# WebControl$^{TM}$ Programmable Logic Controller User Guide

|  |  |
|---|---|
| Version: | 3.2.23 |
| Hardware Version: | 2.2.2 |
| Firmware Version: | 3.02.23 |
| Firmware datastamp: | 9/8/2015 |
| Doc last modified: | 9/8/2015 |

## Table of Contents

## Table of Figures

# 1 Introduction

This document provides an overview of the technical aspects of using WebControl$^{TM}$ PLC. Version 3.02.xy firmare is for WebControl 8 hardware rev 2.2.2 board. It describes the inputs and outputs offered by the PLC version of the WebControl$^{TM}$ hardware and firmware. The WebControl PLC function, usage and syntax as well as many examples are provided there to help you get started. The PLC version of firmware provides greater flexibility in I/O control; but in return, expects user to learn programming concepts and write an assembly like PLC program. A PLC program has the ability to read write and compare values of the available inputs, outputs, variables and timers. With a PLC program loaded and running, WebControl$^{TM}$ PLC can operate on its own, without a network connection. The PLC programming module and programming guide are provided for a learn-on-your own experience. Assistance in writing or debugging PLC code is not provided as part of the regular technical support for WebControl$^{TM}$ PLC configuration. WebControl$^{TM}$ PLC 3-02-18d firmware does not replace 3-02-18b firmware. 3-02-18b supports DS2438 sensor, 3-0218d removed DS2438 support, but added I2C and SPI bus.

## 1.1 Scope

The scope of this document is to be a guide for configuring and using the features provided by WebControl$^{TM}$. The reader is expected to be technically competent in all the technical areas within this document, and is strongly advised to use this document alongside the other reference material listed in the reference section.

## 1.2 Table of Definitions

The following table is a list of definitions used though out the document.

| Definition | Description |
|---|---|
| HTTP | Hypertext transfer protocol |
| DNS | Domain name server |
| SMTP | Simple mail transport protocol |
| SNTP | Simple network time protocol |
| 1-wire | Special bidirectional serial data bus from Maxim |
| RH | Relative humidity |
| NetBios | Human readable name used as an alternative to an IP address for accessing the server on a network. E.g. http://WebControlTM |
| IP | Internet protocol |
| DHCP | Dynamic host configuration protocol |
| ROM | Read only memory |
| PLC | Programmable Logic Controller |

## 2 WebControl™ I/O

The current hardware version of WebControl™ is equipped with a number of inputs and outputs; these are shown in below in figure 1.0.

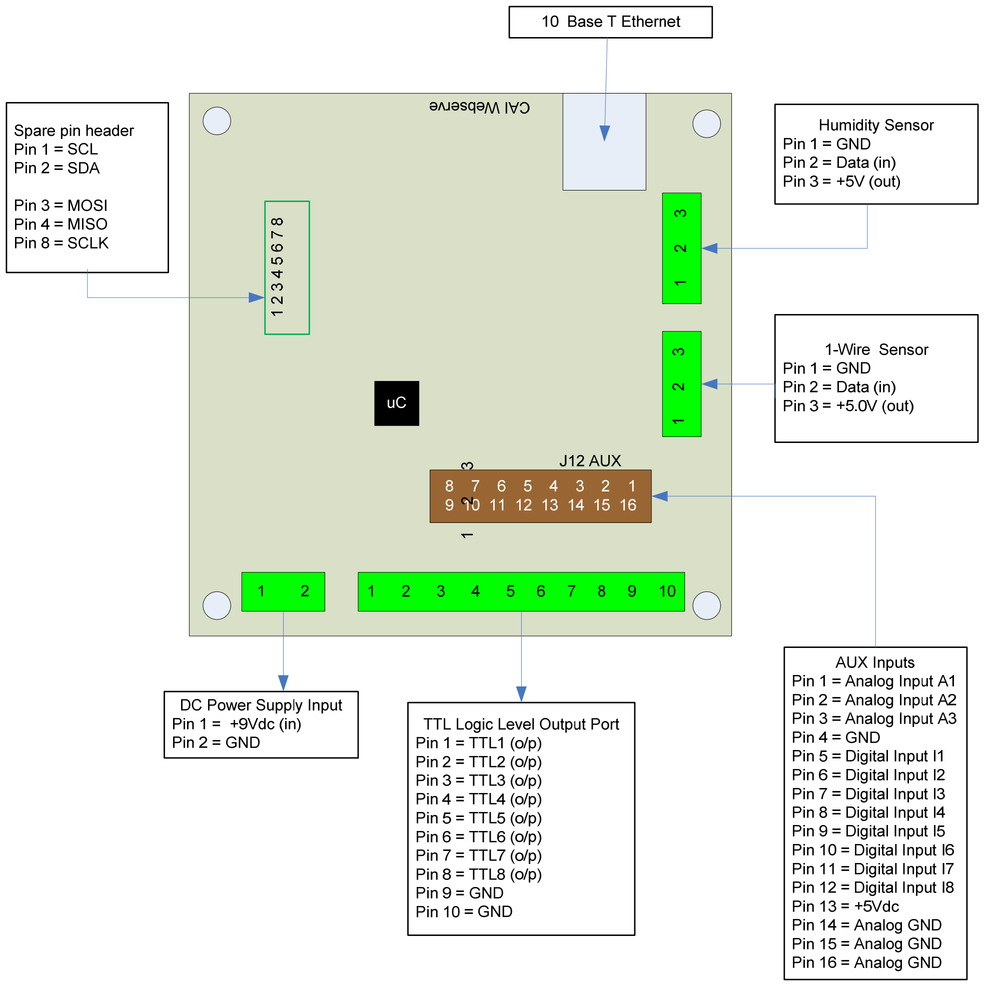


*Figure 1.0 WebControl™ PCB inputs and outputs diagram*

### 2.1 Power Supply Inputs

The DC power supply input is the electrical supply to the board. The recommended input voltage is 9V DC with 2A rating. Any voltage greater than 12V applied to this input may require adding heatsink to regulator, otherwise it may overheat.

### 2.2 TTL Output Port

The TTL logic level output port has 8 TTL outputs that can each be set or cleared using web GUI command codes, or by setting up a PLC using the available inputs, variables, and timers, so that the particular output is set when a particular condition is met. The maximum current that can be sourced or sinked by one of these outputs at a time is 20mA or 100mA for the whole board. TTL 0 level defined as 0-2V and TTL 1 level is define 3-5V with TTL 0 almost 0V and TTL 1 almost 5V depending on the load. TTL output current is capable to drive standard solid state relay, but will not be able to drive those coil based relays. Be aware of the current requirements of the circuits/loads attached to these outputs. Damage will occur to the board if too much current is allowed.

## 2.3 AUX Input Port

The AUX input connector offers digital and analog inputs that can be monitored or programmed to set a single TTL output or several TTL outputs. Three analog inputs are available, each have an input range of 0 to 10Vdc. Eight digital inputs are available and use TTL input levels (0 and 5Vdc). The AUX Input Port pin out diagram (labeled as J12AUX in Figure 1) shows the layout of this port in more detail. The matching connector for this port is TYCO ELECTRONICS - **1658622-3** .

## 2.4 1-Wire Sensor Input

The 1-wire sensor port allows up to eight Maxim DS1822/DS18B20/DS18S20 1-wire temperature sensors to be connected. It also allows one Maxim DS2417 real time clock, and one DS2450 4 channel A/D converter. When DS2438 sensors are used, it will take upper four DS18xy sensor location for temperature reading, and DS2450's analog inputs. The temperature and analog input can be displayed via the HTTP browser and can be used as inputs to the PLC engine to control the TTL output port. Since hardware revision 2.2.2 uses 5V to power 1-wire bus, there is no need for any external power to the sensors. Please note if more than one DS2450 is attached to 1-wire bus, only the first one will be used by the firmware. If more than 10 1-wire devices attached to the bus, only the first 10 will be used.

## 2.5 Humidity Sensor Input

The current WebControl$^{TM}$ hardware has been designed to use the Honeywell 4000 series relative humidity sensor devices. Using any other humidity sensor without careful consideration may result in malfunction of this feature. The RH sensor can be used to display relative humidity via HTTP or be used to control the TTL outputs as an input in the PLC engine.

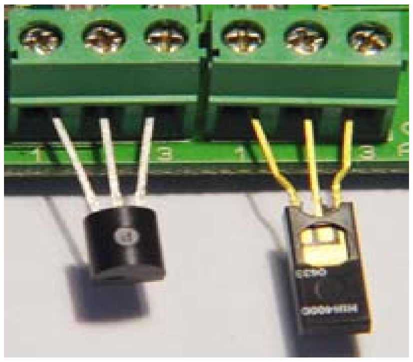


*Figure 2.0 WebControl$^{TM}$ sensor connections*

**Please make sure the temp sensor and humidity sensor connected similar to this picture. Reverse the polarity may cause damage to the board.**

## 2.6 Spare Pin Header

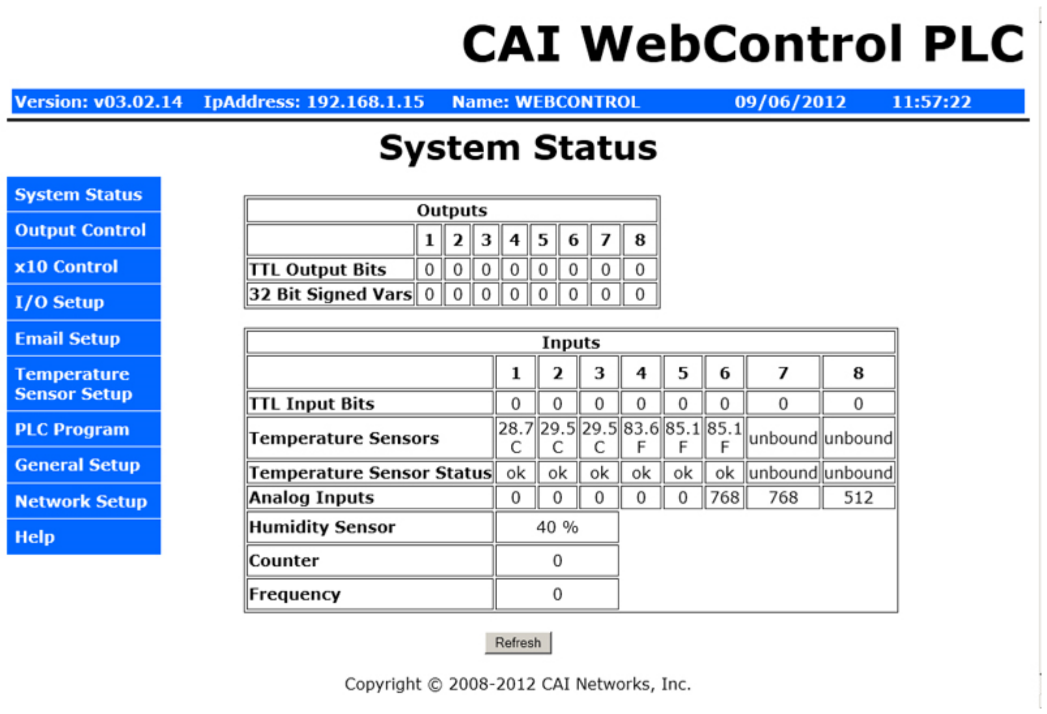
There are 8 pins on this header. Pin 1 and pin 2 are I2C bus pins, they are pulled up to 3.3V by 10K resistors. Pin 3, pin4, and pin8 are SPI bus pins. Pin 3 and 4 are pulled up by 10K resistors, Pin 8 is not pulled up nor pulled down. All these I2C and SPI pins are 5V tolerate pins, meaning they can work both with 3.3V or 5V bus. In real world, most I2C sensors are 5V, and most SPI sensors are 3.3V.

## 2.7 Real Time Clock Support

Through I2C bus, WebControl 8 board supports RTC chip DS1307 I2C module. With battery on the DS1307 module, WC8 board can maintain accurate clock. This support is transparent to user without any PLC coding. However, to make sure there is no I2C bus conflict, it will count internally when START bit is inserted on I2C bus. If for any reason, PLC commands will not complete the rest I2C logic, it must insert STOP bit on the I2C bus, so that RTC can be executed inside kernel.

