

i.CanDoIt[®] Guided Tour Modbus TCP Servers



i.Board i.CanDoIt[®]
*For flexible, powerful
Remote Monitoring & Control*

*Embedded Web Server
Email Event Notifications
Fill-in-the-blank Alarm Templates
Field Programmable
User Web Pages*

i.CanDoIt[®] Guided Tour Table of Contents

- [1. Background](#)
- [2. Device Overview](#)
- [3. I/O Configuration](#)
- [4. Viewing Data](#)
- [5. Calculations and I/O Cascade](#)
- [6. Data Trending and Plotting](#)
- [7. Thresholds or Event Rules](#)
- [8. Data Logging](#)
- [9. Event Logging](#)
- [10. Email notifications](#)

[11. XML Configuration Files](#)

[12. Script Basic Programming](#)

[13. Internet Network Configuration](#)

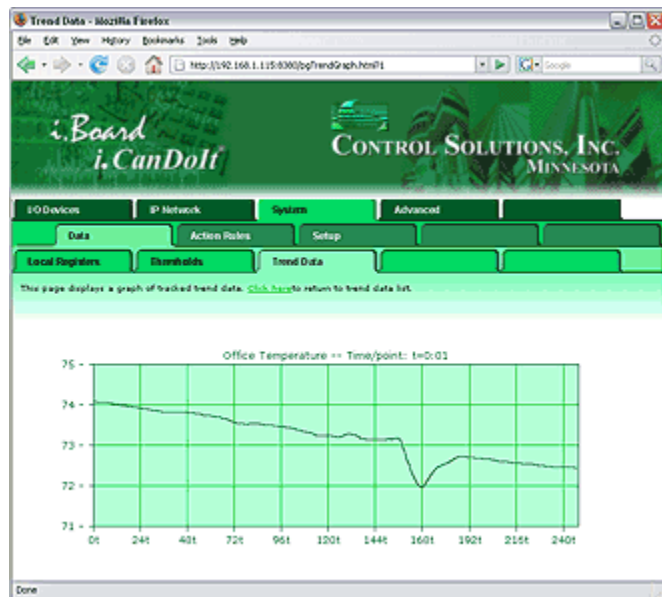
[14. Modbus Gateway](#)

[15. SNMP Agent](#)

[16. User HTML/JavaScript and CGI](#)

i.CanDoIt® Guided Tour, part 1 Background

[**\(return to index\)**](#)



i.Board™ i.CanDoIt® Background

The goal of **i.CanDoIt®** is to provide a simple and cost effective facility management and remote monitoring solution suitable for use in small sites. The **i.CanDoIt®** has no site licenses or installation fees. The only software tool needed is a web browser. The **i.CanDoIt®** is programmable, even though it is rather powerful without programming.

Conceptually, **i.Board™ i.CanDoIt®** is a big spread sheet. It is register oriented just like Modbus, even though the registers are accessible in ways other than Modbus, such as SNMP. When data is provided by SNMP, or obtained from local I/O, the data is placed into generic registers that are universally accessible to features of the system.

i.Board™ i.CanDoIt® includes template or rule based alarm processing with email notification and/or I/O activation in response to events. The resulting register manipulation may generate network communication with remote devices, control local I/O, or change setpoints for example.

i.Board™ i.CanDoIt® is user programmable using Script Basic. Once a program has been created, it can be selected as the auto-boot program, meaning this program will automatically run every time i.Board boots up.

The **i.CanDoIt®** web server includes a large number of predefined web pages for configuration and monitoring. Support is also provided for user defined web pages using HTML and JavaScript. Any of the data registers are accessible to user defined web pages via internal CGI processing of user tags. This allows a fully customized look

and feel for the system.

i.CanDoIt® Guided Tour, part 2 **Device Overview**

[\(return to index\)](#)



The original "i.Board" embedded web servers have been renamed Models IB-100 and IB-110. The **Model IB-100** includes 10 I/O points (8 inputs, 2 outputs). The FET outputs drive 24VDC at 1 amp. The analog/universal inputs can be configured for 0-10VDC, thermistor, discrete or dry contact. The inputs will also accept 4-20mA with an external dropping resistor. The **Model IB-110** includes 8 Input points plus an RS-485 port with Modbus RTU support.

i.Board™ Features a 32-bit ARM Processor

The **i.Board™** is Control Solutions' answer to customers who said, "We like your products, but can we get something in a smaller, low cost package?" So for all those applications where you need the sophistication of an Internet Appliance but only have a handful of points to monitor, i.Board is the answer.

Input data is presented in scaled integer and floating point format. Linearization tables are provided for 10K and 20K type II, III, and IV thermistors on analog inputs. Input and output type and scaling are configured in web pages served by the internal web server.

Hardware Details

• 8 Analog/universal inputs

- - - 0-10VDC, Thermistor, dry contact
- - - Software selectable input type
- - - 10-bit resolution
- - - 2 channels capable of pulse count input

• 2 Discrete outputs

- - - Open drain FET, 1A current sink
- 10/100BaseT Ethernet (Modbus TCP, SNMP, HTTP, email)
- Event notification via email, SNMP trap, HTTP Get
- Web interface including user HTML wrapper & CGI processing
- User programmable via Script Basic
- Powered by 24VDC (only)

- Power Consumption:
 - - - 0.15A @ 24VDC max.
- 38mm H x 52mm W x 103mm D
- Operating temperature -40°C to +85°C
- Humidity 5% to 90%
- FCC Class B, CE Mark

