

WebControl™ PLC

User Guide

Version:	3.1.1
Hardware Version:	2.0.2
Firmware Version:	3.1.6
Date last modified:	7/17/2011

Table of Contents

1	Introduction	1
1.1	Scope.....	1
1.2	References	1
1.3	Table of Definitions	1
2	WebControl™ I/O.....	2
2.1	Serial Data Ports.....	2
2.2	Power Supply Inputs.....	3
2.3	TTL Output Port.....	3
2.4	AUX Input Port.....	3
2.5	DS1822 Temperature Sensor Input.....	3
2.6	Humidity Sensor Input	3
3	WebControl™ PLC Web GUI Configuration	1
3.1	Network Settings.....	5
3.2	Access Settings	5
3.3	I/O Setup.....	6
3.4	Email Notification Setup.....	7
3.5	General Setup.....	9
3.6	Reset Setup	9
4	AUX System Inputs.....	10
4.1	Digital Inputs	10
4.2	Analog Inputs.....	10
4.3	DS1822/DS18B20 1 Wire Temperature Sensors	11
4.4	Honeywell 4000 Series Relative Humidity Sensor.....	11
4.5	Timers.....	12
4.6	Direct Query System Inputs and Outputs	12
5	Real Time Clock.....	14
5.1	Network Requirements to Use the NTP.....	14
6	WebControl™ PLC Programming	15
6.1	The Basics of PLC Programming	15
6.2	WebControl™ PLC Instructions	16
6.3	WebControl™ PLC I/O Identifiers	19
6.4	WebControl PLC Examples	20
6.4.1	Example 1 Set Output based on condition	20
6.4.2	Example 2: Flash TTL output	21
6.4.3	Example 3: Push Button Input Control Output.....	22

6.4.4	Example 4: Send EMAIL	22
6.4.5	Example 5, Parallel I/O.....	23
6.4.6	Example 6, Sequential I/O.....	24
6.4.7	Example 7, Traffic Lights	25
6.4.8	Example 8, Time based Control	26
6.4.9	Example 9, Battery Charger	29
6.4.10	Example 10, RFID reader and browser Control	30
6.4.11	Example 11, Your Examples	31

Table of Figures

Figure 1.0 WebControl™ PCB inputs and outputs diagram	2
Figure 2.0 WebControl™ sensor connection	1
Figure 3.0 WebControl™ PLC system status.....	1
Figure 4.0 WebControl™ PLC Network configurable	5
Figure 5.0 I/O setup	6
Figure 6.0 WebControl™ Output Control	7
Figure 7.0 WebControl™ email setup	8
Figure 8.0 Network defaults	9
Figure 9.0 WebControl™ J12 Input pins	10
Figure 10.0 WebControl™ Temperature sensor assignments	11

1 Introduction

This document provides an overview of the technical aspects of using WebControl™ PLC. It describes the inputs and outputs offered by the PLC version of the WebControl™ hardware and firmware. WebControl PLC Programming Guide is in chapter 6 of this document. WebControl™ PLC is different from WebControl™ BRE. PLC version firmware provides greater flexibility in I/O control but also expects user having the knowledge to write assembly like PLC program. It can compare the input and output value, between analog inputs, digital inputs, or timers. WebControl can operate without network connection on its own. To assist writing PLC code is not included in the regular support for WebControl™ PLC configuration.

1.1 Scope

The scope of this document is to be a guide for configuring and using the features provided by WebControl™. 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 References

The following references are referred to throughout this document. It is expected that the reader will use these along with this document in order to understand and use WebControl™ PLC.

Reference	Description
Ref1	WebControl™ PLC Programming Guide (chapter 6)
Ref2	This guide for how to configure WebControl™ through web GUI
Ref3	Wiring diagram of the WebControl™

1.3 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://WebControl™
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.