# 3 WebControl$^{TM}$ PLC Web GUI Configuration

To access WebControl$^{TM}$ PLC, connect WebControl$^{TM}$ to the local network and power supply. The default IP address will be 192.168.1.15. Be aware that once you enable DHCP in your WebControl network settings page, you will need to check your DHCP server log for which IP address DHCP assigned to the board. You need to know and understand the network you attaching this hardware to in order to know where to look for DHCP server logs. For home networks, DHCP serving is commonly enabled at the router. Each WebControl$^{TM}$ PLC board has a unique MAC address. Look in the DHCP log to find the matching MAC address, then you can find the IP address DHCP assigned to the hardware. From a computer connected to the same network, start a web browser, and point to that IP address: for example: http://ww.xx.yy.zz when the web page loads, you will be prompt for login. The default user ID and password is: "**admin/password**", all in lower case. You will see this page once logged in:



CAI WebControl PLC

Version: v03.02.14 IpAddress: 192.168.1.15 Name: WEBCONTROL 09/06/2012 11:57:22

System Status

Outputs

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| TTL Output Bits | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32 Bit Signed Vars | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Inputs

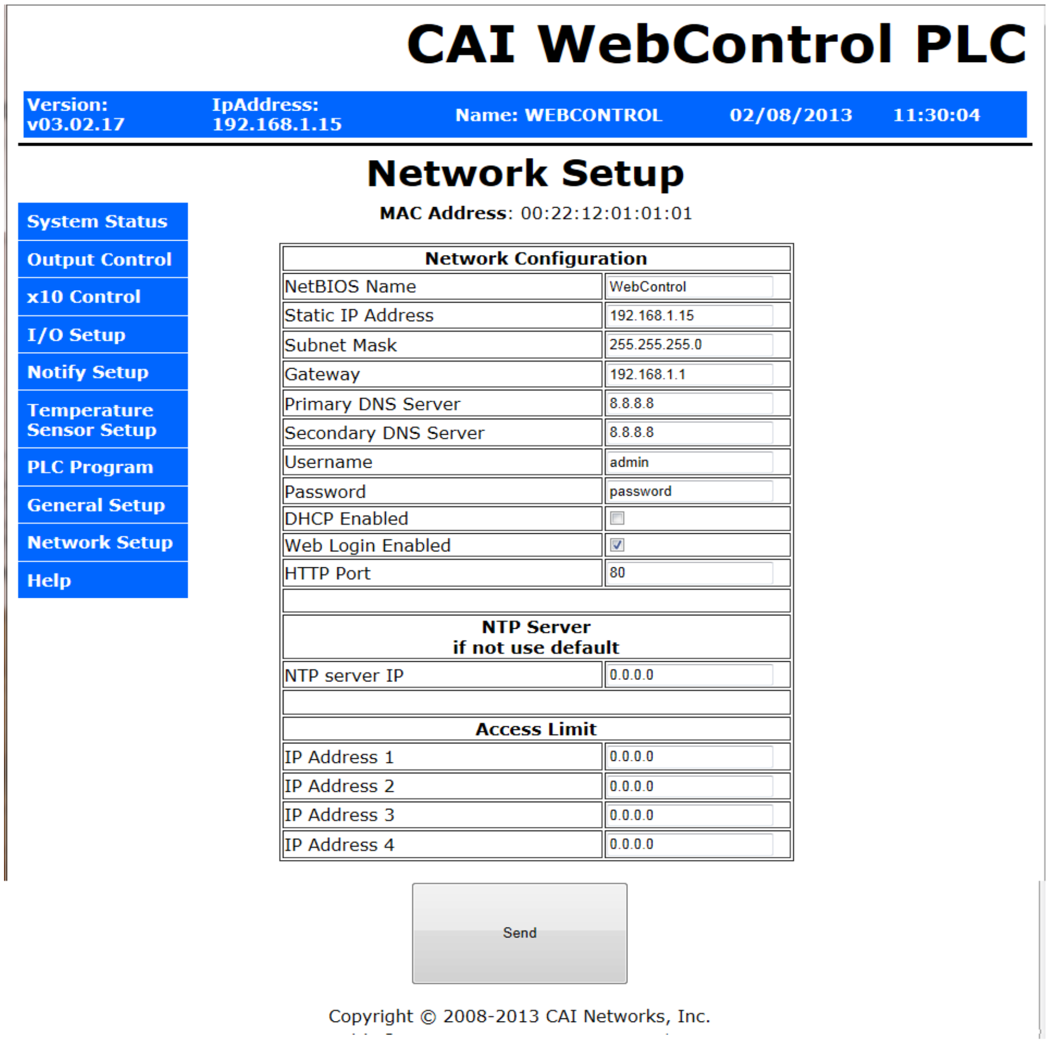
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| TTL Input Bits | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Temperature Sensors | 28.7 C | 29.5 C | 29.5 C | 83.6 F | 85.1 F | 85.1 F | unbound | unbound |
| Temperature Sensor Status | ok | ok | ok | ok | ok | ok | unbound | unbound |
| Analog Inputs | 0 | 0 | 0 | 0 | 0 | 768 | 768 | 512 |
| Humidity Sensor | 40 % | | | | | | | |
| Counter | 0 | | | | | | | |
| Frequency | 0 | | | | | | | |

Copyright © 2008-2012 CAI Networks, Inc.

*Figure 3.0 WebControl$^{TM}$ PLC system status*

## 3.1 Network Settings

These are the basic settings that need to be configured in order for WebControl$^{TM}$ to work successfully on a network. It may be necessary to connect the board directly to a PC and access it using its default IP address before connecting directly to a live network in order to configure it correctly. Figure 4.0 shows the Network settings WebControl$^{TM}$ has.



Network Setup

MAC Address: 00:22:12:01:01:01

| Network Configuration | |
|---|---|
| NetBIOS Name | WebControl |
| Static IP Address | 192.168.1.15 |
| Subnet Mask | 255.255.255.0 |
| Gateway | 192.168.1.1 |
| Primary DNS Server | 8.8.8.8 |
| Secondary DNS Server | 8.8.8.8 |
| Username | admin |
| Password | password |
| DHCP Enabled | ☐ |
| Web Login Enabled | ☑ |
| HTTP Port | 80 |
| | |
| NTP Server if not use default | |
| NTP server IP | 0.0.0.0 |
| | |
| Access Limit | |
| IP Address 1 | 0.0.0.0 |
| IP Address 2 | 0.0.0.0 |
| IP Address 3 | 0.0.0.0 |
| IP Address 4 | 0.0.0.0 |

*Figure 4.0 WebControl$^{TM}$ PLC Network configurable*

### 3.1.1 NTP Server Settings

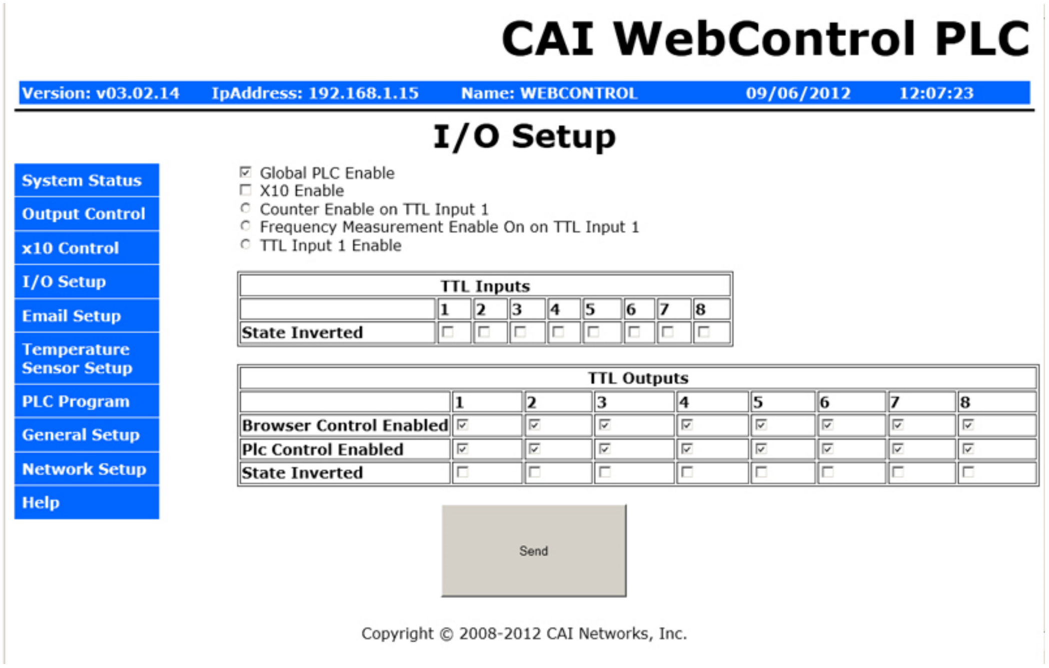
This version changed from previous versions by reduced Allowed Limits host IP address from eight to four. In addition, it added NTP Server section to allow user to enter his local NTP server's IP address to overwrite the default pool.ntp.org resolved IP address. By default NTP server field should be 0.0.0.0, so that WebControl$^{TM}$ will use pool.ntp.org as the Network Time Protocol server.

### 3.1.2 Access Settings

For security purpose, you can decide your own user name and password. For being used over Internet, you may also set the access list, so that only host in the list can access the WebControl$^{TM}$ board over the network.  Please note that if any entry is 0.0.0.0, all IP addresses can access it.

## 3.2 I/O Setup

WebControl$^{TM}$ PLC allows users to directly from browser control each output, or through the PLC logic to control the TTL output. For flexibility, user can decide if the TTL input or out to be inverted between logic 0 and logic 1. To use PLC logic, first user must check the “Global PLC enable” to start the PLC engine. Then for each TTL output, user can decide if the PLC control will be applicable.



CAI WebControl PLC

Version: v03.02.14 IpAddress: 192.168.1.15 Name: WEBCONTROL 09/06/2012 12:07:23

I/O Setup

- ✓ Global PLC Enable
- ☐ X10 Enable
- ○ Counter Enable on TTL Input 1
- ○ Frequency Measurement Enable On on TTL Input 1
- ○ TTL Input 1 Enable

| TTL Inputs | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| State Inverted | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |

| TTL Outputs | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Browser Control Enabled | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Plc Control Enabled | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| State Inverted | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |

Copyright © 2008-2012 CAI Networks, Inc.

*Figure 5.0 I/O setup*

WebControl$^{TM}$ PLC allows user using browser to directly control the output state. If a command line utility or third party control software used, you will need to enable the Browser Control for that TTL output, also may need to go to network configuration to disable the Web login (see figure 4.0).

The inverted state will be remembered in EEPROM even after power loss. The invert state will be restored upon power resumed.

X10 Enable will allow user to send X10RF commands through TTL port7 and TTL port8. Those two ports will not be able to perform any other function, once X10 is enabled.

Counter Enable /Frequency Measurement Enable/TTL Input Enable are radio buttons that you can only select one of them. Those functions sharing the same TTL1 input. Please note the Counter is triggered at rising edge after the falling edge. If your counting source TTL level was zero from start with, you may miss count by 1. Frequency measurement is up to 2MHz.

## 3.3 X10 RF Control

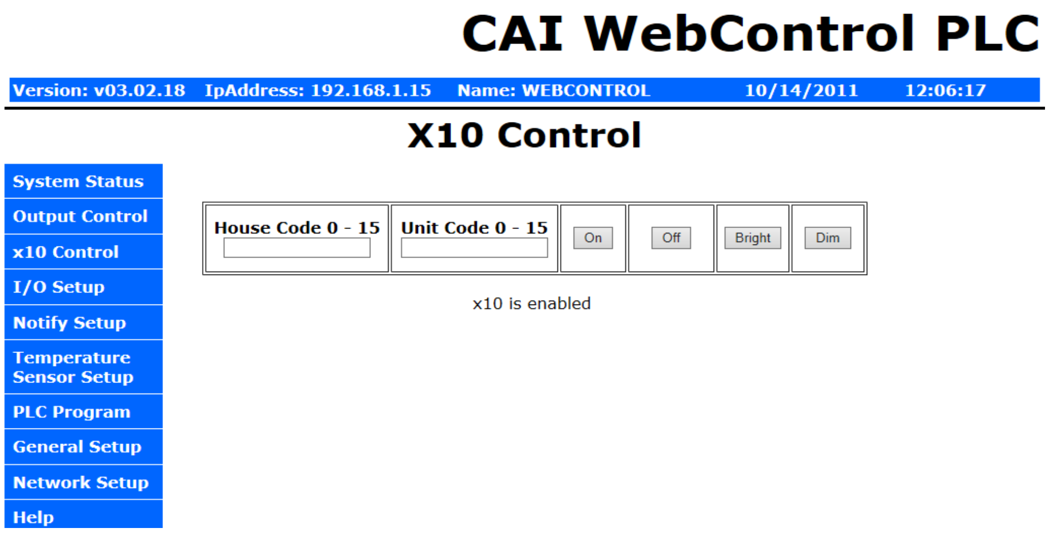


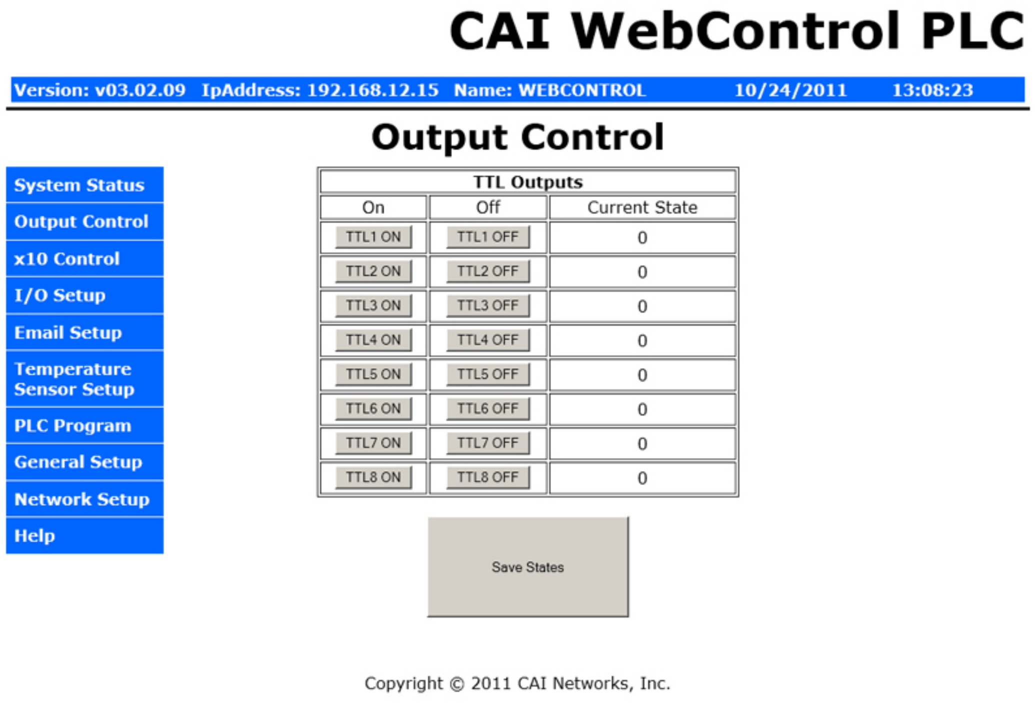
*Figure 6.0 WebControl$^{TM}$ X10RF Control*