i.CanDoIt®* Guided Tour, part 3** ***I/O Configuration

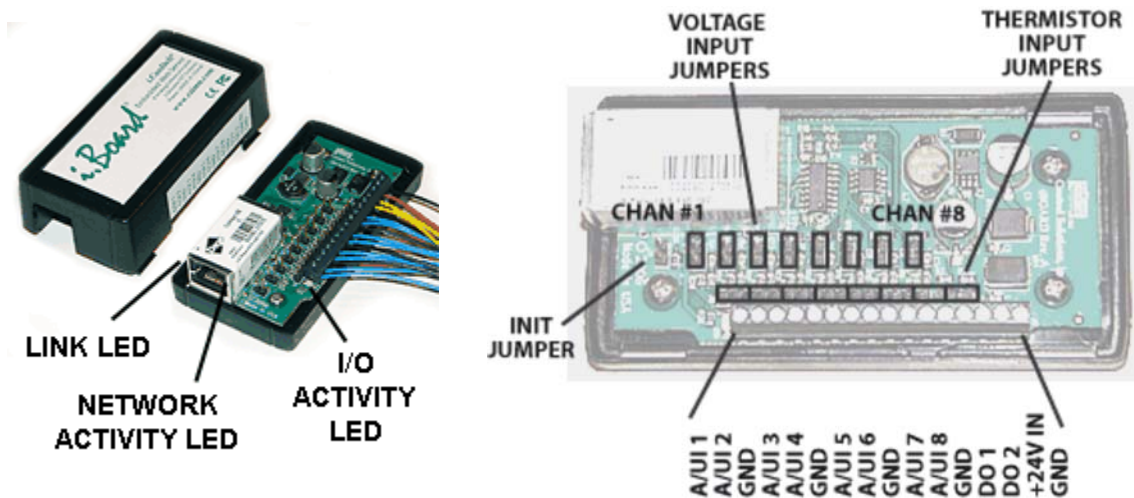
[\(return to index\)](#)

This page displays analog input hardware configuration.

Show data Show calibration helper Showing 1 to 8 of 8 Update

Register	Type	Scaling	Slope	Intercept	Threshold (%)	Register Name
0001/1001	Analog	10k II (F)	1.000000	0.000000	0	Temperature
0002/1003	Analog	0-10V	1.000000	0.000000	0	Sensor 2
0003/1005	Discrete	0-10V	1.000000	0.000000	0	Sensor 3
0004/1007	Analog	10k III (F)	1.000000	0.000000	0	
0005/1009	Discrete	10k II (F)	1.000000	0.000000	50	Switch 1
0006/1011	Discrete	20k IV (F)	1.000000	0.000000	50	Switch 2
0007/1013	Analog	10k III (C)	1.000000	0.000000	0	
0008/1015	Analog	10k II (C)	1.000000	0.000000	0	

Model IB-100 (originally i.Board)



Configuration of the IB-100 requires selecting one of two jumper positions for each input, and then selecting the corresponding input type in software via the web page. The voltage jumper position is used for voltage or 4-20mA input, or discrete input. The thermistor jumper position is used for thermistors, and also for dry contact input. The thermistor jumper applies a voltage to the input terminal via a pullup resistor. A contact closure to ground will produce a change of state when software is configured for dry contact input.

Configure an **Analog Input** for temperature by selecting the appropriate thermistor type. Configure 0-10V input by selecting 0-10V, and entering a slope and intercept that corresponds to your sensor. If using a 4-20mA sensor, you will need a dropping resistor connected in parallel across the sensor input. If you wish to use an input as a simple on/off input, select discrete or dry contact.

A calibration helper page is built in. Simply enter data from readings you have taken, or data from the sensor spec sheet, then enter the desired display range. The helper will calculate slope and intercept for you. It will even take resistor values into account when a dropping resistor is used. Simply click on the calibration helper link at the top of the Analog Inputs configuration page.

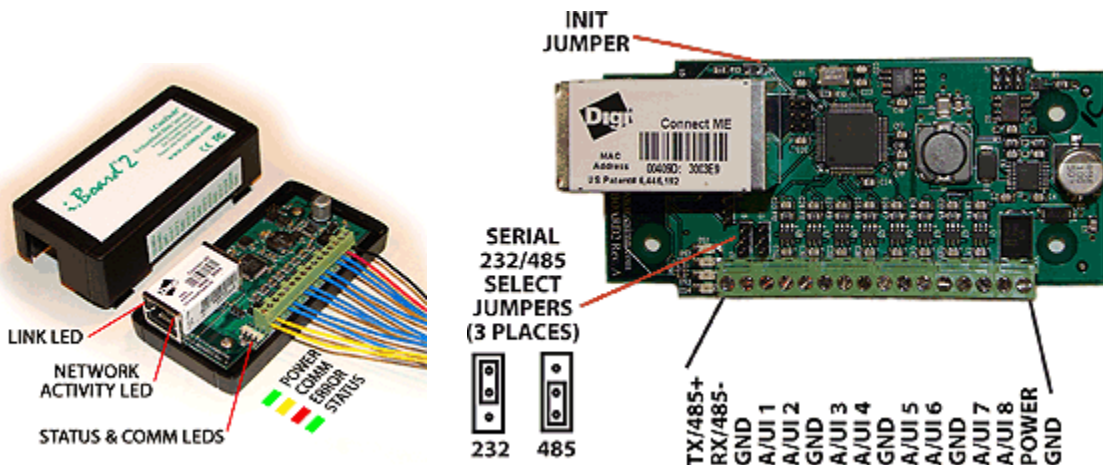
The difference between discrete and dry contact is this: With a discrete input, it is expecting you to provide external voltage excitation (up to 24VDC). With a dry contact input, it provides its own excitation, and you simply provide a contact closure to ground.

Discrete Outputs have an optional timer associated with them. The timer can do one of two things: (1) generate a pulsed output, or (2) impose a minimum on time qualification to prevent rapid cycling.

Model IB-110 (originally i.Board2)

The IB-110 inputs are configured by software controlled FET switches. There are no physical jumpers to move.

Select RS-485 or RS-232 for the serial port by positioning 3 jumpers as illustrated. Do not use the INIT jumper unless instructed.



i.CanDoIt® Guided Tour, part 4 ***Viewing Data***

[**\(return to index\)**](#)

Data may generally be viewed relative to its source in addition to a global system wide view. Data from I/O points may be viewed under the I/O Devices tab. Data from Modbus registers may be viewed under one or more Modbus tabs, etc. The following illustration shows analog input data. Note that for physical I/O points, the data is displayed in both integer and floating point form.

Register	Int Data	Float Data	Register Name
0001/1001	7516	75.16000	Temperature
0002/1003	0	0.0	Sensor 2
0003/1005	0	0.0	Sensor 3
0004/1007	0	0.0	
0005/1009	1	10.00000	Switch 1
0006/1011	1	10.00000	Switch 2
0007/1013	0	0.0	
0008/1015	0	0.0	

Raw data from local I/O is stored in the integer registers. Scaled data is stored in the corresponding floating point registers. Integer registers are numbered 1 to 999. Floating point registers are numbered from 1001 to 1999 and are allocated 2 registers per floating point value consistent with Modbus practice. The first 10 of each data type are reserved for physical I/O on i.Board. The number of reserved registers varies in other models according to the number of physical points.