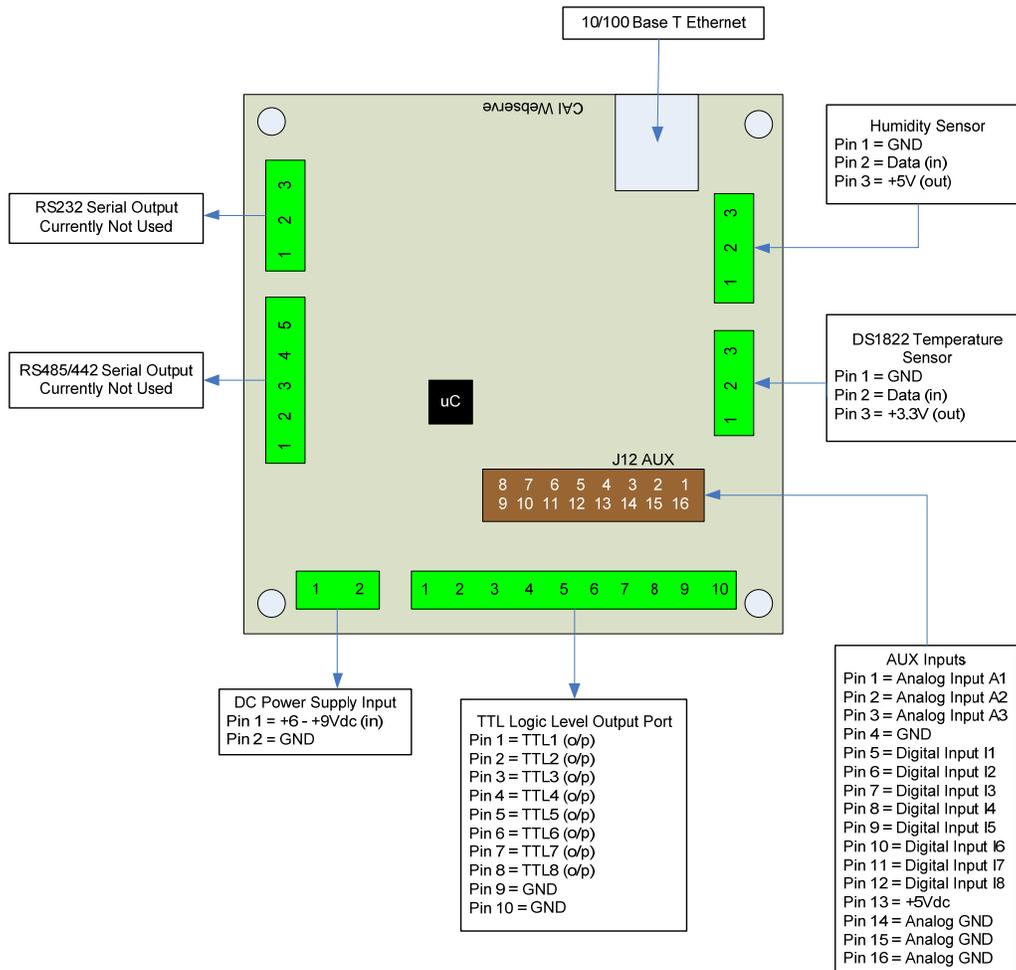


Figure 1.0 WebControl™ PCB inputs and outputs diagram

2.1 Serial Data Ports

Currently the RS232 and RS485/442 serial data ports provided by the hardware are not used by the firmware. The chip for those two protocols is not on the board.

2.2 Power Supply Inputs

The DC power supply input is the main DC supply to the board. The input voltage range is 9V DC. Any voltage greater than 12V applied to this input may overheat the board.

2.3 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 so that the particular output is set when a particular input condition is met. The maximum current that can be sourced or sunk 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 not be able to drive those coil based relays.

2.4 AUX Input Port

The AUX input connector offers digital and analog inputs that can be configured to set a TTL output upon a single input or a combination of inputs. 3 analog inputs are offered that have an input range of 0 – 10Vdc. 8 digital inputs are offered that use TTL input levels (0 and 5Vdc). The AUX input pin out diagram [ref3] shows the pin out of this port in more detail. Connector is TYCO ELECTRONICS - 1658622-3 .

2.5 DS1822 Temperature Sensor Input

The temperature sensor input allows up to eight Maxim DS1822/DS18B20 1-wire temperature sensors to be connected. These sensors can then display the temperature via the HTTP server or be used as inputs to the Boolean engine controlling the TTL output port. When DC supply voltage is too low, temp sensor may not have stable reading, due to the protection diode in the circuit. Please check your supply voltage when you noticed the temp sensor reading not correct.