WebControl uses TTL7 and TTL8 to control X10RF devices. User must already enabled X10RF function in the I/O Setup screen. An external RF transmitter must be attached to the output port TTL7, TTL8, and ground. In the USA, the X10 RF frequency is 310MHz, and in Europe, the frequency is 433 MHz  WebControl can send all 16 house codes and 16 unit codes, controlling total 256 devices.

X10 RF function is supported in the PLC programming also.
In PLC programming, PLC command is in the format:
X10   H   U   C
Where H is house code from 0-15, U is unit code, also from 0-15; C is one of following four commands: 0 – Off, 1 – On, 2 – Bright, 3 – Dim.  H,U,C can be variables so that user can dynamically change the X10 operation in PLC logic.

If the X10 function enabled, TTL7 and TTL8 should not be used for controlling other devices. Please note PLC logic uses 0-15 or 0-0xF for code, older web GUI only use 0-F.

## 3.4 Output Control


CAI WebControl PLC

Version: v03.02.09 IpAddress: 192.168.12.15 Name: WEBCONTROL 10/24/2011 13:08:23

Output Control

TTL Outputs

| On | Off | Current State |
|---|---|---|
| TTL1 ON | TTL1 OFF | 0 |
| TTL2 ON | TTL2 OFF | 0 |
| TTL3 ON | TTL3 OFF | 0 |
| TTL4 ON | TTL4 OFF | 0 |
| TTL5 ON | TTL5 OFF | 0 |
| TTL6 ON | TTL6 OFF | 0 |
| TTL7 ON | TTL7 OFF | 0 |
| TTL8 ON | TTL8 OFF | 0 |

Copyright © 2011 CAI Networks, Inc.

*Figure 6.0 WebControl$^{TM}$ Output Control*

When click on each TTL output On or Off, the current state of the TTL output will change. However, the display may or may not update depending on the “Web pulling” enabled in the General Setup screen. If you did not enable the “Web Pulling” to save the bandwidth, you will need to use browser refresh to see the output state change. Please note “Save States” button will make WebControl store the state in the EEPROM. When power lost and reapplied, the output will remember the states being saved.

## 3.5 Notification Setup

WebControl$^{TM}$ allows up to 8 different email notifications to be send from the PLC program. Please note the port can be any number, but WebControl current hardware cannot support SSL enabled email. Certain email servers like Gmail requires use SSL enabled email client. WebControl 8 does not support that. However, WebControl does allow sending email out from port other than 25. Most ISP provides the non-SSL email server for its own customers, on specified port, for example gmx provide on port 587. Please contact your local ISP first to find out how to sending out non-SSL encrypted email. For example, GMX email can be used:

SMTP Host = **mail.gmx.net** Port 587
username = your-GMX-email-account,
password = your password used
From field = Your gmx mail

From firmware 3.02.16 and newer firmware, user can specify different email address for different email notification. So that different email can be send to different person for a different event.

WebControl 3.02.16 and above firmware also allow HTTP GET to be used to set the VAR or output state to another WebControl board, provided the other board does not enable logic. If the security is concern, please set up allowed host in Network Setup screen. This HTTP GET feature can also be used to send notification to the remote HTTP server, given the remote server does have CGI logic to process the HTTP GET sent from WebControl. User can specify up to 8 different URLs on each board. There are three fields in each URL. The first field is IP address and port, in the format of 192.168.2.15:81 . The ":" is to separate the IP address and remote HTTP port. There is no space. It is followed by virtual host name. The URI portion of the URL is a string leading all the way to the value being set on the WebControl or remote HTTP server. The last part of the URL is the Base64 encrypted use name and password. This will allow one WebControl to set value in another WebControl with password protected. If the other side does not require login, this field can be left blank.

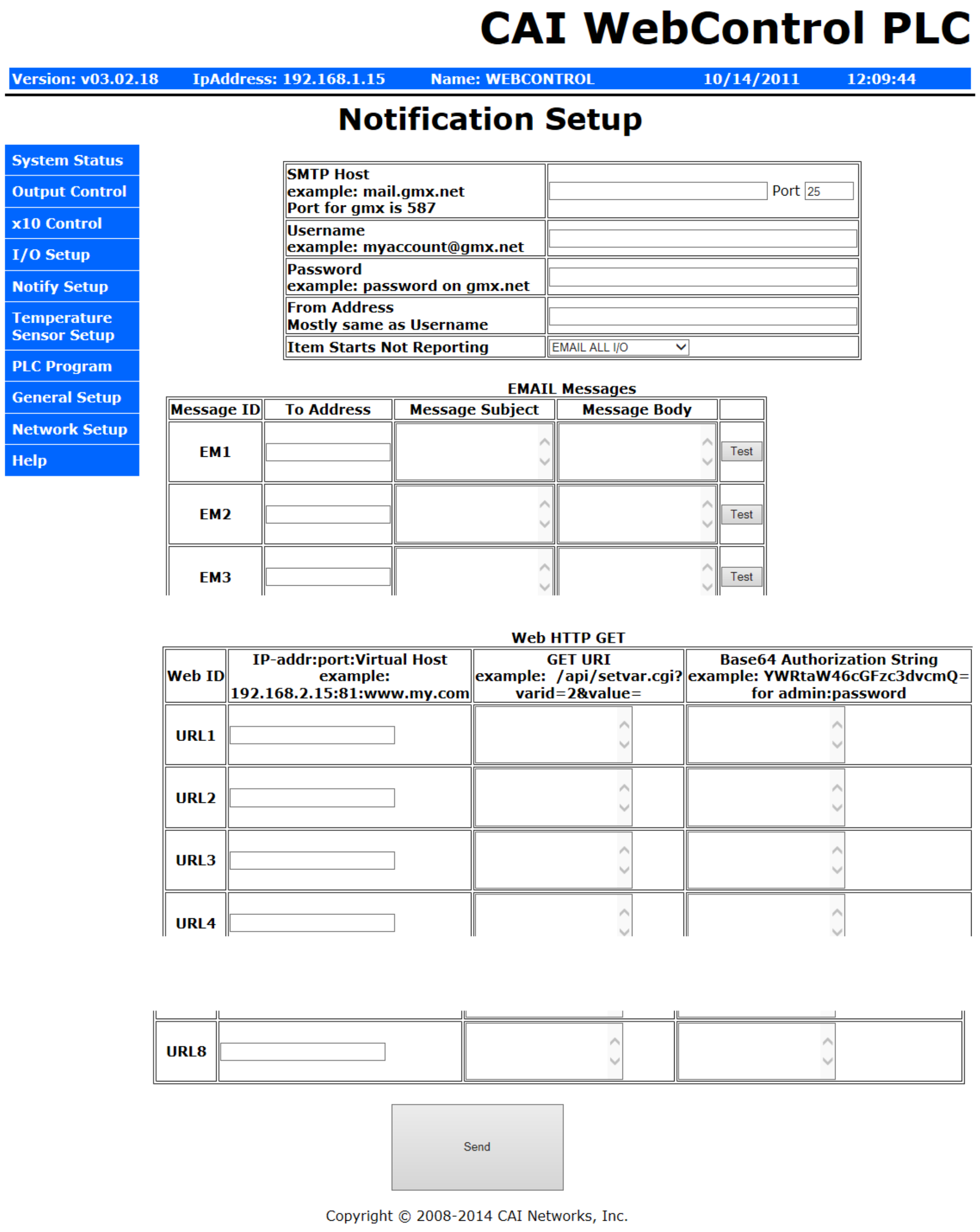
For example, if you want to set a remote WebControl board VAR1 to value 1234, you simple in the PLC logic calling:

```
WEBSET   URL1   1234
```

Or

WEBSET URL2 VAR1

Please note WEBSET has sending queue of four. If there are WEBSET requests coming more than the queue can hold, the older requests will get lost. In addition, if the remote WebControl or server does not respond, the WEBSET will fail.



| | |
|---|---|
| SMTP Host<br>example: mail.gmx.net<br>Port for gmx is 587 | Port 25 |
| Username<br>example: myaccount@gmx.net | |
| Password<br>example: password on gmx.net | |
| From Address<br>Mostly same as Username | |
| Item Starts Not Reporting | EMAIL ALL I/O |

EMAIL Messages

| Message ID | To Address | Message Subject | Message Body | |
|---|---|---|---|---|
| EM1 | | | | Test |
| EM2 | | | | Test |
| EM3 | | | | Test |

Web HTTP GET

| Web ID | IP-addr:port:Virtual Host example: 192.168.2.15:81:www.my.com | GET URI example: /api/setvar.cgi? varid=2&value= | Base64 Authorization String example: YWRtaW46cGFzc3dvcmQ= for admin:password |
|---|---|---|---|
| URL1 | | | |
| URL2 | | | |
| URL3 | | | |
| URL4 | | | |
| URL8 | | | |

*Figure 7.0 WebControl$^{TM}$ email setup*

Starting from the 3.2.15 firmware, we allow user to configure less I/O being report. By default, all I/O status is reported in the email.

Please note that although there are 8 EMx and 8 URLy, WebControl internally only has queue for up to four emails and eight URLs queued for sending. If PLC logic made such a call too fast, the newer email or WEBSET notification will overwrite the older ones in the queue.

Most email notification problems are either the configuration problem or the SMTP host rejected email. In this version PLC firmware, user can send a test email to see if the email notification working or not. In earlier version user guide, it was not clearly stated the email message ID is in format of EM1 through EM8. Using number only or other string may send out wrong email message.

WEBSET feature allows setting VAR value in another WebControl board, or setting value on your web server, if you have proper CGI code developed to take. For example, to set a value in ISY from Universal Devices, you can configure the URL1 to:
IP_address**:**port**:**virtual_host for example: 192.168.2.222**:**80**:**www.mydomain.com
Please note ISY99 using port 80
URI example: /rest/vars/set/1/15/
Base64 encrypted username:password: Example: YWRtaW46cGFzc3dvcmQg
Between username : and password, there is no space. In earlier version, you need to specify “AUTHBASIC:” from 3.02.17 firmware, the third field is dedicate for the encrypted user ID and password, so that “AUTHBASIC:” can be omitted.