The system wide global view of data is found under the System tab. This shows data from I/O points, and includes data retrieved from remote devices via Modbus or other local area networks. It also includes data generated by user programs, or written into the register by a remote client (e.g. Modbus TCP client).

I/O Devices		IP Network		System		Advanced			
Data		Action Rules		Setup					
Local Registers		Thresholds		Trend Data					
This page displays data as presently found in the local registers maintained by this device.									
Showing registers from <input type="text" value="1001"/> <input type="button" value="Update"/> <input prev"="" type="button" value("<=""/> <input type="button" value="Next >"/>									
Local Register #	Register Name	Hex	Update	Register Data	Unsigned	Register Type	Default Data	Server Timeout (S)	
01001	Sensor #1	<input type="checkbox"/>	<input type="checkbox"/>	5.016400	<input type="checkbox"/>	Flt*	0.0	0.0	
01003	Office Temperature	<input type="checkbox"/>	<input type="checkbox"/>	74.90000	<input type="checkbox"/>	Flt*	0.0	0.0	
01005	Sensor #3	<input type="checkbox"/>	<input type="checkbox"/>	5.016023	<input type="checkbox"/>	Flt*	0.0	0.0	
01007	Sensor #4	<input type="checkbox"/>	<input type="checkbox"/>	5.015269	<input type="checkbox"/>	Flt*	0.0	0.0	
01009	Switch #1	<input type="checkbox"/>	<input type="checkbox"/>	10.00000	<input type="checkbox"/>	Flt*	0.0	0.0	

i.CanDoIt® Guided Tour, part 5 ***Calculations, I/O Cascade***

[\(return to index\)](#)

There is a tab called "Action Rules" under System. Some of these actions provide very simply programming without any programming. For example, you can do simple logic using the Calculate rules.

Suppose you want to turn on an output when any of several switches are tripped. The following Calculate rule will logically OR registers 6 through 9 and place the result in register 28. We would need to configure analog inputs 6 through 9 to be discrete or dry contact, with the invert bit set appropriately so that we get a "1" when the switches are closed. Writing "1" to register 28 (on AddMe III) will turn on relay #4. Inputs 1 through 8 can be OR'ed on i.Board, and written to registers 9 or 10 to turn on the FET outputs on i.Board.

Rule #	Perform Operation	Using Register #	And/Through	This Register #	Place Result in Register #
1	logic OR	6	thru	9	28

Another simply but useful action rule is the Cascade rule. Suppose you want other outputs to turn on as a result of one particular output coming on. The following example will copy register 25 to registers 26 and 27 any time register 25 changes. In this AddMe III example, it means when relay #1 turns on, relays 2 and 3 will also come on automatically. Similar examples can be implemented on i.Board.

Rule #	Source Register #	Destination Register #	Last Destination Register # in Range
1	25	26	27

There is also a rule named Constant. This rule simply writes a fixed value to a specified register one time at system startup (or re-initialization). This can be used for establishing initial setpoints, etc.

i.CanDoIt® Guided Tour, part 6 ***Data Trending & Plotting***

[\(return to index\)](#)

Data trending provides two main functions: (1) Track minimum, maximum, and average, and (2) graph the trend over some recent period.

The min-max-average trending is meaningful when trending is being logged via HTTP Get to a remote server. At the end of each tracking period, the data is logged, and then trend values are reset. If no logging is being done, the min-max-average values will have little meaning since they are essentially "average forever" values. However, the graph is still useful without logging because it is always a sliding window of actual data samples.

Trending is set up under the System->Action Rules->Trending tab. You simply select which data point you wish to track and identify it by register number. Then allocate destination registers for the trend results. You can use the auto-allocate button for this.

This page allows setup of min/max/average tracking of selected registers.

Showing 1 to 1 of 1 Update < Prev Next >

Tracked Register #	Register Name	Average Register	Min. Register	Max. Register	Reset	Period HH:MM	Slices	Flash
1003	Office Temperature	1071	1073	1075	When logged	0:01	10	<input type="checkbox"/>

Trend data may be viewed under the System->Data->Trend Data tab. The values that will be logged next if logging is enabled will appear here. You have the option of resetting the values. The illustration below shows that the trend has just been reset, in which case minimum, maximum, and average are the same value.

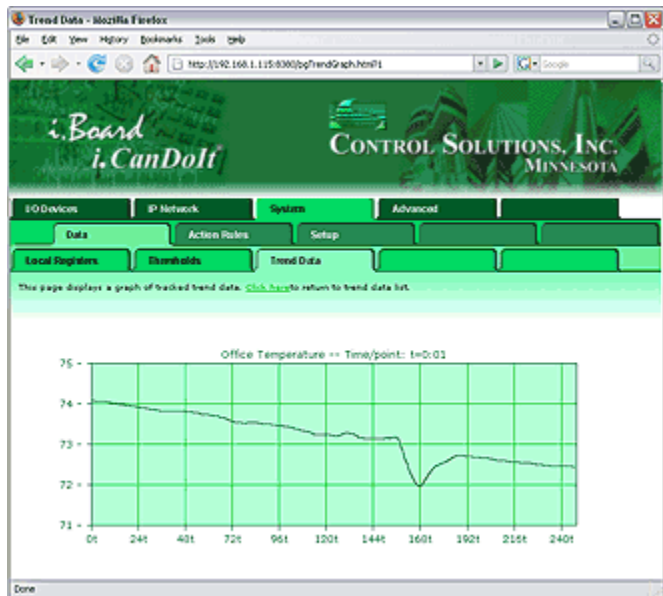
This page displays trend data that has been tracked for the registers indicated.

Showing 1 to 1 of 1 Update < Prev Next >

Tracked Reg #	Register Name	Average Value	Min. Value	Max. Value	Reset	Graph Status
1003	Office Temperature	75.19000	75.19000	75.19000	<input type="checkbox"/>	0 Graph

The most useful link on the Trend Data page is the Graph link on the far right. Click on

this link, and the respective register's sliding window of data samples will be displayed graphically.



i.CanDoIt® Guided Tour, part 7 ***Thresholds or Event Rules***

[\(return to index\)](#)

What are Threshold Rules? What are Events? Where are the Alarms?