2.6 Humidity Sensor Input

The current WebControl™ hardware has been designed to use the Honeywell 4000 series relative humidity sensor devices. Using any other humidity sensor with out 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 into the PLC.

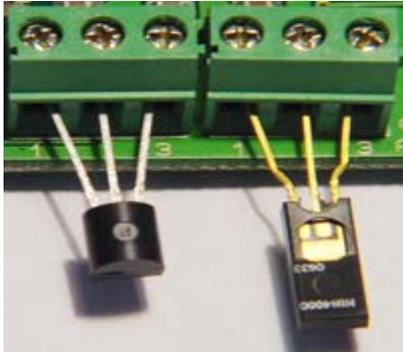


Figure 2.0 WebControl™ sensor connection

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

3 WebControl™ PLC Web GUI Configuration

To access WebControl™ PLC, connect WebControl™ to the network and power supply. The default IP address will be 192.168.1.15. If you enabled DHCP in your WebControl, please check your DHCP server log for which IP address assigned to the board. Each WebControl™ PLC board has its unique MAC address. Look in the HDCP log for matching MAC address you can find its IP address. From any computer with browser points to that IP address: <http://ww.xx.yy.zz> 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: v03.01.06 IpAddress: 192.168.1.15 Name: WEBCONTROL 06/29/2011 00:38:19

System Status

- System Status
- Output Control
- x10 Control
- I/O Setup
- Email Setup
- Temperature Sensor Setup
- PLC Program
- General Setup
- Network Setup
- Help

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	unbound							
Temperature Sensor Status	unbound							
Analog Inputs	0	1	0					
Humidity Sensor	0 %							

Refresh

Copyright © 2011 CAI Networks, Inc.

Figure 3.0 WebControl™ PLC system status

3.1 Network Settings

These are the basic settings that need to be configured in order for WebControl™ 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 2.0 shows the connection settings WebControl™ has.

CAI WebControl PLC

Version: v03.00.06	IpAddress: 192.168.12.15	Name: WEBCONTROL	06/29/2010 14:41:30
-----------------------	-----------------------------	------------------	---------------------

Network Setup

MAC Address: 00:22:12:A2:00:00

Network Configuration	
NetBIOS Name	webControl
Static IP Address	192.168.12.15
Subnet Mask	255.255.255.0
Gateway	192.168.12.251
Primary DNS Server	208.201.224.11
Secondary DNS Server	208.201.224.33
Username	admin
Password	password
DHCP Enabled	<input type="checkbox"/>
Web Login Enabled	<input checked="" type="checkbox"/>
HTTP Port	80

Access Limit	
IP Address 1	192.168.12.108
IP Address 2	1. 2. 3. 4
IP Address 3	2. 3. 4. 5
IP Address 4	3. 4. 5. 6
IP Address 5	4. 5. 6. 7
IP Address 6	5. 6. 7. 8
IP Address 7	6. 7. 8. 9
IP Address 8	7. 8. 9. 0

Send

Copyright © 2010 CAI Networks, Inc.

Figure 4.0 WebControl™ PLC Network configurable

3.2 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 board over the network. Please note that if any entry is 0.0.0.0, all IP addresses can access it.

3.3 I/O Setup

WebControl™ 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.