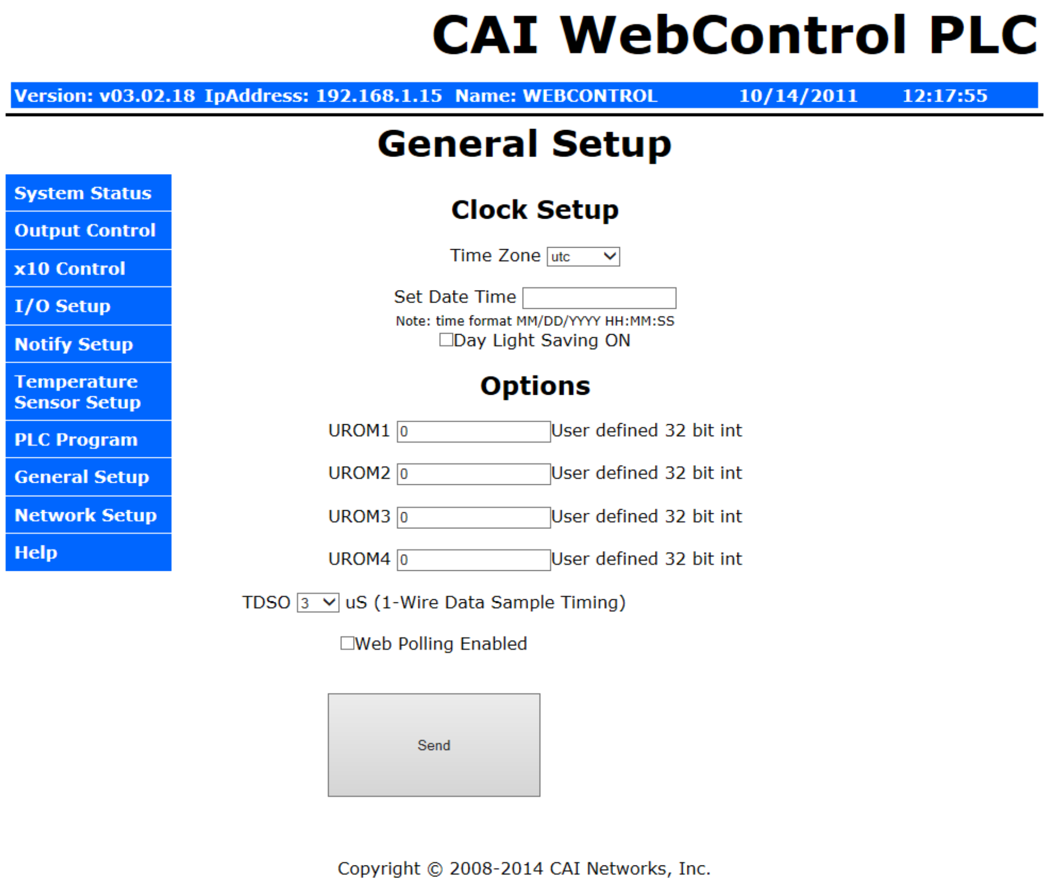
In your PLC logic, you can put this command:
WEBSET URL1 VAR1
Then it will set ISY variable 15 to the value of VAR1.

To do base64 encryption, you can go to our web site at:
http://www.cainetworks.com/support/base64-encoding.html
From 3.02.17 firmware, server CGI code can return a 32bit signed number to WebControl during WEBSET call. Please refer to the chapter 6 example 15 for how to write the CGI code and how to refer to this returned value.

To obtain support for email or WEBSET feature, full TCP capture data is required. The captured data must be able to fully display in WireShark software.

## 3.6 General Setup

When WebControl™ is connected to the network it will obtain NTP time from Internet. User need to set correct time zone on WebControl™ PLC.


CAI WebControl PLC

Version: v03.02.18 IpAddress: 192.168.1.15 Name: WEBCONTROL 10/14/2011 12:17:55

General Setup

Clock Setup

Time Zone utc

Set Date Time

Note: time format MM/DD/YYYY HH:MM:SS

☐Day Light Saving ON

Options

| | | |
|---|---|---|
| UROM1 | 0 | User defined 32 bit int |
| UROM2 | 0 | User defined 32 bit int |
| UROM3 | 0 | User defined 32 bit int |
| UROM4 | 0 | User defined 32 bit int |

TDSO 3 uS (1-Wire Data Sample Timing)

☐Web Polling Enabled

Copyright © 2008-2014 CAI Networks, Inc.

*Figure 8.0 Network defaults*

If the WebControl cannot reach to pool.ntp.org over Internet, it will use its own built-in clock with less accuracy. If a DS2417 real time clock chip connected to 1-wire bus, WebControl will query it for the time. User may change and update the clock from this screen by typing MM/DD/YY HH:MM:SS format in the Set Date Time input field, then click “Send” button. If a DS2417 chip is discovered on 1-wire bus, the real time clock will be set to this time. When synced with NTP server, real time clock chip DS2417 is also set every 10 minutes.

UROM1…UROM4 are four identifiers can be used in PLC program. They can help user to fine tune their PLC program for different setup. These four values are stored in the EEPROM, so that when power lost, its value will be kept by the EEPROM. When power restored, UROM value will remain whatever they set before. There are two ways to set UROM value. One is to set it on this web GUI page by save modified value. Another way to set their value is to use /api/seturom.cgi call. Please refer to the FAQ section for that method.

For 1-wire devices on a complex bus configuration, its timing is critical. Time Data Sample Offset (TDSO) field is for user making his own timing adjustment based on his setup. Default TDSO is 3uS. Most user will be fine with default. For shorter bus, it could reduce to 1uS or lower. For longer bus, 30uS or longer may be desirable. 1-wire signals are travelled in CAT5 cable at speed about 20Meter/uS. However, we recommend to add filter resistor to reduce bounced signals from stub wire and far end of the bus, instead of change the timing.

When Web polling enabled, WebControl$^{TM}$ PLC building active Java code will constantly update the browser display for temperature, humidity, and I/O status. Please note with such a update, it may take significant amount bandwidth from your network, as well as WebControl’s processor power.

## 3.7 Reset Setup

If the configuration was totally mess up, user can reset the board to factory default configuration by shorting the RESET jumper while powering up the WebControl$^{TM}$. Please note reset will wipe out the entire configuration, including the PLC program. Please make sure backup your PLC program before reset. After reset, the login ID and password will be restored to “**admin/password**”. Default IP address is 192.168.1.15 with DHCP disabled.

# 4 AUX System Inputs

The system AUX inputs of WebControl$^{TM}$ are used as inputs to the PLC engine (see section 5.1). This section describes the connector ladled as J12. Digital and analog inputs come through J12 connector.

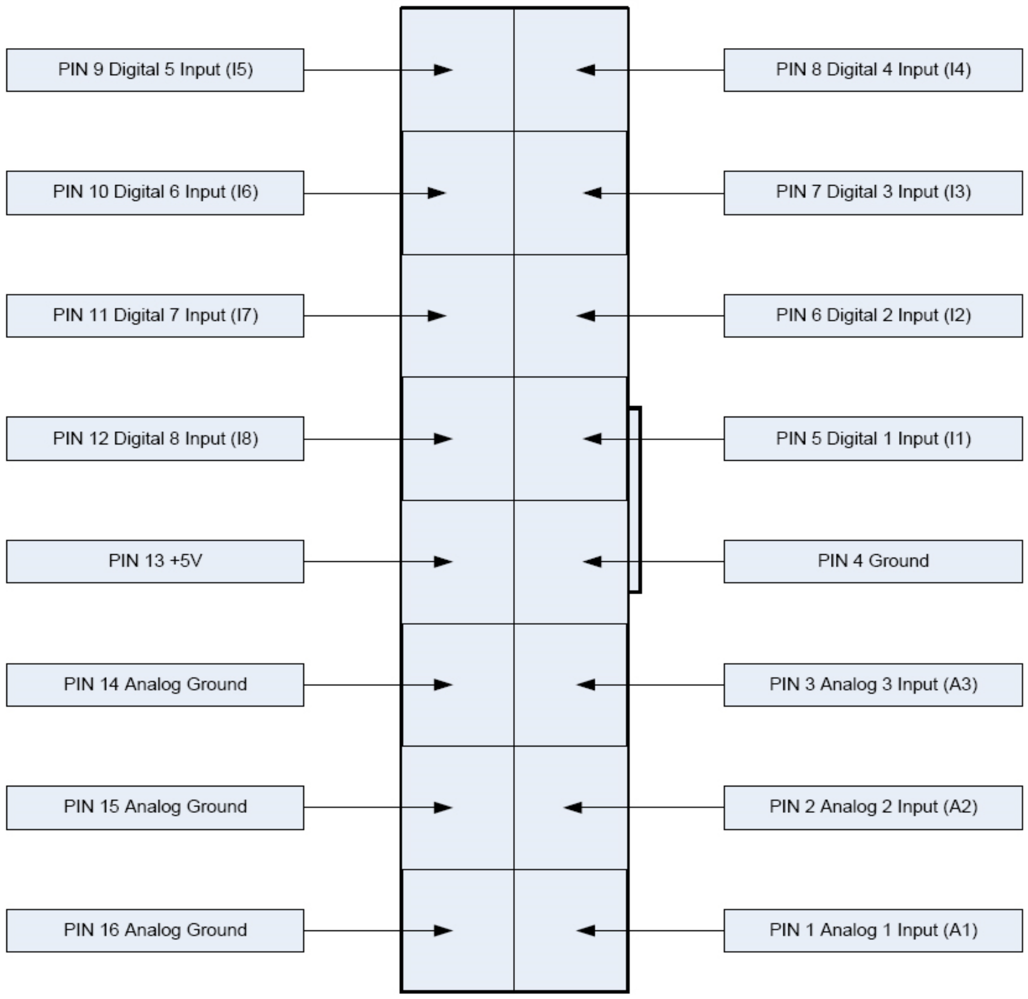

PIN 9 Digital 5 Input (I5)
PIN 8 Digital 4 Input (I4)
PIN 10 Digital 6 Input (I6)
PIN 7 Digital 3 Input (I3)
PIN 11 Digital 7 Input (I7)
PIN 6 Digital 2 Input (I2)
PIN 12 Digital 8 Input (I8)
PIN 5 Digital 1 Input (I1)
PIN 13 +5V
PIN 4 Ground
PIN 14 Analog Ground
PIN 3 Analog 3 Input (A3)
PIN 15 Analog Ground
PIN 2 Analog 2 Input (A2)
PIN 16 Analog Ground
PIN 1 Analog 1 Input (A1)

*Figure 9.0 WebControl$^{TM}$ J12 Input pins*

## 4.1 Digital Inputs

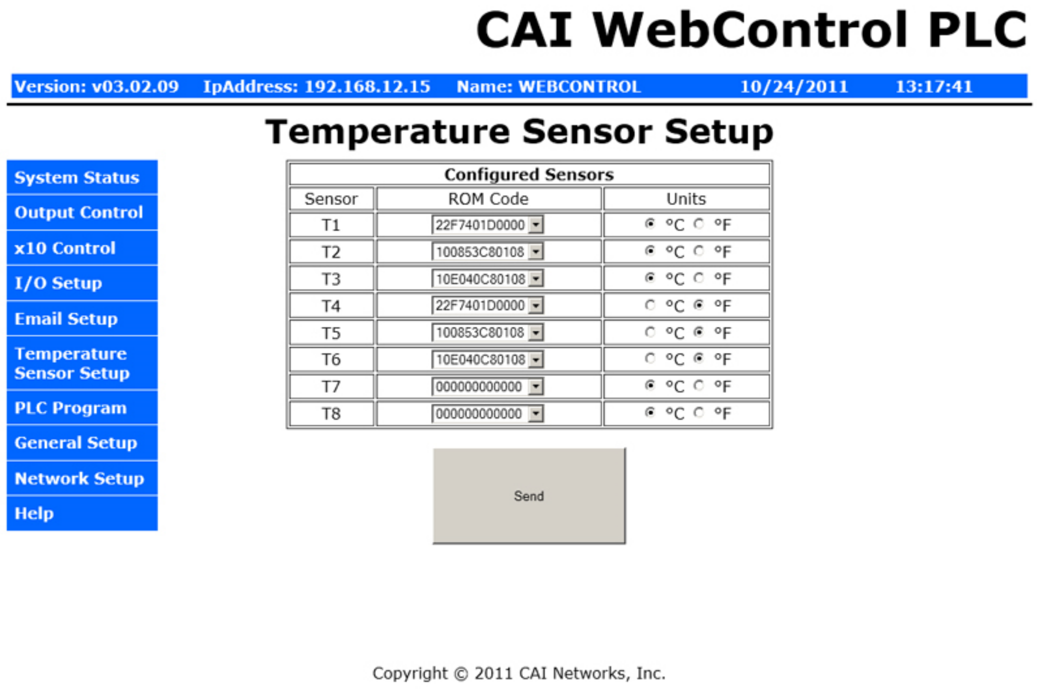
WebControl$^{TM}$ has eight digital TTL inputs, each of which can be configured to be inverted upon input to the system. The PLC engine will then look for a true of false case of the input. TTL level 0 is defined 0-1.75V; TTL level 1 is defined as 3-5V. When TTL1 being used as counter/frequency measurement purpose, its counts pulses up to 2MHz and max count is 32bit number. Counter starts counts from rising edge after the falling edge.

## 4.2 Analog Inputs

WebControl$^{TM}$ has three built-in analog inputs each having an input voltage range of 0 to +10V. When a DS2450 chip is connected to the 1-wire bus, additional analog input AIP5,6,7, and 8 are displayed. DS2450 chip has 10 bit resolution and full scale voltage is 0 to +5.0V. Both built-in and optional DS2450 A/D inputs have the full scale range of 0 – 1023. When DS2438 sensor is on the bus, DS2438's analog input will take the AIP5,6,7,8 and based on assignment in temp sensor setup to show in user defined location. DS2438 has 10V input range. DS2438 also has current measurement funciton, we discuss that later. In PLC code, user can reference to the fourth analog input value AIP4, which is the reading from humidity sensor.