These questions all point to essentially the same answer. You build an alarm by defining a threshold rule. When the threshold rule is triggered or activated, we call this an event. There are several things you can do with an event, and you may treat any of them as an alarm. An event may result in turning specific I/O on or off. An event may result in a notification message being sent (e.g., via email to your cell phone). An event may send a message to a central server which then decides what to do. An event may generate an SNMP trap.

How do I set up an Alarm?

I/O point data is placed in "registers" by easily configured I/O. Threshold "rules" determine what constitutes an event. The image below is a screen shot of a threshold rule that will result in an event when the sensor on input #7 exceeds a level of 1000. Data values are scaled to any units you decide. You find the threshold rules in the System->Action Rules->Thresholds page.

Read local source register # for this event named

Event is TRUE if the value is this value: this local register:

Qualified by this hysteresis value: this minimum On Time: this minimum Off Time:

How do I prevent spurious events when the test value is hovering around the threshold?

Hysteresis will prevent spurious events near the threshold. In the following example, the event will occur when register 7 reaches a level of 1000. If register 7 drops below 1000 by a margin *less than 50*, then returns to above 1000, the event will not repeat. The value of register 7 must drop below 950 and return to over 1000 before the trap will repeat.

Read local source register # for this event named

Event is TRUE if the value is this value: this local register:

Qualified by this hysteresis value: this minimum On Time: this minimum Off Time:

How do I generate an event only after the condition has existed for a minimum time?

The on/off time qualifications are used to add time to the criteria. The following example shows a minimum on time of 30 seconds. This means the data in register 7,

most likely placed there by sensor hardware, must exceed a level of 1000 for at least 30 seconds before an event will be generated. If the minimum off time was also set, the rule must test false for that amount of time before it can repeat the "true" event.

Read local source register # for this event named

Event is TRUE if the value is this value: this local register:

Qualified by this hysteresis value: this minimum On Time: this minimum Off Time:

i.CanDoIt[®] Guided Tour, part 8 ***Data Logging***

[**\(return to index\)**](#)

Data may be logged using HTTP Get requests to a server that has been configured to log data in cooperation with one or more i.Board devices. The initial log rate is set in the Network configuration page. Once logging begins, the server will determine how often data is reported. The advantage of this type of data logging is that it requires no special on-site configuration of network routers.

Register #	Register Name	HTTP Report
00001	Temperature	<input checked="" type="checkbox"/>
00002	Sensor 2	<input checked="" type="checkbox"/>
00003	Sensor 3	<input checked="" type="checkbox"/>
00004		<input type="checkbox"/>

i.CanDoIt[®] Guided Tour, part 9 ***Event Logging***

[\(return to index\)](#)

The HTTP Get logging to a remote server can also be configured to provide event reports as they occur. This type of data logging really means i.Board is opening a web page on a designated server and entering data on the page it finds there.

Rule #	Event Name	HTTP Report
1	Low Temp Alarm	<input checked="" type="checkbox"/>

The "Low Temp Alarm" example checked off for event logging above was set up with a threshold rule like shown below. It is not necessary to include a destination register in the threshold rule when the rule's only purpose is logging or notification.

Read local source register # for this event named

Event is TRUE if the value is this value: this local register:

Qualified by this hysteresis value: this minimum On Time: this minimum Off Time:

i.CanDoIt® Guided Tour, part 10

Email Notifications

[\(return to index\)](#)

One of the useful things you can do with an event defined by a "threshold rule" is send an email to notify someone that something has happened. You may define "recipients" which can be one or more email addresses. You select or define the message, and then pick which events cause this email to be sent.

This page allows you to identify email Recipients and select the message format they will receive when Events trigger an email alert message.

Showing 1 of 3 Update < Prev Next >

Recipient: Send Test

Subject:

Default Email Message:
 This is an automated email alert originating from <system name> at <system location> at <time/date>. The event named <event name> with a test criteria of <rule> a value of <test value> tested <true/false> for register <n> <register name> with a value of <value>.

Include standard message (default or cell phone)

Cell Phone Message (160 char. max.):
 <system name>/<system location>/<register name>/<value>=<event name>

Send Status: (0 = No Error)

Recv Status:

Reset Errors

Each threshold rule that has been defined under System->Action Rules->Thresholds is listed on the email events page. Each event may be checked off for delivery to any of the recipients. You have the option of sending the email when the event first occurs (notify on true) and when the event clears (notify on false).

The email handling capability of i.CanDoIt includes the ability to receive email as well. If you would like email messages to be repeated until somebody acknowledges the event, check the "Expect Ack" box and enter a repeat time. To acknowledge the event, you only need to "Reply To" the email you received. The mail will contain a coded acknowledge string generated automatically when the email is sent. If the reply contains this expected reply string, the event is considered acknowledged.

Rule #	Event Name	Notify on True	Notify on False	Email Recip 1	Email Recip 2	Email Recip 3	Expect Ack	Repeat Time DD,HH:MM
1	Low Temp Alarm	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0,0:00"/>

Use of email requires that you have an email account available to i.CanDoIt. You must have a user name and password for the account. Most email servers will require that

you have an email address that matches the account information. You need to configure the SMTP and POP3 server names (or IP addresses if fixed). You should also be aware that many email servers do not allow sending email from outside of their own domain. You will need to work with your ISP, or network or IT representative, to get email configured properly.

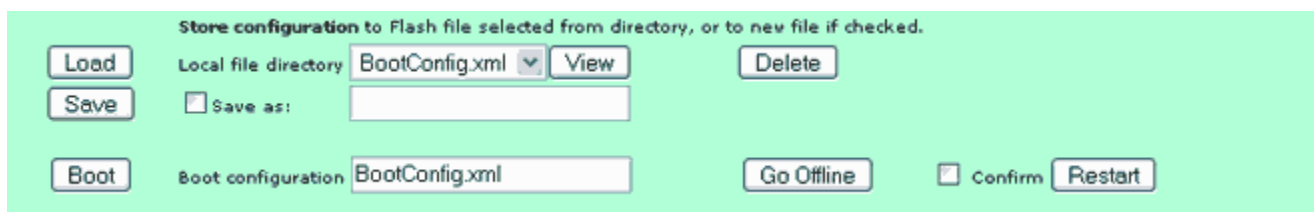
Recipients	Events	Email Server		
This page tells the system where to log in to send and receive emails.				
<hr/>				
<hr/>				
SMTP IP Address: <input type="text" value="0.0.0.0"/> <input type="checkbox"/> Use Authentication <input type="button" value="Update"/>				
SMTP Host Name: <input type="text" value="smtp.comcast.net"/> (send email) <input type="button" value="Network"/>				
SMTP Port: <input type="text" value="25"/> SMTP Domain: <input type="text"/>				
POP3 IP Address: <input type="text" value="0.0.0.0"/>				
POP3 Host Name: <input type="text" value="mail.comcast.net"/> (receive email)				
User Name: <input type="text" value="icandoitonline"/>				
Password: <input type="password" value="XXXXXXXXXX"/>				
<hr/>				
System Name: <input type="text" value="AddMe III"/> <input type="button" value="Update"/>				
System Location: <input type="text" value="White Bear Lake, Minnesota"/>				
System Contact: <input type="text" value="icandoitonline@comcast.net"/>				
Note: System contact must be email account name to use email notification.				