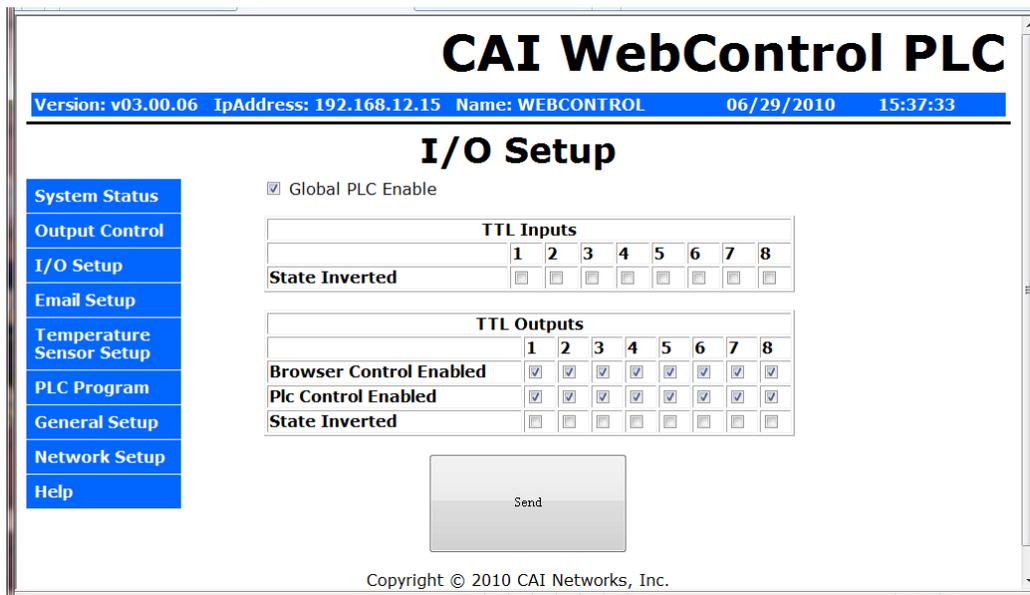


Figure 5.0 I/O setup

WebControl™ 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.

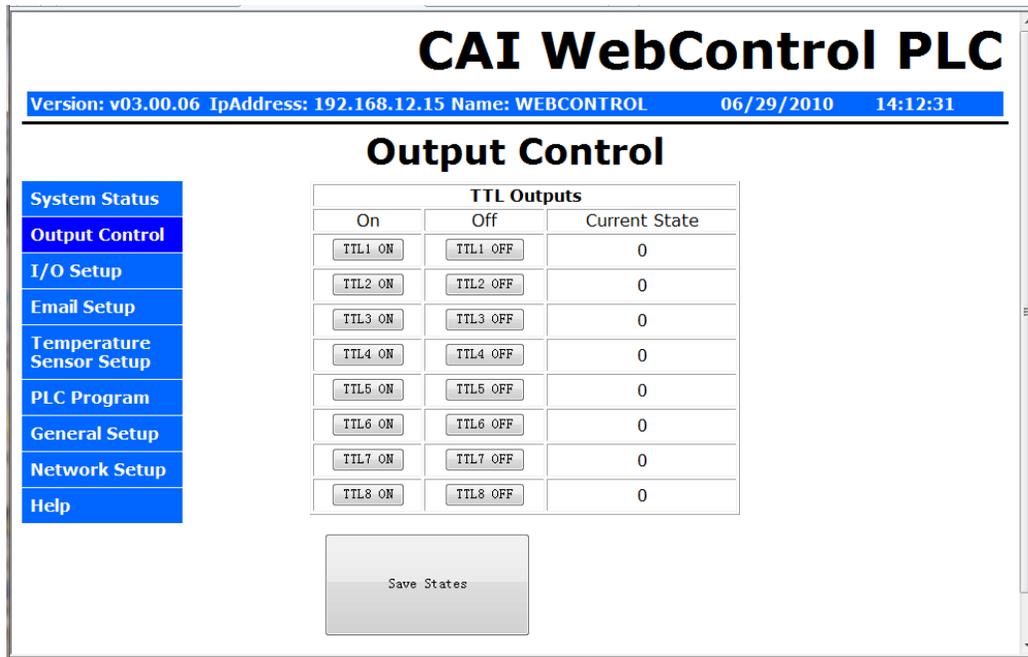


Figure 6.0 WebControl™ Output Control

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.

3.4 Email Notification Setup

WebControl™ 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 does not support that.

To many people the 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.