## 4.3 DS1822/DS18B20 1 Wire Temperature Sensors

WebControl$^{TM}$ supports up to eight Maxim DS1822 /DS18B20 12bit 1 wire temperature sensors. Each temperature sensor must be first assigned a temp sensor number T1-T8.



Version: v03.02.09 IpAddress: 192.168.12.15 Name: WEBCONTROL 10/24/2011 13:17:41

Temperature Sensor Setup

Configured Sensors

| Sensor | ROM Code | Units |
|---|---|---|
| T1 | 22F7401D0000 | [✓ °C] °F |
| T2 | 100853C80108 | [✓ °C] °F |
| T3 | 10E040C80108 | [✓ °C] °F |
| T4 | 22F7401D0000 | °C [✓ °F] |
| T5 | 100853C80108 | °C [✓ °F] |
| T6 | 10E040C80108 | °C [✓ °F] |
| T7 | 000000000000 | [✓ °C] °F |
| T8 | 000000000000 | [✓ °C] °F |

Copyright © 2011 CAI Networks, Inc.

*Figure 10.0 WebControl$^{TM}$ Temperature sensor assignments*

Each temp sensor has unique ROM code. The first two digit shows the sensor type. When DS2438 presnts on the bus, it will start from "26". We recommend to assign them to temp sensor T5-T8. One temp sensor can be assigned for more than one sensor number. User can also select the unit as Centigrade or Fahrenheit. If sensor failed later due to sensor failure or wire problem, WebControl$^{TM}$ will display the state as failed, but keep the last valid sensor value. This is to prevent the PLC logic turn on heater/cooler or motor undesirably.

## 4.4 Honeywell 4000 Series Relative Humidity Sensor

The WebControl$^{TM}$ hardware is designed to support one Honeywell 4000 series relative humidity sensor. The Honeywell sensor output is 'almost' linear voltage between 0 - +5V dc proportional to the relative humidity. This output is fed into one of the A/D converter channels on the microcontroller. It can be read as H1 in PLC.

WebControl PLC has calibrated the humidity curve at 11%, 25%, 45%, and 78%. However, due to sensor differences, some sensors may read the humidity value different from actual value. Adding a 5K linear potentiometer (pot) in series with pin 3 (5V supply line) can help user to adjust your sensor match the accurate humidity reading. If the humidity reading varies a lot, which is an indication your power supply is not providing enough voltage to the board.

## 4.5 I2C Support

The WebControl$^{TM}$ firmware added I2C support on the top two spare pins on the 8 pin header. Please look Figure 1 spare pin header. The first one with pull up resistor is SCL I2C clock pin and second one with pull up resistor is Data pin SDA. These two pins are pulled up to 3.3V, but they are 5V tolerate, so that I2C device can be 3.3V or 5V. I2C device power can be 3.3V or 5V, please check the spec sheet. For example, BMP180 pressure sensor is 3.3V device, however, market selling module has a 5V to 3.3V regulator on it, so that the module must be powered by 5V. The communication is through added PLC commands I2CWRITE and I2CREAD. I2CWRITE from this version will set ZBIT equal to ACK from the device, that can be used to identify if device exist on the I2C bus or not. Please check the PLC command syntax in chapter six. WC8's I2C support includes clock stretching up to 255 uS. It does not support bus mater arbitrition at this time. Any communication to the slave I2C device must start with SEND_START, the last communication it must send SEND_STOP. Please see the PLC program example section for I2C usage.

## 4.6 SPI Support

The WebControl$^{TM}$ firmware added SPI support on the spare pin header, please refer to Fig 1 for detailed pin location and function. To communicate with SPI devices, a PLC command SPIBYTE is added, it takes 3 parameters, mode, byte-out, byte-in.

Example: SPIBYTE 1 RAM11 RAM21 ; where RAM11 is send to bus, RAM21 contains byte received from bus. If more than one device on PLC bus, a TTL out pin may be used to enable the device on SPI bus.

In the current released firmware, only mode 1 is supported. In technical term, mode 1 means phase CPHA=0 and polarity CPOL=0. This is most common mode in SPI devices. Please contact us for any different mode support.

## 4.7 Timers

WebControl$^{TM}$ PLC is different from BRE engine. User must write PLC code to implement timers and delay functions. There is no hard limit of number of timers and delays, There are blocking DELAY and non-blocking delay functions detailed in section 6.2. WebControl PLC has built-in timer reference value for PLC coding:

| | |
|---|---|
| ❑ CD | Current date mm/dd/yyyy format |
| ❑ CT | Current time hh:mm:ss format |
| ❑ CDW | Current day of week |
| ❑ CH | Current hour of day |
| ❑ CM | Current minute of hour |
| ❑ CS | Current second of minute |
| ❑ CDAY | Current day of month |
| ❑ CMONTH | Current month of year |
| ❑ CYEAR | Current year |
| ❑ CTS | UTC time, seconds since 1/1/1970 |

WebControl timer variables are in one mili-second minimum resolution. However, due to high priority I/O like handling, like 1-wire interrupt, the accuracy of the timer may in the range greater than 10mS. For how to program the timer functions, please read Chapter 6 completely. There are a few examples described how to program the WebControl PLC to act based on the input conditions and timer conditions.

## 4.8 Direct Query System Inputs and Outputs

To integrate with other servers, directly query I/O status maybe desirable. Using wget command from other servers or computers, user can directly query each input and output status. The following HTTP commands are directly return the I/O values:

“geta1.cgi, geta2.cgi, geta3.cgi” – to query analog input values “geth1.cgi” – to query the humidity sensor readings

“geti1.cgi, geti2.cgi,..geti8.cgi” – to query each digital input values

“geto1.cgi, geto2.cgi,...geto8.cgi” – to query each TTL output status

“gett1.cgi, gett2.cgi,...gett8.cgi” – to query each temperature sensor value

“getts1.cgi, getts2.cgi,...getts8.cgi” – to query each temperature sensor status

“getcounter.cgi” – to query the counter value

“getfcounter.cgi” – to query the frequency measurement per second value.

“getall.cgi” – to query all sensor and I/O status

VAR1-VAR8 can also be set by web UI. Please see FAQ section for how to do that.

# 5 Real Time Clock

WebControl$^{TM}$ has a build-in clock functions that doing time keeping by using an atomic clock via the simple network timer protocol (SNTP). Configuration is required to set the correct time zone of the clock. When NTP is not available, WebControl$^{TM}$ will use its own crystal to maintain the clock accuracy. If NTP is available, only need to set the time zone in the clock page. If NTP is not available, please set your local clock in the clock page also. Without NTP, its internal clock may not as accurate. Firmware 3.2.22 and later supports DS1307 I2C real time clock IC. This RTC module can be easily obtained from market places.

To support user I2C real time clock module, from 3.02.19, we have removed DS2417 support. DS1307 support is within this firmare without any PLC coding. CTS now uses GMT format, it starts from 1/1/1970.

.

## 5.1 Network Requirements to Use the NTP

WebControl$^{TM}$ must be connected to a network that has access to a DNS server. The primary and secondary DNS server addresses of WebControl$^{TM}$ are required to be configured correctly. WebControl$^{TM}$ will use DNS to resolve the name of the timeserver. The NTP server WebControl accessing is pool.ntp.org. If internal NTP server must be used, DNS server must resolve the name record for pool.ntp.org to your local NTP server's IP address.

# 6 WebControl$^{TM}$ PLC Programming

The WebControl$^{TM}$ PLC firmware can be programmed to execute programmable logic sequences, including comparison and sub routines. This is the major change from the BRE(Boolean Run Engine) version firmware. WebControl$^{TM}$ PLC uses assembly like PLC language. Its main program starts with "START" and finishes with "END". The PLC program is pasted into the web GUI. WebControl will automatically store it into its EEPROM so that if recycle power will not lose the program. The limitation of the PLC is 1000 line of code. The support for PLC programming is not included in the free support for configuration of WebControl$^{TM}$ PLC.

## 6.1 The Basics of PLC Programming

WebControl PLC program is NOT hard! We have included many examples toward the end of this chapter. A PLC program is made up of main routine and optional subroutines.

The main routine is enclosed between mandatory START and END instructions e.g.

```
START
#main instructions go here
END
```

To change the logic flow, GOTO instruction can be used as unconditional jump from one section of logic to another without call stack to return. CALLSUB instruction is for unconditional logic flow that may return to the original call stack. BNZ and BZ are conditional GOTO to branch to different section in logic. CNZ and CZ are conditional CALLSUB instruction allowing returning to call stack address.

Both GOTO and CALLSUB use **label** to identify where to execute next instruction. Label can be any string less than 10 characters. Label cannot be identical to any instruction keyword. If sub routines are used then they are coded after the END of the main routine body. Sub routines start at their label and must end with the instruction RET e.g.

```
TEST_IO_SUB:
#instructions here
RET
```

Subroutines can be called from the main program and from within other subroutines. Note that WebControl PLC has a return **program address stack** depth of 8 (or call stack 8).

The program control block has a zero bit that is updated implicitly on TEST instructions. This zero bit is set by any one of these TEST instructions: TSTEQ, TSTNE, TSTGT, TSTLT, TSTGE, TSTLE, ANDT, ORT, XORT, ANDBT, ORBT, XORBT, TSTB. Zero bit flag can be accessed by IO name ZBIT. This zero bit can also be used implicitly when using branch and call instructions, like BZ, BNZ, CZ, CNZ. E.g. the following test instruction yields a Boolean result which will implicitly set the state of the zero bit. Next a branch instruction is used which branches on the state of the zero bit.

```
TSTEQ IP1 1   # sets zero bit based on the result of the test instruction
BNZ label     # branches to label if zero bit is non-zero
```

Format of instructions:

```
label: (optional)
        opcode operands
```

Labels must be terminated with a colon ':' and can be a maximum of 10 characters.

## 6.2 WebControl$^{TM}$ PLC Instructions

The following symbols are used in the table below:

d = destination

a,b..c = operands

() = optional, any operand enclosed in parenthesis mean it is an optional operand.

[] = non-blocking delay operator optional to TTL input/output and VARs. 32 bit unsigned number, represent resolution of 0.001 seconds. When the delay operator is used on input operands the current value of that input is only used if it has had that value for greater than the delay period specified between the brackets. When this operator is used on an output operand the output value is only set for the period specified in the brackets. All delay periods are specified in milliseconds. Note that accuracy and timer resolution is approximately 100ms, even though the delay operator value is in unit of 1mS.

| Opcode | Operands | Description |
|---|---|---|
| START | | Start of main program |
| PROTECTED | | Must be 2nd Opcode in PLC, will not display PLC code. |
| TSTEQ<br><br><br>TSTNE | a[] b[] (d[])<br><br><br>a[] b[] (d[]) | Tests if a is equal to b. Boolean result loaded into optional destination (d). Zero bit updated with result. If test evaluates to false then the next instruction is skipped.<br>Tests if a is NOT equal to b. Boolean result loaded into optional destination (d). Zero bit updated with result. If test evaluates to false then the next instruction is skipped. |
| TSTGT | a[] b[] (d[]) | Test if a is greater than b. Boolean result loaded into optional destination (d). Zero bit updated with result. If test evaluates to false then the next instruction is skipped. |
| TSTLT | a[] b[] (d[]) | Tests if a is less than b. Boolean result loaded into optional destination (d). Zero bit updated with result. If test evaluates to false then the next instruction is skipped. |
| TSTGE | a[] b[] (d[]) | Tests if a is greater than OR equal to b. Boolean result loaded into optional destination (d). Zero bit updated with |

