time at Y4 shows the remaining time until the next pulse (on or off).

Example:
Cycle time = 60 sec. Pulswidth = 1 sec.
An analog value on X1 = 10% results in 6 pulses per 60 seconds.
When the analog value on X1 = 50% it will result in 30 pulses per 60 seconds (1 second on, 1 second off). If the input value is 60%, the pulswidth become 2 seconds and the interval time will be divided to get 60% on and 40% off.
Controller - REGPWMVAR

Puls Width Modulator with a variable puls length. The minimal pulswidth is 1 second.

The reference time at Y4 shows the remaining time untill the next puls (on or off).

Example:
Cycle time = 60 sec. Pulswidth = 1 sec.
An analog value on X1 = 10% results in puls (Y1 = 1) for 6 seconds and wait (Y1 = 0) for 54 second. When the analog value on X1 = 50% it will result in a pulse (Y1 = 1) of 30 seconds and wait (Y1 = 0) for 30 seconds.
Controller - REGP

Analog input X1
Analog input X2 (external setpoint)
Digital input X3 (controller disabled)
Analog output Y1
Analog output Y2 (function)
Analog output Y3 (local setpoint)
Analog output Y4 (minimum output)
Analog output Y5 (maximum output)
Analog output Y6 (proportional range)
Analog output Y7 (bias)
Analog output Y8 (working setpoint)

Proportional regulator.
Using digital input X3 the controller can be reset and disabled.

Formula:
\[ WSP (Y8) = \text{external setpoint (X2) + local setpoint (Y3)} \]

The function setting on Y2 can have the following value:
1 - Direct (deviation = X1 - WSP / control range 0 to max)
2 - Reverse (deviation = WSP - X1 / control range 0 to max)
3 - Direct (deviation = X1 - WSP / control range min to max)
4 - Reverse (deviation = WSP - X1 / control range min to max)

Function 1 and 2:
\[ \text{Analog output Y1} = \frac{(\text{deviation} \times Y5)}{Y6 + Y7} \]

Function 3 and 4:
\[ \text{Analog output Y1} = \frac{(\text{deviation} \times (Y5 - Y4))}{Y6 + Y7} \]
Appendix A

Using a serial device for RS232 or RS485 via TCP/IP

World wide there are several hardware interfaces available to connect an analog modem or other serial device using RS232 or RS485 to your Mac, iPhone, iPod or iPad.

For example: the NPort 5230 from Moxa Technologies Co., Ltd.

Be sure that your local IP-address is in the same range as the NPort 5230.

The NPort 5230 has two serial ports.

Port 1 is used for RS232 and Port 2 is used for RS422/485.

In this example the local IP-address is set to 192.168.94.33 and the IP-address of the NPort 5230 is set to 192.168.94.254 (default 192.168.127.254).
To configure the NPort 5230 for Port 2 (RS422/485), open the NPort Web Console with a Webbrowser (e.g. Safari).

Set the Operation mode for Port 2 (RS422/485) to **TCP server mode** and set the **Local TCP** port to the port number you want to use in your application.
The Serial Settings depend on the configuration settings of your device. In this example the interface is set for **RS485 2 wire**.
Check an IP-address on the network

You can use the **Ping** function of the Network utility to check if you have access to the IP-address.

Go to the Ping tab.

You can also use the Terminal to check if you have access to the IP-address.

```
imac:~ user$ ping 192.168.94.254
PING 192.168.94.254 (192.168.94.254): 56 data bytes
64 bytes from 192.168.94.254: icmp_seq=0 ttl=64 time=0.941 ms
64 bytes from 192.168.94.254: icmp_seq=1 ttl=64 time=2.025 ms
64 bytes from 192.168.94.254: icmp_seq=2 ttl=64 time=1.331 ms
^C
Use ctrl + C to abort
--- 192.168.94.254 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.941/1.432/2.025/0.448 ms
imuac:~ user$
```
Appendix B

Modbus information

This overview shows the two protocols used by oPlc; Modbus TCP and Modbus RTU over TCP.

Modbus RTU over TCP is the same as Modbus RTU (RS485), but for communication you can use a standard ethernet to serial converter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction ID</td>
<td>2 bytes</td>
<td>For synchronization between messages of server &amp; Client</td>
</tr>
<tr>
<td>Protocol ID</td>
<td>2 bytes</td>
<td>Zero for Modbus TCP</td>
</tr>
<tr>
<td>Length</td>
<td>2 bytes</td>
<td>Number of remaining bytes in this frame</td>
</tr>
<tr>
<td>Unit ID</td>
<td>1 byte</td>
<td>Unit address (sometimes optional)</td>
</tr>
<tr>
<td>FCode</td>
<td>1 byte</td>
<td>Function code</td>
</tr>
<tr>
<td>Data</td>
<td>n bytes</td>
<td>Data as response or commands</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>3.5c idle</td>
<td>Minimal 3-1/2 character times of silence</td>
</tr>
<tr>
<td>Slave ID</td>
<td>8 bits</td>
<td>Slave address</td>
</tr>
<tr>
<td>FCode</td>
<td>8 bits</td>
<td>Function code</td>
</tr>
<tr>
<td>Data</td>
<td>n bytes</td>
<td>Data as response or commands</td>
</tr>
<tr>
<td>CRC</td>
<td>16 bits</td>
<td>Error checks</td>
</tr>
<tr>
<td>End</td>
<td>3.5c idle</td>
<td>Minimal 3-1/2 character times of silence between frames</td>
</tr>
</tbody>
</table>
Read Coil Status (01)

Request
Byte 0: FCode = 01
Byte 1-2: Starting coil
Byte 3-4: Number of coils

Response
Byte 0: FCode = 01
Byte 1: Byte count of response (B=(bit count+7)/8)
Byte 2-(B+1): Bit values (least significant bit is first coil!)