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

CAI WebControl PLC

Version: v03.01.06	IpAddress: 192.168.1.15	Name: WEBCONTROL	06/29/2011	00:09:24
--------------------	-------------------------	------------------	------------	----------

SMTP Email Setup

System Status
Output Control
x10 Control
I/O Setup
Email Setup
Temperature Sensor Setup
PLC Program
General Setup
Network Setup
Help

SMTP Host	<input type="text"/>	Port	<input type="text" value="25"/>
Username (leave blank if none)	<input type="text"/>		
Password (leave blank if none)	<input type="text"/>		
To Address	<input type="text"/>		
From Address	<input type="text"/>		

Message ID	Message Subject	Message Body
EM1	<input type="text"/>	<input type="text"/> <input type="button" value="Test"/>
EM2	<input type="text"/>	<input type="text"/> <input type="button" value="Test"/>
EM8	<input type="text"/>	<input type="text"/> <input type="button" value="Test"/>

Copyright © 2011 CAI Networks, Inc.

Figure 7.0 WebControl™ email setup

3.5 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.

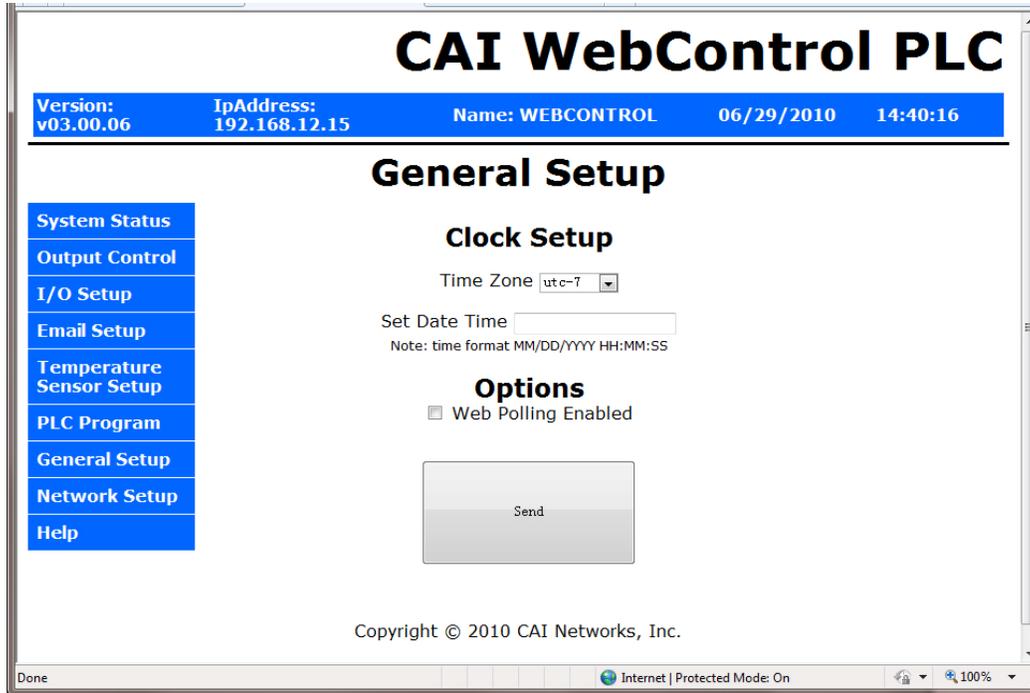


Figure 8.0 Network defaults

If the WebControl cannot reach to pool.ntp.org over Internet, it will use its own building clock with less accuracy. User may change and update the clock from this screen.

When Web polling enabled, WebControl™ 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.

3.6 Reset Setup

If the configuration was totally mess up, user can reset the board to factory default configuration by shorting the RESET holes while powering up the WebControl™. Reset holes on located between Ethernet port and RS-232 connector. Please note reset will wipe out all the 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 192.168.1.15 with DHCP disabled.