| | | |
|---|---|---|
| | | result. If test evaluates to false then the next instruction is skipped. |
| TSTLE | a[] b[] (d[]) | Tests if a is less than OR equal to b. Boolean result loaded into optional destination (d). Zero bit updated with result. If test evaluates to false then the next instruction is skipped. |
| SET | a[] b | Sets I/O id a to the value of b. No zero flag update. |
| SETB | a[] b[] d[] | Read a and set bit b and puts the result into d. |
| GETB | a[] b[] d[] | Read a bit position b and puts the result into d. |
| CLRB | a[] b[] d[] | Reads a and clear its bit position b and puts the result into d. |
| TSTB | a[] b[] d[] | Reads a and test its position b and puts the result into d. Zero bit updated with result. |
| ADD | a[] b[] d[] | Adds a and b and puts the result into d. Zero bit updated with result. |
| SUB | a[] b[] d[] | Subtracts b from a and puts the result into d. Zero bit updated with result. |
| DIV | a[] b[] d[] | Divides a by b and puts the result into d. Zero bit updated with result. |
| MOD | a[] b[] d[] | Divides a by b and puts the residue into d. Zero bit updated with result. |
| MUL | a[] b[] d[] | Multiplies a by b and puts the result into d. Zero bit updated with result. |
| DEC | a | Decrements a by 1. Zero bit updated. |
| INC | a | Increments a by 1. Zero bit updated. |
| AND | a[] b[] (d[]) | Logical AND's a with b and optionally puts Boolean result into d. Zero bit updated. |
| ANDB | a[] b[] (d[]) | Bitwise AND's a with b and optionally puts bitwise AND result into d. Zero bit updated. |
| OR | a[] b[] (d[]) | Logical OR's a with b and optionally puts Boolean result into d. Zero bit updated. |
| ORB | a[] b[] (d[]) | Bitwise OR's a with b and optionally puts bitwise OR result into d. Zero bit updated. |
| XOR | a[] b[] (d[]) | Logical XOR's a with b and optionally puts Boolean result into d. Zero bit updated. |
| XORB | a[] b[] (d[]) | Bitwise XOR's a with b and optionally puts bitwise result into d. Zero bit updated. |
| ANDT | a[] b[] (d[]) | Logical AND's a with b and optionally puts Boolean result into d. Zero bit updated. Skip next line if zero. |
| ANDBT | a[] b[] (d[]) | Bitwise AND's a with b and optionally puts bitwise AND result into d. Zero bit updated. Skip next line if zero. |
| ORT | a[] b[] (d[]) | Logical OR's a with b and optionally puts Boolean result into d. Zero bit updated. Skip next line if zero. |
| ORBT | a[] b[] (d[]) | Bitwise OR's a with b and optionally puts bitwise OR result into d. Zero bit updated. Skip next line if zero. |
| XORT | a[] b[] (d[]) | Logical XOR's a with b and optionally puts Boolean result into d. Zero bit updated. Skip next line if zero. |
| XORBT | a[] b[] (d[]) | Bitwise XOR's a with b and optionally puts bitwise result into d. Zero bit updated. Skip next line if zero. |
| BNZ | (a) b | If the optional a operand is specified it is tested for a non-zero value. If a is not specified then the zero bit is tested for |

| | | |
|---|---|---|
| | | non-zero. If true then program jumps to label specified in operand b. |
| BZ | (a) b | Same as BNZ but tests for zero value. |
| CNZ | (a) b | Same as the branch instruction but calls a subroutine instead of branching. See section on program address stack. |
| CZ | (a) b | Same as above but tests for zero result. |
| CALLSUB | a | Calls subroutine with label a. See section on program address stack. |
| GOTO | a | Branches to program address specified by label a. |
| DELAY | a | Delay instruction, delay specified in 1/1000 seconds. This delay is blocking delay, so that next PLC instruction will not execute until delay is over. |
| NOP | | A no operation instruction. |
| RET | | A return from subroutine instruction. |
| EMAIL | a | Sends email, a = message ID to send EM1 - EM8. |
| X10 | a b c | a: house code 0-15, b: device code 0-15, c: ON, OFF, BRIGHT, DIM |
| WEBSET | a b | a: URL1-8, b: number or VAR, RAM or any other readable |
| SIND | a b | a: degree, b: VAR or RAM to store the result |
| COSD | a b | a: degree, b: VAR or RAM to store the result |
| TAND | a b | a: degree, b: VAR or RAM to store the result |
| ROTL | a b c | a: source register, b: number of bits; c: result register rotate to the left, overflow bits will be feed into right |
| ROTR | a b c | a: source register, b: number of bits; c: result register rotate to the right, overflow bits will be feed to the left |
| SETLED | 0\|1\|2 | 0 to turn off green LED, 1 to turn on green LED, 2 default blinking. |
| IPTS | a b | reading TTL input last state change time tick counts stores in first parameter a. 2nd paramter b is which TTL inut. |
| BCD2DEC | a b | convert BCD value b into Decimal value and store it in a |
| DEC2BCD | a b | convert Decimal value b into BCD value and store it in a |
| IPEDGE | undefined | not implemented yet |
| I2CREAD | a b c | a: send ack, b: send stop, c: byte to read |
| I2CWRITE | a b c | a: send start, b: send stop, c: byte to write. Ack value from I2C slave sets ZBIT value, so that PLC program can detect if slave device on bus or not. |
| SPIBYTE | a b c | a: mode, b: send byte, c: byte read from bus |
| SREBOOT | | Force board to restart. |
| FWVERSION | a | return firmware version |
| END | | End of main program. This instruction will set the program counter back to zero and the program will start executing from the beginning. |

## Operands

An operand can be any of the following:

- a signed 32 bit decimal number. e.g. 100 or 1 or 0 etc.
- a hexadecimal number. e.g. 0xABF.

- a date stamp in the format MM/DD/YYYY e.g 02/10/2010
- a time stamp in the format HH:MM:SS e.g. 20:25:00
- a day of week identifier enclosed in single quotes e.g. 'sun'. Day of week identifiers are 'sun' 'mon' 'tue' 'wed' 'thu' 'fri' 'sat'
- an I/O identifier that is a place holder for the real I/O value that the PLC engine will get at runtime. Valid I/O identifiers are explained in next section.

## 6.3 WebControl™ PLC I/O Identifiers

The following are the valid I/O identifiers

```
OP1            TTL Outputs 1...8    Valid range 0 - 1
OP2
OP3
OP4
OP5
OP6
OP7
OP8

IP1            TTL Inputs 1...8 Valid range 0 - 1
IP2            When non-blocking delay added to these input,
IP3            its value will return TRUE, if input from 0->1
IP4            longer than the delay value. For example,
IP5            IP1[1000] will return FALSE, if TTL input 1 from 0
IP6            to 1 last state change shorter than 1000ms.
IP7            IP1[1000] return TRUE only when input 1 from 0 to 1
IP8            and stay at logic 1 for longer than 1000ms.

IPINV1         TTL Invert Inputs 1...8 Valid range 0 - 1
IPINV2         This is exactly same TTL input as IP1,..IP8
IPINV3         except is inverted for filter short pulse purpose
IPINV4         its usage is like: IPINV1[1000] to filter 1->0
IPINV5         pulse shorter than 1000ms. If 1->0 pulse is shorter than
IPINV6         1000ms, it will return false. Only when input state
IPINV7         changed from 1 to 0 and stay that level for longer than
IPINV8         1000ms, the value will be TRUE.

AIP1           Analog Inputs 1...3 Valid range 0 - 1024
AIP2           AIP1 to AIP3 are built-in A/D converters.
AIP3
AIP5           16 bit signed value can store I2C 16 bit ADC result.
AIP6           16 bit signed value can store I2C 16 bit ADC result.
AIP7           16 bit signed value can store I2C 16 bit ADC result.
AIP8           16 bit signed value can store I2C 16 bit ADC result.

T1             Temperature sensor inputs 1...8 Valid range -550 - +1250.
T2             Note that temperature values are specifies in 10's of
T3             degrees. So to test for 21.6 degrees C you would use the
T4             value 216.
T5
T6
T7
T8

TS1            Temperature sensor state 1 or 0.
TS2            1 == GOOD sensor, 0 == bad sensor
TS3
TS4
```

| | |
|---|---|
| TS5 | |
| TS6 | |
| TS7 | |
| TS8 | |
| H1 | Humidity sensor valid range 0 - 100 |
| EM1 | Email identifiers 1...8 |
| EM2 | |
| EM3 | |
| EM4 | |
| EM5 | |
| EM6 | |
| EM7 | |
| EM8 | |
| URL1 | WEBSET these URLs 1...8 |
| URL2 | |
| URL3 | |
| URL4 | |
| URL5 | |
| URL6 | |
| URL7 | |
| URL8 | |
| CD | Current date mm/dd/yyyy format |
| CT | Current time hh:mm:ss format |
| CDW | Current day of week |
| CH | Current hour of day |
| CM | Current minute of hour |
| CS | Current second of minute |
| CDAY | Current day of month |
| CMONTH | Current month of year |
| CYEAR | Current year |
| CTS | Current total seconds since 1/1/1970 (32bit signed number) |
| VAR1 | 32 bit signed integer variables 1...8 |
| VAR2 | The value will be displayed in System Status |
| VAR3 | Delay operator is valid on these |
| VAR4 | It will help debug your program, if you store debug |
| VAR5 | value in VARx. |
| VAR6 | |
| VAR7 | |
| VAR8 | |
| RAM1 | 32 bit signed integer general purpose RAM 1...8. Delay |
| RAM2 | operator is not valid on these. Not displayed anywhere |
| RAM3 | |
| RAM4 | |
| RAM5 | |
| RAM6 | |

| | |
|---|---|
| RAM7<br>RAM8 | |
| RAM1H<br>RAM2H<br>RAM3H<br>RAM4H<br>RAM5H<br>RAM6H<br>RAM7H<br>RAM8H | 16 bit unsigned integer general purpose RAM 1...8. Delay operator is not valid on these. Not displayed anywhere store in the SAME RAM location as RAM1-8 higher 16 bits |
| RAM1L<br>RAM2L<br>RAM3L<br>RAM4L<br>RAM5L<br>RAM6L<br>RAM7L<br>RAM8L | 16 bit unsigned integer general purpose RAM 1...8. Delay operator is not valid on these. Not displayed anywhere store in the SAME RAM location as RAM1-8, lower 16 bits |
| RAM10<br>RAM20<br>RAM30<br>RAM40<br>RAM50<br>RAM60<br>RAM70<br>RAM80 | 8 bit unsigned integer general purpose RAM 1...8. Delay operator is not valid on these. Not displayed anywhere store in the SAME RAM location as RAM1-8, byte 0 location |
| RAM11<br>RAM21<br>RAM31<br>RAM41<br>RAM51<br>RAM61<br>RAM71<br>RAM81 | 8 bit unsigned integer general purpose RAM 1...8. Delay operator is not valid on these. Not displayed anywhere store in the SAME RAM location as RAM1-8, byte 1 location |
| RAM12<br>RAM22<br>RAM32<br>RAM42<br>RAM52<br>RAM62<br>RAM72<br>RAM82 | 8 bit unsigned integer general purpose RAM 1...8. Delay operator is not valid on these. Not displayed anywhere store in the SAME RAM location as RAM1-8, byte 2 location |
| RAM13<br>RAM23<br>RAM33<br>RAM43<br>RAM53 | 8 bit unsigned integer general purpose RAM 1...8. Delay operator is not valid on these. Not displayed anywhere store in the SAME RAM location as RAM1-8, byte 3 location |

| | |
|---|---|
| RAM63<br>RAM73<br>RAM83 | |
| RAM1B1<br>RAM1B2<br>…<br>RAM1B32 | 1 bit access to purpose RAM 1 only. This only availavble for RAM1 Delay operator is not valid on these. Not displayed anywhere, store in the SAME RAM1 location |
| UROM1<br>UROM2<br>UROM3<br>UROM4 | 32 bit signed integer user value stored in EEPROM read only, value sets through general tab on web GUI. |
| COUNTER<br>FCOUNTER | 32 bit counter can be read, compare, or set<br>read only, frequency per second up to 2MHz. |
| WSRPLY | read and write, automatically set by web server during WEBSET call when server specified a value like "SET_WC=12345678". |
| LED | read and write, When read 0 means LED is off, 1 means LED is on, 2 means LED is heart beat. When write, valie is 0,1,or 2. |
| ALLINS<br>ALLOUTS | read only, All 8 TTL as a byte. Only in 3.02.17a firmware<br>read write, All 8 TTL output as a byte, only in 3.02.17a firmware. |
| ZBIT | TSTxy and any SET command or command store value into 3rd paramter with result zero or not. From this version firmware, I2CWRITE will also set slave device ACK value to ZBIT. |
| EMAILQ | read only, When read 0 means no email waiting, >3 means queue is full. |

## 6.4 WebControl PLC Examples

For best understanding how PLC logic working, you can try to copy and paste the examples below into your WebControl PLC program screen to check them out. Please note PLC logic will execute from START to END. Then it will continue from START to END, forever repeating. If you last line of PLC code could skip next instruction, like those TST instruction, it might skip your first line when condition met. Please do pay attention to it. If you use CALLSUB to run subroutine, after finishing the subroutine, the logic will return back to where CALLSUB called and continue.

### 6.4.1 Example 1 Set Output based on condition

Control incubator heater connected to TTL output 1. If temperature T3 is less than 37 degree C turn ON the heater; if T3 is greater or equal to 39 degree C, turn the heater OFF,.

```
START
        TSTLE T3 370 OP1
        NOP
        TSTGT T3 389
        SET OP1 0
END
```

However, this does not work, when temperature reached above 37 degree C, the OP1 turned off, instead of reaching to 39 degree C. The problem is the OP1 can be set to ON or OFF by the “TSLE T3 370 OP1” line alone. To reduce heater relay constantly turning ON and OFF, it is better to have two subroutines to handle the OP1 state.

```
START
        TSTLE T3 370
        CALLSUB  HEAT_ON
        TSTGT T3 389
        CALLSUB  HEAT_OFF
END

HEAT_ON:
        SET  OP1  1
RET

HEAT_OFF:
        SET  OP1  0
RET
```

What if the heater relay stuck, or the heater elements burn out? You can add additional function in this PLC program to send you email notice or sound an alarm for such a situation.