i.CanDoIt® Guided Tour, part 11 ***XML Configuration Files***

[\(return to index\)](#)

Most configuration information is stored in an XML file in the device's internal Flash file system. The only exceptions are things like the device's IP address, the name of the XML file from which to load configuration at boot-up, and things needed to get as far as loading that file in the first place.

Since configuration is stored in XML format, it is easily ported from one system to another. It is also easy to review the configuration - simply click on the file name, and your browser will display XML. You load a configuration file from your PC, or save it back to your PC.



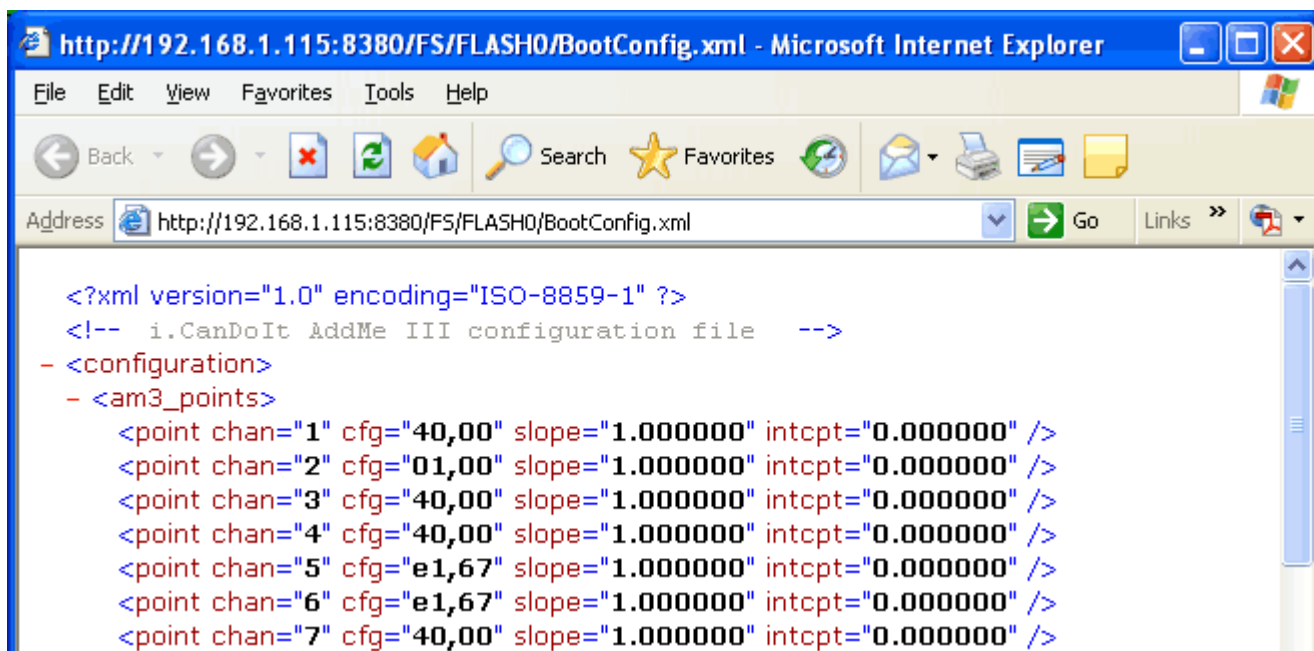
Store configuration to Flash file selected from directory, or to new file if checked.

Load Local file directory: BootConfig.xml View Delete

Save Save as: []

Boot Boot configuration: BootConfig.xml Go Offline Confirm Restart

The first few lines of XML viewed in the browser look like this... (AddMe III example shown here, but i.Board is identical in format.)



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- i.CanDoIt AddMe III configuration file -->
- <configuration>
- <am3_points>
  <point chan="1" cfg="40,00" slope="1.000000" intcpt="0.000000" />
  <point chan="2" cfg="01,00" slope="1.000000" intcpt="0.000000" />
  <point chan="3" cfg="40,00" slope="1.000000" intcpt="0.000000" />
  <point chan="4" cfg="40,00" slope="1.000000" intcpt="0.000000" />
  <point chan="5" cfg="e1,67" slope="1.000000" intcpt="0.000000" />
  <point chan="6" cfg="e1,67" slope="1.000000" intcpt="0.000000" />
  <point chan="7" cfg="40,00" slope="1.000000" intcpt="0.000000" />
```

i.CanDoIt[®] Guided Tour, part 12 ***Script Basic Programming***

[\(return to index\)](#)

The i.Board includes Script Basic for user programming of control algorithms. Access to I/O points and all data registers is available from Basic. You can access remote devices connected via Modbus TCP from your Basic program.

This Basic implementation includes the usual program structures including FOR..NEXT, IF..THEN..ELSE, WHILE..DO, etc. You can write callable functions. All standard math, algebra, and trigonometry functions are included.

An extended optional feature of Script Basic is the ability to open sockets and do string manipulation. This makes it possible to do HTTP based interaction with remote devices beyond the HTTP based data and event logging support already provided.

For more extensive information about Script Basic, download the compiled help package available on the product page for IB-100/IB-110. A summary of Script Basic programming follows.

Script BASIC Help Summary

The following summary provides an overview of most-used features. Script BASIC (SB) supports string variables and string operations, but these are of minimal use for the register based programming most typical for this product, and are therefore not discussed here. It should also be noted that file handling is not currently supported in SB. Some examples are shown in all upper case. Script BASIC is not case sensitive, therefore keywords may be upper or lower case. This also applies to variable names. Therefore, myvar and MYVAR are the same variable.

LET Statement

The "LET" keyword is archaic, and not used in SB. Line numbers are also not used. Simply begin an assignment line with the variable you wish to assign. For example:

```
OurAverage = (YourVariable + MyVariable) / 2  
A = B + C  
phi = sin (theAngle)
```

Register Access

You may read any of the local registers using the getreg function, and write them using the setreg command. Note that getreg is a function while setreg is not; it is a command and therefore requires no parenthesis. Consider the following example:

```
MyData = getreg (22)  
MyData = MyData * 2.5
```

```
setreg 24, MyData
```

The above example will place the contents of register #22 in the variable MyData, then multiply it by 2.5, then place that value in register #24. Variables may be used in place of the constants used in the above example.

Time Functions

The pre-defined variable NOW will provide the present real time in seconds since January 1, 1970 (Unix style) if real time clock support is present, or seconds since boot-up otherwise. Saving the value of NOW in a variable and calculating the difference some time later will allow measurement of real time. You use NOW in the same manner as any other variable except that you cannot assign a new value to NOW. (Use the System->Setup->Time & Date web page to set the internal real time clock if available.)

The function SLEEP will suspend program execution for some number of seconds, or generate a delay. Examples:

```
sleep (5)
sleep (0.25)
```

The first example will pause program execution for 5 seconds. The second example will pause for a quarter of a second.

Math Functions & Operators

The following is a summary of math operators and functions recognized in SB:

+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Integer Division
%	Modulus
^	a^b produces a raised to the power of b
ABS(n)	Returns absolute value of 'n'
ACOS(n)	Calculates arc cosine of 'n'
ASIN(n)	Calculates arc sine of 'n'
ATAN(n)	Calculates arc tangent of 'n'
COS(n)	Calculates cosine of 'n'
FIX(n)	Returns integral part of argument, rounding toward zero, i.e., int(-3.3) = -3
FRAC(n)	Returns the fractional part of the argument
INT(n)	Returns integral part of argument, rounding down, i.e., int(-3.3) = -4
LOG(n)	Produces the natural logarithm of n

LOG10(n)	Products the logarithm of n
PI	Produces the constant pi
POW(n)	Produces 10 raised to the power of n
RND	Provides a random number (use RANDOMIZE(n) to seed random number generator)
SIN(n)	Calculates sine of 'n'
SQR(n)	Calculates square root of 'n'
TAN(n)	Calculates tangent of 'n'

Logic Operators

The following is a summary of logic operators recognized in SB:

AND	Logical AND - bitwise for values, or logical in "if" statements
NOT	Logical NOT - bitwise for values, or logical in "if" statements
OR	Logical OR - bitwise for values, or logical in "if" statements
XOR	Logical Exclusive OR - bitwise for values

Be sure to use parenthesis to establish precedence as necessary.

IF THEN ELSE Statement

There are two different ways to use this command: single line IF and multi line IF. (IF/THEN and other keywords are shown in upper case in these examples for clarity; however, they may be either upper or lower case.)

A single line IF has the form

```
IF condition THEN command
```

There is no way to specify any ELSE part for the command in the single line version. If you need the ELSE command you have to use the multi line IF.

The multi line IF should not contain any command directly after the keyword THEN. It should have the format:

```
IF condition THEN
  commands
ELSE
  commands
END IF
```

The ELSE part of the command is optional, thus the command can have the format

```
IF condition THEN
  commands
END IF
```

The `condition` is any valid comparison or expression. Examples include:

```
if a > b then print "greater"  
if a <> b then print "not equal"  
if getreg(3) then print "is enabled"
```

Contitional operators are:

=	Equal
<>	Not equal
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal

FOR NEXT Statement

Implements a FOR loop. The variable var gets the value of the start expression exp_start, and after each execution of the loop body it is incremented or decrement by the value exp_step until it reaches the stop value exp_stop.

```
FOR var= exp_start TO exp_stop [ STEP exp_step]  
  ...  
  commands to repeat  
  ...  
NEXT var
```

The STEP part of the command is optional. If this part is missing then the default value to increment the variable is 1.

If

- the expression exp_start is larger than the expression exp_stop and exp_step is positive or if
- the expression exp_start is smaller than the expression exp_stop and exp_step is negative

then the loop body is not executed even once and the variable retains its old value.

When the loop is executed at least once the variable gets the values one after the other and after the loop exists the loop variable holds the last value for which the loop already did not execute.

DO WHILE Statement

This command implements a looping construct that loops the code between the line DO WHILE and LOOP until the expression following the keywords on the loop starting line becomes false.

Practically this command is same as the command WHILE with a different syntax to be compatible with different BASIC implementations.


```
do while
  ...
loop
```

You can use the construct that creates more readable code.

WHILE Statement

Implements the 'while' loop as it is usually done in most basic implementations. The loop starts with the command while and finished with the line containing the keyword wend. The keyword while is followed by an expression and the loop is executes so long as long the expression is evaluated true.

```
while expression
  ...
  commands to repeat
  ...
wend
```

The expression is evaluated when the looping starts and each time the loop is restarted. It means that the code between the while and wend lines may be skipped totally if the expression evaluates to some false value during the first evaluation before the execution starts the loop.

In case some condition makes it necessary to exit the loop from its middle then the command GOTO can be used.

DO UNTIL Statement

This command implements a looping construct that loops the code between the line DO UNTIL and LOOP until the expression following the keywords on the loop starting line becomes true.

```
DO UNTIL expression
  ...
  commands to repeat
  ...
LOOP
```

The expression is evaluated when the looping starts and each time the loop is restarted. It means that the code between the DO UNTIL and LOOP lines may be skipped totally if the expression evaluates to some TRUE value during the first evaluation before the execution starts the loop.

This command is practically equivalent to the construct

```
WHILE NOT expression
  ...
  commands to repeat
```

```
...  
WEND
```

You can and you also should use the construct that creates more readable code.

User Functions

The FUNCTION command is used to define a function. A function is a piece of code that can be called by the BASIC program from the main part or from a function or subroutine.

```
FUNCTION fun(a,b,c)  
...  
fun = returnvalue  
...  
END FUNCTION
```

The end of the function is defined by the line containing the keywords END FUNCTION. This function would be called by the line `x=fun(a,b,c)` and the result would be placed in 'x'.

The SUB command should be used to define a subroutine. A subroutine is a piece of code that can be called by the BASIC program from the main part or from a function or subroutine.