4 AUX System Inputs

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

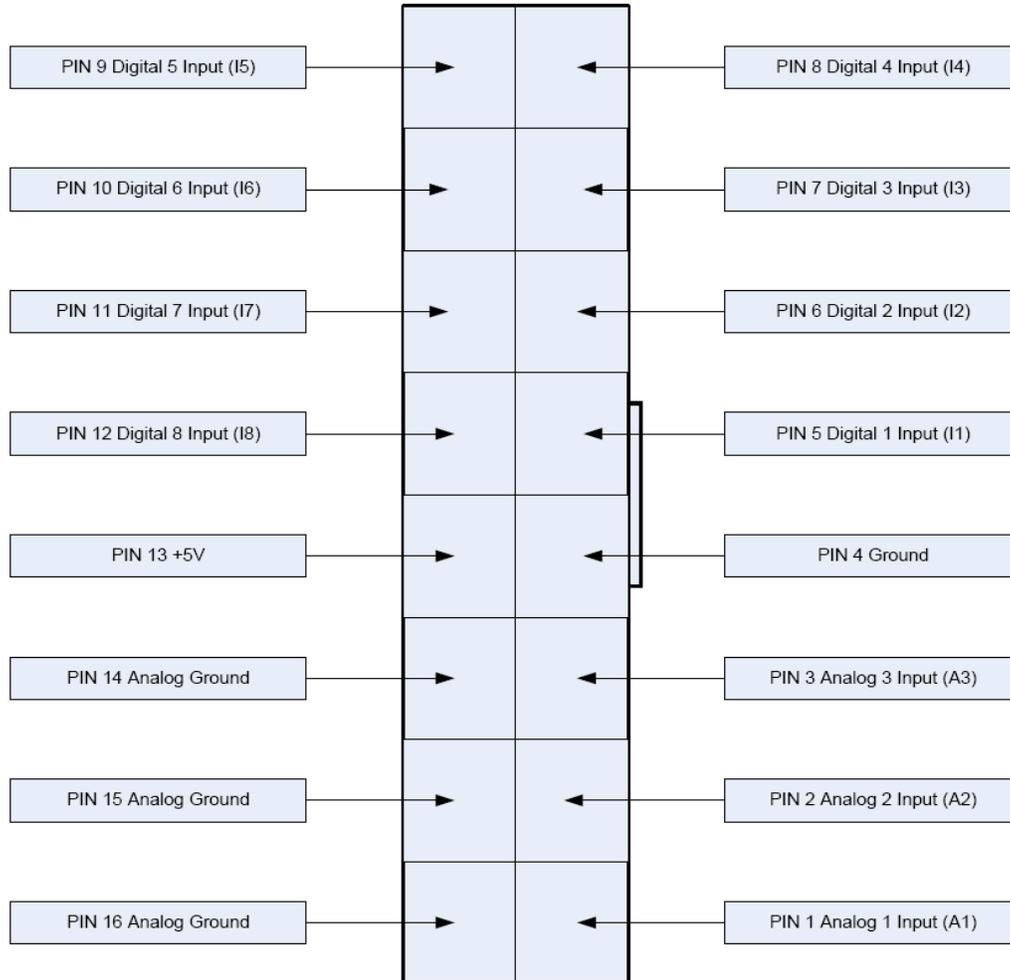


Figure 9.0 WebControl™ J12 Input pins

4.1 Digital Inputs

WebControl™ 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 or false case of the input. TTL level 0 is defined 0-1.75V; TTL level 1 is defined as 3-5V.

4.2 Analog Inputs

WebControl™ has three analog inputs each having an input voltage range of 0 to +10V. Each analog input can be configured to have an upper and lower threshold

which can then be used in the PLC logic. These upper and lower thresholds use the full scale range of 0 – 1023.

4.3 DS1822/DS18B20 1 Wire Temperature Sensors

WebControl™ 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.

Configured Sensors		
Sensor	ROM Code	Units
T1	28BFD5B70100	<input checked="" type="radio"/> °C <input type="radio"/> °F
T2	000000000000	<input type="radio"/> °C <input checked="" type="radio"/> °F
T3	28AB0DF60100	<input checked="" type="radio"/> °C <input type="radio"/> °F
T4	28AB0DF60100	<input type="radio"/> °C <input checked="" type="radio"/> °F
T5	000000000000	<input checked="" type="radio"/> °C <input type="radio"/> °F
T6	000000000000	<input checked="" type="radio"/> °C <input type="radio"/> °F
T7	000000000000	<input checked="" type="radio"/> °C <input type="radio"/> °F
T8	000000000000	<input checked="" type="radio"/> °C <input type="radio"/> °F

Send

Copyright © 2010 CAI Networks, Inc.