Exceptions
Byte 0: FCode = 81 (hex)
Byte 1: Exception code = 01 or 02

Example
Read 1 coil at reference 0 (00001 in Modicon 984) resulting in value 1
01 00 00 00 01 => 01 01 01

Read Input Status (02)

Request
Byte 0: FCode = 02
Byte 1-2: Starting input
Byte 3-4: Number of inputs

Response
Byte 0: FCode = 02
Byte 1: Byte count of response (B = (bit count + 7) / 8)
Byte 2 - (B + 1): Bit values (least significant bit is first coil!)

Exceptions
Byte 0: FCode = 82 (hex)
Byte 1: Exception code = 01 or 02

Example
Read 1 discrete input at reference 0 (10001 in Modicon 984) resulting in value 1
02 00 00 00 01 => 02 01 01
Read Holding Registers (03)

Request
Byte 0: FCode = 03
Byte 1-2: Starting register
Byte 3-4: Number of registers

Response
Byte 0: FCode = 03
Byte 1: Byte count of response (B = 2 x number of registers)
Byte 2 - (B + 1): Register values

Exceptions
Byte 0: FCode = 83 (hex)
Byte 1: Exception code = 01 or 02

Example
Read 1 register at reference 0 (40001 in Modicon 984) resulting in value 1234 hex
03 00 00 00 01 => 03 02 12 34

Read Input Registers (04)

Request
Byte 0: FCode = 04
Byte 1-2: Starting register
Byte 3-4: Number of registers

Response
Byte 0: FCode = 04
Byte 1: Byte count of response (B = 2 x number of registers)
Byte 2 - (B + 1): Register values

Exceptions
Byte 0: FCode = 84 (hex)
Byte 1: Exception code = 01 or 02

Example
Read 1 input register at reference 0 (30001 in Modicon 984) resulting in value 1234 hex
04 00 00 00 01 => 04 02 12 34
**Force Single Coil (05)**

*Request*
- Byte 0: FCode = 05
- Byte 1-2: Coil
- Byte 3: = FF to turn coil ON, =00 to turn coil OFF
- Byte 4: = 00

*Response*
- Byte 0: FCode = 05
- Byte 1-2: Coil
- Byte 3: = FF to turn coil ON, =00 to turn coil OFF (echoed)
- Byte 4: = 00

*Exceptions*
- Byte 0: FCode = 85 (hex)
- Byte 1: Exception code = 01 or 02

*Example*
Write 1 coil at reference 0 (00001 in Modicon 984) to the value 1
05 00 00 FF 00 => 05 00 00 FF 00

**Preset Single Register (06)**

*Request*
- Byte 0: FCode = 06
- Byte 1-2: Register
- Byte 3-4: Register value

*Response*
- Byte 0: FCode = 06
- Byte 1-2: Register
- Byte 3-4: Register value

*Exceptions*
- Byte 0: FCode = 86 (hex)
- Byte 1: Exception code = 01 or 02

*Example*
Write 1 register at reference 0 (40001 in Modicon 984) of value 1234 hex
06 00 00 12 34 => 06 00 00 12 34
**Force Multiple Coils (15)** - used by oPlc to force a single coil

Request
Byte 0: FCode = 0F (hex)
Byte 1-2: Starting coil
Byte 3-4: Number of coils
Byte 5: Byte count \( B = (\text{bit count} + 7) / 8 \)
Byte 6 - (B + 5): Data to be written (least significant bit = first coil)

Response
Byte 0: FCode = 0F (hex)
Byte 1-2: Starting coil
Byte 3-4: Bit count

Exceptions
Byte 0: FCode = 8F (hex)
Byte 1: Exception code = 01 or 02

Example
Write 3 coils at reference 0 (00001 in Modicon 984) to values 0,0,1
0F 00 00 00 03 01 04 => 0F 00 00 00 03

**Preset Multiple Registers (16)** - used by oPlc to preset a single register

Request
Byte 0: FCode = 10 (hex)
Byte 1-2: Starting register
Byte 3-4: Number of registers
Byte 5: Byte count \( B = 2 \times \text{number of registers} \)
Byte 6 - (B + 5): Register values

Response
Byte 0: FCode = 10 (hex)
Byte 1-2: Starting registers
Byte 3-4: Number of registers

Exceptions
Byte 0: FCode = 90 (hex)
Byte 1: Exception code = 01 or 02

Example
Write 1 register at reference 0 (40001 in Modicon 984) of value 1234 hex
10 00 00 00 01 02 12 34 => 10 00 00 00 01
Exception codes

All exceptions are signaled by adding 0x80 to the function code of the request, and following this byte by a single reason byte for example as follows:

03 12 34 00 01 => 83 02

Request read 1 register at index 0x1234 response exception type 2 - illegal data address.

Here are some of the most common exception codes:

01 ILLEGAL FUNCTION

The function code received in the query is not an allowable action for the slave. This may be because the function code is only applicable to newer controllers, and was not implemented in the unit selected. It could also indicate that the slave is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.

02 ILLEGAL DATA ADDRESS

The data address received in the query is not an allowable address for the slave. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed, a request with offset 96 and length 5 will generate exception 02.

03 ILLEGAL DATA VALUE

A value contained in the query data field is not an allowable value for the slave. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.

04 ILLEGAL RESPONSE LENGTH

Indicates that the request as framed would generate a response whose size exceeds the available MODBUS data size. Used only by functions generating a multi-part response, such as functions 20 and 21 (not used by oPlc).