### 6.4.2 Example 2: Flash TTL output

Flashes output 2 at a rate of 1Hz. (Please note OP2[500] has no space in between which delays 500ms). Please note the delay function used in this example, which is non-blocking, that means the PLC logic will immediately execute next TSTEQ instruction until the delay is over.

```
START
        TSTEQ OP2[500] 0
        SET OP2 1
        TSTEQ OP2[500] 1
        SET OP2 0
END
```

Another way to implement this is:

```
START
        XOR OP2 1 OP2
        DELAY 500
END
```

### 6.4.3 Example 3: Push Button Input Control Output

When a push button connected to TTL input 1 being pushed, set the TTL output 3 ON.

```
START
        TSTEQ IP1 1 OP3
        TSTEQ OP3 1
        SET OP3 0
END
```

Although this example works, it has flaw. Because all the pushbutton switches from market will not have clean instant ON or OFF, they actually produce a bunch of ON and OFF signals when pushed. If you use a scope to watch IP1 input line or OP3 output line, you will see they are many ON and OFF pulses during pushing or releasing the button. The better way to handle the case is like Example 10 later in this manual. However, if TTL OIP1 connected to a clean signal source, this example does work well.

### 6.4.4 Example 4: Send EMAIL

To send email 1 when T3 – T2 >= 20 degrees you would use:

```
START
        SUB T3 T2 RAM1
        TSTGE RAM1 200
        EMAIL EM1
END
```

The above rule is a bit too simple because an email will constantly be generated while RAM1 is greater than or equal to 200. (20 degrees in this case) To guard against that the following logic should be considered. This implementation will only send one email when the temperature comparison beyond the range. If you turn on heater or cooler instead of sending email, similar consideration also should be excised:

```
        START
        SET RAM2 0
LOOP:
        SUB T3 T2 RAM1
        TSTGE RAM1 200 RAM1
        GOTO SEND
        SET RAM2 0
        GOTO LOOP
END

SEND:
        BNZ RAM2 LOOP
        SET RAM2 1
        EMAIL EM1
        GOTO LOOP
```

Please note in SEND portion of the code, RAM2 is being checked, if it is already 1, it will skip sending email. Only when RAM2 == 0, an email will be send. Only when RAM1 < 200, the LOOP will skip SEND logic and reset RAM2 to 0.

### 6.4.5 Example 5, Parallel I/O

This simple program performs 4 separate I/O checks and sets OP1 to OP4 states.
In this example, we use subroutine feature of the PLC logic. Although in this case, GOTO can do same thing as CALLSUB, CALLSUB can help program more readable. In certain logic, you have to use CALLSUB instead of GOTO, depending on the logic flow. We will explain more when we run into those examples.

The logic in this example follows:

OP1 is set if T3 > 50

OP1 is cleared if T3 < 50

OP2 is set if IP1 == 1 for more than 300ms

OP2 is cleared if IP1 == 0

OP3 is set if AIP1 + AIP2 > 1024

OP3 is cleared if IP4 == 1

OP4 is set if OP1 == 1

OP4 is cleared if OP1 == 0

The PLC code written for the above scenarios would be as follows:

```
START
        CALLSUB checkOP1
        CALLSUB checkOP2
        CALLSUB checkOP3
        CALLSUB checkOP4
END

checkOP1:
        TSTGT T3 500 OP1
        RET

checkOP2:
        TSTEQ IP1[300] 1 OP2
        RET

checkOP3:
        ADD AIP1 AIP2 RAM1
        TSTGT RAM1 1024
        BNZ l1
        TSTEQ IP4 1
        BNZ l2
        RET

l1:
        SET O3 1
        RET

l2:
        SET O3 0
        RET

checkOP4:
        TSTEQ OP1 1 OP4
        RET
```

Please note in this example, we assume the TTL input, analog input and temperature reading are all perfect, no bouncing up and down. In reality, you have to add consideration to it.

### 6.4.6 Example 6, Sequential I/O

The following simple program shows how to set-up sequential I/O.

```
OP1 is set when IP1 rises from 0 to 1
OP4 is cleared when IP1 rises from 0 to 1
OP1 is cleared when O4 == 1
OP2 is set when T3 > 25 AND OP1 == 1
OP2 is cleared when OP1 == 0
EMAIL1 is sent when OP2 is set
O4 is set when OP2 == 1 for more than 1 second
```

The assembly language written for the above scenario would be as follows:

```
START
        BNZ IP1 start
l1:
        TSTEQ IP1 1
        BZ  l1
        SET OP1 1
        SET OP4 0

l2:
        TSTGT T3 250 RAM1
        AND OP1 RAM1
        BZ  l2
        SET OP2 1
        EMAIL EM1

l3:
        TSTEQ OP2[1000] 1
        BZ  l3
        SET OP4 1
        SET OP1 0
END
```

Please note in this example, we assume the TTL input, analog input and temperature reading are all perfect, no bouncing up and down. In reality, you have to add consideration to it.

### 6.4.7 Example 7, Traffic Lights

This example will let pedestrian to push a button to change the light on a busy street, so that he can cross the street safely. IP1 hooks up to the pedestrian crossing button. If someone pushed cross button, the street will have amber light on for 10 seconds, then red light to stop all the cars, allowing pedestrian to cross street in next 30 seconds. At the end of 30 seconds, it will flash the amber and red light for 5 seconds. VAR1 in the main program will let the crossing light turn on every 100 seconds, does not matter anybody push the crossing button or not.

OP1 Red + Pedestrian crossing light
OP2 Amber
OP3 Green
IP1 Pedestrian Push Button

```
START
        CALLSUB LIGHTS_GO
loop:
        SET VAR1[10000] 1
loop1:
        TSTEQ IP1 1
        BNZ sr
        BZ VAR1 sr
        GOTO loop1
sr:
        CALLSUB STOP
        GOTO loop
END

LIGHTS_ST:
        SET OP1 1
        SET OP2 0
        SET OP3 0
        RET

LIGHTS_GO:
        SET OP1 0
        SET OP2 0
        SET OP3 1
        RET

LIGHTS_AM:
        SET OP1 0
        SET OP2 1
        SET OP3 0
        RET

STOP:
        CALLSUB LIGHTS_AM
        DELAY 5000
        CALLSUB LIGHTS_ST
        DELAY 60000
        CALLSUB LIGHTS_AM
        SET RAM2 5
flash:
        XOR OP2 1 OP2
        DELAY 500
        DEC RAM2
        BNZ flash
        CALLSUB LIGHTS_GO
        RET
```

### 6.4.8 Example 8, Time based Control

WARNING: For time critical application, please make sure to check CYEAR correct before taking action based on system clock. When WebControl$^{TM}$ boot up, it will uses ROM hard coded time in 2011. If PLC reporting year 2011, the NTP or real time clock has not sync the local clock yet.

This example will have five subroutines. WebControl$^{TM}$ PLC will continuously loop through them. The "HOURLY" routing will compare analog input 1 and analog input 2, if A1 - A2 > 10, send an email notice 1. You can use similar logic to adjust your solar panel orientation each hour, etc. The "PERIOD" subroutine will turn on night light hooked up to OP1 after 18:00 hours and turn it off at 5AM. The "DAILY" subroutine will start the water sprinkler at 6:30AM for four zones. The "MONTHLY" routing will check the "salt low" sensor AIP3 on the water softener and send email notice 2. The "YEARLY" routing will ring the New Year's bell connected to OP6 on each and every New Year's Day for the whole day!

```
START
        CALLSUB HOURLY
        CALLSUB PERIOD
        CALLSUB DAILY
        CALLSUB MONTHLY
        CALLSUB YEARLY
END

HOURLY:
        TSTNE RAM1 CH
        GOTO T1
RET

T1:
        SET RAM1 CH
        SUB AIP1 AIP2 RAM2
        TSTGT RAM2 10
        EMAIL EM1
RET

PERIOD:
        TSTGE CH 18 RAM2
        NOP
        TSTLE CH 5 RAM3
        NOP
        OR RAM2 RAM3 OP1
RET
```

```
DAILY:
        TSTEQ CH 7
        SET RAM5 0
        TSTEQ CH 6
        BZ  NOTYET
        TSTGT CM 30
        CALLSUB WATERING
NOTYET:
RET

MONTHLY:
        TSTNE RAM4 CMONTH
        GOTO T2
RET

T2:
        TSTLE CH 8
        GOTO 2EARLY
        SET RAM4 CMONTH
        TSTLT AIP3 20
        EMAIL EM2
2EARLY:
RET

YEARLY:
        TSTEQ CMONTH 1 RAM2
        NOP
        TSTEQ CDAY 1 RAM3
        NOP
        AND RAM2 RAM3 OP6
RET

WATERING:
        BNZ RAM5 W_DONE
ZONE1:
        SET OP2 1
        TSTLE CM 35
        GOTO ZONE1
        SET OP2 0
ZONE2:
        SET OP3 1
        TSTLE CM 40
        GOTO ZONE2
        SET OP3 0
ZONE3:
        SET OP4 1
        TSTLE CM 45
        GOTO ZONE3
        SET OP4 0
ZONE4:
        SET OP5 1
        TSTLE CM 50
        GOTO ZONE4
        SET OP5 0
        SET RAM5 1
W_DONE:
RET
```

Please note in DAILY subroutine, we call another subroutine “watering”. In this place, we have to use subroutine, since we only want to call this routine after 6:30AM. If current minute is not 30, we will skip. Watering will be done before 7AM, so that we clear the flag RAM5 at 7AM. Also, please notice RAM1, RAM4 and RAM5 are holding static value and RAM2 and RAM3 are temporary storage being used by more than one subroutine. You can decide which RAM is for temporary data, which is for static value.

In the PERIOD subroutine, we constantly compare the time and set the OP1 ON or OFF. That is okay for solid state relay or other control relay, since the logic level did not change all the time. However, if you are sending a X10 command to turn on and off different lights, you want to make sure the X10 command only issued once, not repeatedly. You may create another subroutine in which set flag only calls

```
X10  1  15  ON
```

only once to turn ON light at house code 2, unit code 16 (please note WebControl’s X10 house code range 0-F, and device code range also 0-F.)

In the MONTHLY routine, we first check the current hour being 8AM then we check the water softener’s salt level. In this way you will not be waked up by email in the midnight.

When program WebControl PLC for time based logic, please make sure the time being used in different part of the program having no conflict between all the subroutines. If you want two things to happen at the same time, you should consider combine them into same routine to handle.

### 6.4.9 Example 9, Battery Charger

This is a PLC program to charge 3 serially connected NiMH batteries. First, it tries to discharge the batteries individually. If any battery discharged to 1V, it will stop the discharge and start charging. When each cell is being charged to 1.25V, it will stop charging. We assume the A1, A2, and A3 being calibrated to 1V=100. The measurement on the battery 2 is the total voltage of battery 1 and battery 2. And the measurement on battery 3 is the total voltage of all three batteries. This example will individually discharge and charge each battery.

```
start
        set op1 1
        set op2 1
        set op3 1
        set RAM1 0
        set RAM2 0
        set RAM3 0
loop:
        cnz op1 check_b1
        cnz op2 check_b2
        cnz op3 check_b3
goto loop
end
check_b1:
        BNZ RAM1 c1
        tstle AIP1 100 RAM1
        bz  e1
c1:
        tstgt AIP1 125
        bnz  e1
        set  op1  0
        set  op4  1
e1:
ret
check_b2:
        BNZ  RAM2  c2
        sub  AIP2  AIP1  RAM4
        tstle  RAM4  100  RAM2
        bz  e2
c2:
        sub  AIP2  AIP1  RAM4
        tstgt  RAM4  125
        bnz  e2
        set  OP2  0
        set  OP5  1
e2:
ret
check_b3:
        BNZ  RAM3  c3
        sub  AIP3  AIP2  RAM4
        sub  RAM4  AIP1  RAM4
        tstle  RAM4  100  RAM3
        bz   e3
c3:
        sub  AIP3  AIP2  RAM4
        sub  RAM4  AIP1  RAM4
        tstgt  RAM4  125
        bnz  e3
        set  OP3  0
        set  OP6  1
e3:
ret
```

### 6.4.10 Example 10, RFID reader and browser Control

For office door using RFID reader, as well as allowing operator remote browser control, the following program provided the example. RFID reader's NC (normally connect) output connects to IP1 on WebControl digital input. A 2.2K pull-up resistor also connected between IP1 and 5V. In this way, each time a valid RFID tag sensed, a TTL “1” feeds to WebControl IP1. Remote operator can also open the door by using browser set OUTPUT TTL1 to on. OP1 connects to the door open switch. TESTEQ logic will make sure the OP1 is an 1 second momentary output.