Figure 10.0 WebControl™ Temperature sensor assignments

Each temp sensor has unique ROM code. One temp sensor can be assigned for different sensor number. User can also select the unit as Centigrade or Fahrenheit. If sensor failed later due to sensor failure or wire problem, WebControl™ 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™ 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. CGI command codes are available to setup an upper and lower threshold for the humidity sensor that can then be used in the Boolean expressions.

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, that is an indication your power supply is not providing enough voltage to the board.

4.5 Timers

WebControl™ PLC 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, but the PLC code space of total 4000 line of code and variables can be used at the same time are limited.. WebControl PLC has build-in timer value for comparison:

- ❑ 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

WebControl timer variables are in one milli-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.

4.6 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

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

5 Real Time Clock

WebControl™ has a build-in clock functions that is kept accurate by using an atomic clock via the simple network timer protocol (SNTP). Configuration is required to set the correct time zone of the clock. CGI commands are provided to do this. When NTP is not available, WebControl™ 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.

5.1 Network Requirements to Use the NTP

WebControl™ must be connected to a network that has access to a DNS server. The primary and secondary DNS server addresses of WebControl™ are required to be configured correctly. WebControl™ 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™ PLC Programming

The WebControl™ 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™ PLC uses assembly like PLC language. It 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™ 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
```

If sub routines are used then they are listed after 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 program return address stack depth of 8.

The program control block has a zero bit that is updated implicitly on most instructions. This zero bit can also be used implicitly when using branch and call instructions. 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™ 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 the delay operator value is in unit of 1mS..

Opcode	Operands	Description
START		Start of main program
TSTEQ	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.
TSTNE	a[] b[] (d[])	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.

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.
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.
OR	a[] b[] (d[])	Logical OR's a with b and optionally puts boolean 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.
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-F, b: device code 0-F, c: ON, OFF, BRIGHT, DIM
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 next below.

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		
IP3		
IP4		
IP5		
IP6		
IP7		
IP8		
AIP1	Analog Inputs 1...3	Valid range 0 - 1024
AIP2		
AIP3		
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	
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
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	
VAR5	
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	
RAM8	

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 $T3 - T2 \geq 20$ 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 implemented, so that it 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
    GOTO start

```

END

checkOP1:

```
TSTGT T3 500 OP1
RET
```

```
checkOP2:
  TESTEQ IP1[300] 1 OP2
  RET
```

```
checkOP3:
  AND AIP1 AIP3 RAM1
  TSTGT RAM 1024
  BNZ I1
  TSREQ IP4 1
  BNZ I2
  RET
```

```
I1:
  SET O3 1
  RET
```

```
I2:
  SET O3 0
  RET
```

```
checkOP4:
  TSREQ 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
I1:
  TST IP1 1
```

```

        BZ I1
        SET OP1 1
        SET OP4 0

I2:
    TSTGT T3 250 RAM1
    AND OP1 RAM1
    BZ I2
    SET OP2 1
    EMAIL EM1

I3:
    TSTEQ OP2[1000] 1
    BZ I3
    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

This example will have five subroutines. WebControl 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 F ON 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 o1 check_b1
    cnz o2 check_b2
    cnz o3 check_b3
goto loop
end
check_b1:
    BNZ RAM1 c1
    tstle A1 100 RAM1
    bz e1
c1:
    tstgt A1 125
    bnz e1
    set o1 0
    set o4 1
e1:
ret
check_b2:
    BNZ RAM2 c2
    sub A2 A1 RAM4
    tstle RAM4 100 RAM2
    bz e2
c2:
    sub A2 A1 RAM4
    tstgt RAM4 125
    bnz e2
    set o2 0
    set o5 1

```

```

e2:
ret
check_b3:
    BNZ RAM3 c3
    sub A3 A2 RAM4
    sub RAM4 A1 RAM4
    tstle RAM4 100 RAM3
    bz e3
c3:
    sub A3 A2 RAM4
    sub RAM4 A1 RAM4
    tstgt RAM4 125
    bnz e3
    set o3 0
    set o6 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, Your Examples

The use of WebControl in Industrial automation, office automation and home automation are almost endless. You can submit your example to us for sharing it with other users.