“LIGHTS” subroutine is for light control outside the office door; the light is on at 7PM and off at 5AM.

```
START
        CALLSUB LIGHTS
        TSTEQ RAM1 0
        CALLSUB SET_OP1
        CALLSUB CHK4LOW
        TSTEQ OP1[1000] 1
        SET OP1 0
END

CHK4LOW:
        TSTEQ IP1 0
        SET RAM1 0
        RET

SET_OP1:
        TSTEQ IP1 1 RAM1
        SET OP1 1
        RET

LIGHTS:
        TSTGE CH 19 RAM2
        NOP
        TSTLE CH 5 RAM3
        NOP
        OR RAM2 RAM3 OP3
        RET
```

In this logic, SET_OP1 must be subroutine, if “TSTEQ RAM1 0” it will call the subroutine SET_OP1, but if RAM1 != 0, then it will skip that call. In this way, we can guarantee OP1 only being turn on once. In the subroutine SET_OP1, it checks if RFID reader did detected valid RDID card present. If so, it will set the flag RAM1 to true, so that not being set over and over again. For gate opening device, this will make sure the gate will not left open all the time.

### 6.4.11 Example 11, Bitwise Operation

There are ANDB, ORB, and XORB operator operate on the VAR or RAM on the bit basis. This will allow each RAM or VAR to store up to 31 binary states. When VAR1 stores value 12345, and RAM1 stores value 256, after execute
ANDB VAR1 RAM1 VAR1
VAR1 stores the value 0

After execute:
ORB VAR1 RAM1 VAR1
VAR1 stores the value 123712

XORB VAR1 RAM1 VAR1
First execution will be 123712, if execute next time, VAR1 will be back to 12345. In another word, XORB can toggle the bit.

### 6.4.12 Example 12, Angle Calculation

From 3.02.16c firmware, angle calculation is supported. In PLC program, users can:
SIND 91 VAR1
Or
COSD 185 VAR2
Or
TAND 630 VAR3
The result for SIND and COSD is x1000, because we can only have integer on this processor. The result for TAND is x100.

### 6.4.13 Example 13, Non-Blocking Delay

Non-blocking delay is expressed in PLC code as [] next to the operators. The number inside [] is micro-seconds. The [] operation can be on both operators during TST operations. Each input and output and VAR associated with a non-blocking delay timer value. That value is set when I/O state is changed or VAR value being modified. If later PLC instruction using non-blocking delay, that timer value will be referenced. If current time is less than stored timer time plus the delay period, the specified operation will not be performed. Reading value with non-blocking delay will return false if timer value is not meet For example,

TSTGT VAR1[1500] IP1[300] RAM2
If any of those delay not reached, its result will be FALSE.

For SET VAR1[15000] IP1
VAR1 will not be set to IP1 value, unless 15000 milliseconds (15seconds) passed.

### 6.4.14 Example 14, WEBSET to get server reply

From 3.02.16 version firmware, WebControl allows PLC call WEBSET to do HTTP GET call to HTTP servers inside another WebControl or other devices, or Apache or IIS servers. The web server CGI code can process that information. From 3.02.17 version firmware, the web server can also send back to the WebControl a specially formatted string. When WebControl received that string, it will set an internal variable for user PLC code to reference. Following is an example CGI code from apache server that will set the WebControl WSRPLY in the WebControl. PLC program can base on that reply to turn on or off an I/O bit or take any other action.

Because PLC code does not execute WEBSET call immediately, rather WEBSET is on a scheduler called from queue, user can not expect to get server reply immediately after WEBSET call. The good practice would to check if the WSRPLY value is zero, if that is zero, the server reply has not fetched back yet. Server must return a non-zero value back. If WSRPLY is non-zero, user PLC must read it into another variable and set it to zero, so that it can be used for next WEBSET call. In this sense, if PLC logic wants to get multiple values from server reply, it must issue one call at a time to avoid different WEBSET call return value clashing.

### 6.4.15 Example 15, Server CGI Handles WEBSET

The code below is for demonstration only, it is written in C on apache server. We do not provide support for writing server CGI code. Please note to get the best result, server CGI code should write back the string as early in the reply as possible. In HTTP server reply processing, server mostly after sending reply will close connection. If the “SET_WC=2147483647” string sending out too late, it could get lost because the connection is closed already. Please test and make sure your server is response fast enough for the WEBSET call.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(void)
{
char *data, *remote_mac, *remote_host;
long m,n;
FILE *fp;
char buffer[18] = "\r\nnew get call\r\n";
printf("%s%c%c\n",
"Content-Type:text/plain;charset=iso-8859-1",13,10);
data = getenv("QUERY_STRING");
remote_mac = getenv("HTTP_USER_AGENT");
remote_host = getenv("REMOTE_ADDR");
fp = fopen("/tmp/webcontrol.txt", "w+");
fwrite(buffer, 1, strlen(buffer), fp);
if(data != NULL) {
fwrite(data, 1, (unsigned int) strlen(data), fp);
fwrite(buffer, 1, strlen(buffer), fp);
}
if(remote_host != NULL) {
fwrite(remote_host, 1, (unsigned int) strlen(remote_host), fp);
fwrite(buffer, 1, strlen(buffer), fp);
}
if(remote_mac != NULL) {
```

```
fwrite(remote_mac, 1, (unsigned int) strlen(remote_mac), fp);
fwrite(buffer, 1, strlen(buffer), fp);
} else
fwrite(buffer, 1, strlen(buffer), fp);
/**************************************************
put your logic here if MAC address not match then do what
You can simply return with an error message
**************************************************/
/***************************************************
Now, you can process the data string send in by WebControl
**************************************************/
printf("to push up to 10 bytes to WebControl SET_WC=-2147483647\n");
printf("close connection\n");
fclose(fp);
return 0;
}
```

### 6.4.16 Example 13, I2C Slave Device Communication

WebControl 8 firmware from this version on supports I2C device communication through the top two spare pins with pull up resistors. The first pin is SCL and second pin is SDA. To communicate with the I2C device, first to send command to the slave device what to do, then send command to slave what to read back. This example will show how to communicate with TI ADS1115 16 bit ADC and store channel 1 data to AIP5

```
START
        CALLSUB ADS1115
END

ADS1115:
        I2CWRITE 1 0 144        # I2c start, write address for chip with addr pin
                                # connected to ground
        I2CWRITE 0 0 1          # configuration register address is 1
        I2CWRITE 0 0 193        # single end AINp = A0, full scale 6.144V,
                                # one shot conversion, see page 18-19
        I2CWRITE 0 1 131        # 128 sample/s, disable comparator,
        DELAY 10                # wait for conversion done,
        I2CWRITE 1 0 144        # I2c starts, write to chip on this I2c address
        I2CWRITE 0 1 0          # tell ADS1115 we address register 0,
                                # send STOP
        I2CWRITE 1 0 145        # I2c starts, tell chip we will read
        BNZ  NO_DEV             # if device exist? not, then do not read
        I2CREAD 0 0 RAM11       # reads MSB byte, notice ADS1115 require
                                # ACK, which is zero in ACK bit
        I2CREAD 0 1 RAM10       # reads lower byte, send stop to tell ADS1115
                                # no more read
        MUL RAM1L 1872 VAR1     # 1872 is the scale factor for 6.144 full scale
        DIV VAR1 10000 AIP5     # divide by 10000 to set value to 0.001V scale,
                                # store in AIP5
        RET
NO_DEV:
        I2CWRITE 1 1 144        # For release I2C bus by insert STOP bit to bus
        SET AIP5 0              # ADS1115 not on I2C bus, so set AIP5 to zero
RET
```

### 6.4.17 Example 14, SPI Device Communication

WebControl 8 firmware from 3.02.18dx added SPI device support.  Following is a short example communicate with the MAX6657 temperature sensor probe.  SPI device has no address, but PLC code need to turn on Chip Select to 0 to enable that SPI sensor to output to the bus. If more than one SPI devices, please make sure turn all sensor CS  pin to 1 before communicate with selected sensor.

```
START
        SET OP1 0               # max6675 chip enable
        SPIBYTE 0 0 RAM11       # reading higher 8 bites, mode 0
        SPIBYTE 0 0 RAM10       # reading lower 8 bits, mode 0
        SET VAR1 RAM1           # copy to see what value
        SET OP1 1               # turn off CS
        ANDB VAR1 32760 VAR2    # remove lower 3 bits
        ROTR VAR2 3 VAR3        # shift to lower 3 bits
        MUL VAR3 25 VAR4        # times 25 for actual value
        DELAY 100               # just wait here or do anything you like
END
```

# WebControl$^{TM}$ PLC FAQ

We include some users frequently asked questions here:

## 6.5 Login and Configuration

1-1Q: Can you tell me how to connect with Windows Explore? I cannot communicate?
1-1A: If you have DHCP server on your network, please check with your DHCP server log, WebControl is likely obtained an IP address from your DHCP server. Once your find out the IP address assigned by DHCP server, you can use browser to connect to:

http://what-ever-dhcp-assigned-ip/
You will see the login screen.

If you don't have DHCP server on your network, WebControl's default IP address is 192.168.1.15. You must change your computer's IP address temporarily to 192.168.1.1, (make sure no other host using that IP address on your network), then from IE browser enter:

http://192.168.1.15/
You will see the login screen.

1-2Q: I have problem to setup the clock, it does not work, even I setup my network and DNS server correctly?
1-2A: Please check with your ISP to make sure the DNS server IP address is valid. If you have Linux computer, you can use dig command to make sure that DNS server IP can resolve the ntp.org. Please note that many ISPs restrict the DNS access from outside its own IP address range. If your DNS server IP is not from your ISP, it may not work.

```
dig ntp.org @your-dns-server-ip
```

1-3Q: What is the default user ID and password, can I change it?
1-3A: The default user ID and password is admin/password, all lower case. User can change both user ID and password.

## 6.6 Temperature Sensor Support

2-1Q: The document stated supporting DS1822/DS1820 1 Wire Temperature Sensors, I assumed the whole DS18xx family of devices would work. I used a DS18S20+ and get about 3.5 degrees C when powered, 10.6 when parasitic powered. It should read 23-24 at room temperature. For power I tried pin 3 of the temperature terminal and pin 3 of the humidity terminal and the result is the same.
2-1A: WebControl™ can simultaneously connect 8 optional DS18B20/DS1822/DS18S20 based digital temperature sensors. In the past we require 12bit temperature sensor. Now, we lose the requirement to 12bit and 9 bit Maxim-IC temp sensors. We still do not support parasite part.

2-2Q: How to read temperature or sensor ROM code from command line in Linux?
2-2A: Use wget http://webcontrol-ip/gett1.cgi to read the temp sensor T1, and use wget http://webcontrol-ip/gett1rc.cgi to read the sensor T1 ROM code.

2-3Q: What can cause my temperature sensor not display correctly?
2-3A: Use solid copper wire for your 1-wire bus, for example, CAT5 cable. Reduce any unnecessary length of the wire. Do not hook up unsupported parts to the 1-wire bus, since they may generate 1-wire signal causing trouble.

2-4Q: I run a long cable between the DS18B20 and WebControl, sensor does not work?
2-4A: According to Maxim-IC, if using a long cable connecting between the sensor and host controller, it may require to add a pull-up resistor 4.7k from 1-wire bus (DQ pin on DS18B20) to the 5V supply near the far end of the 1-wire bus. Please check out Maxim-IC AppNote148. You do not have to add external power, if you do not experience any problem.

## 6.7 Turn on/off TTL output from another programming language

3-1Q: I want to turn on/off TTL output from Visual Basic, can I do it?
3-1A: You can reference how the browser does it and emulate that in your VB or script. Please make sure to disable login in the "Network Setup" screen. For security purpose, please specify the IP address in the access list, or your application sending encrypted user ID/password in the same HTTP call. Depends on the IP address and which TTL you want to control by programming language, you may refer to these two browser URL lines:

http://192.168.1.15/api/setttloutput.cgi?output=1&state=1 to turn on,
http://192.168.1.15/api/setttloutput.cgi?output=1&state=0 to turn off.

and to manually set a VAR value from outside, varid from 1 to 8:
http://192.168.1.15/api/setvar.cgi?varid=1&value=23456789

from 3.02.17 firmware, WebControl supports manually set a UROM value from outside, uromid from 1 to 4:
http://192.168.1.15/api/seturom.cgi?uromid=1&value=23456789

Command line browser wget can be used to do manually control. The above lines maybe need in double quotes to work.

## 6.8 External Analog Chip DS2450

4-1Q: I have DS2450 on the 1-wire bus, but reading is not stable?
4-1A: Maxim-IC DS2450 known for its sensitivity to noise. Please make sure to add 100pF disc capacitors across the chip on its VDD to ground, and from each analog input immediately to the ground, also. Any anlog input channel should have 1K input resistor to block some the incoming noise.

## 6.9 Power Supply Requirement

5-1Q: What kind of power required to run WebControl?
5-1A: WebControl 2.2.2 hardware can operate from 7.5 to 12VDC power supply. However, when using DC12V, the regulator will be VERY hot. It is recommended to run on 9VDC2A power supply. Please make sure power supply has good filtering capacitors. Any noise in the power supply could cause problem during execution of WebControl PLC logic. If a relay board is also used with WebControl, please make sure the power supply has enough reverse current to handle the spike during relay switching.