```
SUB sub(a,b,c)  
...  
END SUB
```

The end of the subroutine is defined by the line containing the keywords END SUB.

Note that functions and subroutines are not really different in ScriptBasic. ScriptBasic allows you to return a value from a subroutine and to call a function using the command CALL. It is just a convention to have separately SUB and FUNCTION declarations. You would call this subroutine with the line `CALL sub`.

Labels and GOTO Statement

The statement GOTO is the most famous statement of all BASIC languages. Many program theorists say that you should not ever use GOTO and they may be right. Even so, the statement GOTO is part of most programming languages and ScriptBasic is no exception.

Using the statement GOTO you can alter the execution order of the statements. GOTO statements use labels and the labels can identify program lines. The form of a GOTO statement is

```
GOTO label
```

where the label is a label, which is defined somewhere in a program.

Labels are local within functions and subroutines. You can not jump into a subroutine or jump out of it. Labels begin at the start of a line, are the only thing on the line, and end with a colon.

```
GOTO mylabel
...
commands
...
mylabel:
...
```

Use of GOTO is usually discouraged and is against structural programming. Whenever you feel the need to use the GOTO statement (except ON ERROR GOTO) think twice whether there is a better solution without utilizing the statement GOTO.

Typical use of the GOTO statement to jump out of some error condition to the error handling code or jumping out of some loop on some condition.

END Statement

End of the program. Stops program execution.

You should usually use this command to signal the end of a program. Although you can use STOP and END interchangeably this is the convention in BASIC programs to use the command END to note the end of the program and STOP to stop the program execution based on some extra condition inside the code.

Variables and Arrays

Variables may be any name that starts with a letter, and after the first letter, may also contain digits, underscore, and dollar sign (for old style denotation of string variables). Variables are automatically typed as integer, double (floating point), or string according to context in which they are used, and may dynamically change type in the course of program execution.

Arrays are created automatically as soon as you use subscripts. Uninitialized elements of the arrays return "undef". Array subscripts use square brackets (not parenthesis which are easily confused with function calls). For example:

```
a[1] = 45.9
a[2] = 99.8
a[3] = "something else"
a[4,12] = 10
```

These are all valid elements of the same array. Normally the index values must be integer. However, SB also supports associative arrays. Associative arrays use "curly" brackets. For example:

```
animal{"cat"} = "Garfield"  
animal{"dog"} = "Snoopy"  
animal{"tiger"} = "Tony"
```

Print Statement

You may use a Print statement to show output in the virtual terminal window. Although difficult to work with via HTML, you can also do a LINE INPUT statement in the input window of the virtual terminal when the program is run in the foreground (RUN button). Only the output window is available when running in the background (Spawn button), and no part of the virtual terminal is available to the auto run program. An example of a print statement might be:

```
print "Register #5 is ", getreg(5), "\n"
```

An example of the line input statement is:

```
line input myline$
```

The use of "\$" as part of a string variable name is optional, and would be included only for readability. Any valid variable name may contain an integer, a floating point value, or a string, and will be dynamically typed according to the context.

Rem Statement

Remarks or comments are any line starting with REM or a single quote. Everything to the end of the line will be ignored. Remarks may not be added to the end of a valid line of code.

[\(return to index\)](#)

i.CanDoIt[®] Guided Tour, part 13 ***Network Configuration***

[\(return to index\)](#)

If you spend much time looking at this page, you're a network administrator - either by training or by default because you had to. If you're not the network administrator, this is the one page you do want that person to look at on your behalf if you are going to use any of the more advanced network features.

The only thing most people need to think about on this page is simply setting the IP address of the device. It comes defaulted to 10.0.0.101. The two most common domains for DSL and cable routers to be shipped with by default are 10.0.0.0 and 192.168.1.0, so we had only half a chance of picking the right one for you. Refer to the quick start guide for tips on getting up and running for the first time. (Additional help is shown at the bottom of each page in the server.)

I/O Devices	IP Network	System	Advanced
Data	Action Rules	Setup	
Config File	Network	User	
This page allows you to change this device's IP address, and optionally set other network parameters including DNS.			
IP Address	<input type="text" value="192.168.1.149"/>	192.168.1.149	<input type="button" value="Review IP"/> Email Server
Subnet Mask	<input type="text" value="255.255.255.0"/>	255.255.255.0	<input type="button" value="Change IP"/>
Gateway	<input type="text" value="192.168.1.1"/>	192.168.1.1	
Primary NTP Server	<input type="text" value="64.236.96.53"/>	Secondary NTP Server	<input type="text" value="68.216.79.113"/>
Daylight Time Start Rule	<input type="text" value="3.2.0/02:00:00"/>	Daylight Time End Rule	<input type="text" value="11.1.0/02:00:00"/>
Standard GMT Offset	<input type="text" value="-360"/> Minutes	Daylight GMT Offset	<input type="text" value="-300"/> Minutes <input type="button" value="Set NTP"/>
NTP Refresh Period	<input type="text" value="60"/> Minutes		
Current Local Time	04/18/1974 11:04:28 <input type="button" value="Refresh"/>		
SNMP Community	<input type="text" value="private"/>	SNMP Enabled	<input type="button" value="Set SNMP"/> <input type="button" value="Reload SNMP"/>
SNMP Get/Set Port	<input type="text" value="161"/>	Trap Port	<input type="text" value="162"/>
SNMP Table Sizes	<input type="text" value="1000"/> Integers	<input type="text" value="500"/> Floating Point Registers	
Static DNS1	<input type="text" value="75.75.75.75"/>	75.75.75.75	<input type="button" value="Apply DNS"/> <input type="button" value="Reset DNS"/>
Static DNS2	<input type="text" value="75.75.76.76"/>	75.75.76.76	
Proxy Server	<input type="text"/>		<input type="button" value="Set Proxy"/>
Proxy Port	<input type="text" value="0"/>		
Dynamic DNS Service	<input type="text" value="None"/> 0.0.0.0	DDNS status:	No DDNS configured.
Host Name	<input type="text"/>		
DDNS User Name	<input type="text"/>	Password	<input type="text"/>
HTTP Port	<input type="text" value="80"/> (default 80) <input type="checkbox"/> Disallow HTTP Query		<input type="button" value="Set Ports"/>
Modbus Server Port	<input type="text" value="502"/> (default 502)		
Telnet Port	<input type="text" value="23"/> (default 23)		
Client Update Host	<input type="text" value="logmydata.net"/> <input type="checkbox"/> Enable		<input type="button" value="Update"/>
Log Page	<input type="text" value="/fireport.php"/>		
Log Parameters	<input type="text"/> (optional)		
Configuration Page	<input type="text"/>		
Update Timeout	<input type="text" value="60"/>	Client ID	<input type="text" value="514149"/>

To change the IP address of this device, enter the address, subnet mask, and gateway, then click "Change IP". Set the IP address to 255.255.255.255 to specify that DHCP should be used to obtain an IP address upon power-up. IP address change will take effect upon next power cycle (or when Restart is clicked on the Config File page). The "Refresh" button only updates the information shown on this page, it does not submit a DHCP renewal request. DHCP renewal is automatic.

NTP setup: Enter a primary and secondary IP address of NTP server, such as 24.56.178.140 for www.nist.gov (go to <http://tf.nist.gov/tf-cgi/servers.cgi> to find more). Enter daylight start/end rules, and offset from GMT for both standard and daylight time. Offset is a negative number in the western hemisphere. Enter an NTP update time in minutes. Do not set NTP to update too frequently or you risk being denied service by the NTP server. Click the Set NTP button after all settings have been made. The Flash update will take several seconds.

Daylight savings time start/end rules consist of "date/time" where the date (m.n.d) indicates the day when summer time starts or ends, and time (hour:min:sec) is the current local time when summer time starts/ends. The date portion of the rule is formatted as follows:

- m indicates the month ($1 \leq m \leq 12$)
- n indicates which week of the month ($1 \leq n \leq 5$). 5 = the last week in the month.
- d indicates what day of the week ($0 \leq d \leq 6$). 0 = Sunday

For example: Start "4.1.0/02:00:00", end "10.5.0/02:00:00" means summer time starts at 2am on the first Sunday in April and ends at 2am on last Sunday in October. That was the old US rule. The new US rule is start "3.2.0/02:00:00" and end "11.1.0/02:00:00", which is start at 2am on the second Sunday in March, end at 2am on the first Sunday in November.

If SNMP is being used, you need to provide a community name. This name acts as a simple password for any SNMP manager that wishes to write to MIB variables maintained by this device. Anybody can read-only with the community name of "public". You must be the root user to change the SNMP community name. (Updating the name will take several seconds.)

SNMP can be completely turned off by setting the community name to a null name (zero length, empty name). To restore SNMP after being disabled, a non-null community name must be entered. The enable/disable change can only take effect upon the next system restart.

SNMP table lengths default to including all registers. If it is desired to limit the table size, change the table sizes, click "Set SNMP" to save the size settings, then click "Reload SNMP" to reload the tables. Changes will also take effect upon each subsequent system restart. It should also be noted that the event table size will be calculated automatically at startup. If new threshold (event) rules are defined, they will not be included in the SNMP table until "Reload SNMP" is used. The table load, if full, will take half a minute or so.

The default port for web page serving is 80. If you wish to change it, enter the port number and click Set Port. This change will take effect upon the next power-up. If the port is anything other than 80, you must include the port number in the URL. For

example, if you would normally use `http://10.0.0.101/` to get here and you change the port to 8215, you would now use `http://10.0.0.101:8215/`. You cannot disable the web port. If set HTTP to port 0, it will revert back to HTTP port 80. (Note: The port change is accepted only if you are logged in as the root user.)

You may also reassign the default Modbus and Telnet ports, as well as default SNMP ports. You may also disable these ports and their services by entering zero. If you will not use the service, it is a good idea to disable the service for better security.

You may use domain names instead of static IP addresses in several instances. If domain names are used, you must supply the IP address of at least one DNS server here. The DNS server must be at a static IP address. These changes take effect immediately. Note: If you are using DHCP, the DNS addresses will be supplied by the DHCP server and should be set to 0.0.0.0 here.

If the device is functioning as an HTTP client behind a proxy server, enter the proxy host name or IP address and its HTTP port in the proxy entries. Set the proxy port to zero to indicate no proxy is being used.

The numbers shown to the right of input windows are the actual numbers in the system. For example, entering an IP address of 255.255.255.255 causes the system to acquire a dynamic IP address via DHCP. The IP address shown to the right is the currently leased IP address. When DHCP is used, static DNS entries are ignored, and the actual DNS entries provided by DHCP are shown.

If you are using a Dynamic DNS service to keep track of this device on the Internet, select the service type, enter the host name you have told that service about for this device, and enter the user name and password associated with that update service. Click "Apply DNS" to register this information, and be sure to save the configuration file to retain this data indefinitely.

Client update requires a special server. HTTP Client Update is a method of reporting data from a remote site with no on-site configuration other than obtaining Internet access. Refer to www.logmydata.com for information about services for using this feature.

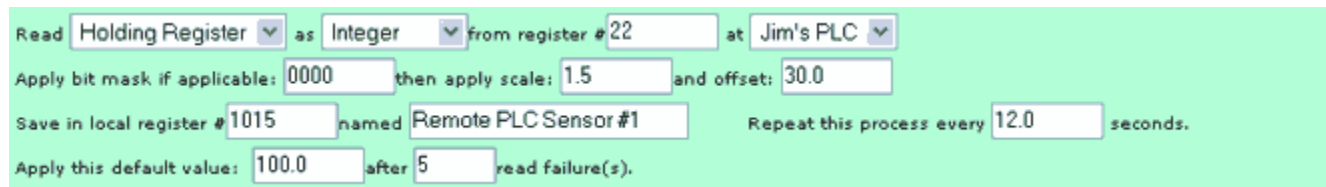
i.CanDoIt® Guided Tour, part 14 ***Modbus Gateway***

[\(return to index\)](#)

The Modbus TCP gateway is capable of being both client and server. As a client, it will poll other Modbus TCP devices. As a server, other Modbus TCP devices may poll i.CanDoIt. The server map even allows i.CanDoIt to mimic another device by replicating its register map.

The IB-110 also includes a Modbus RTU Master. Set the serial port baud rate, etc, on the RTU Port Setup page (under I/O Devices), and set up read maps and write maps in the same manner as setting up the maps for Modbus TCP.

The remote device "read rule" mainly says what remote register to read and where to put the data. But you have the option of scaling data as it comes in. You also have the option of applying a default value if the read fails.



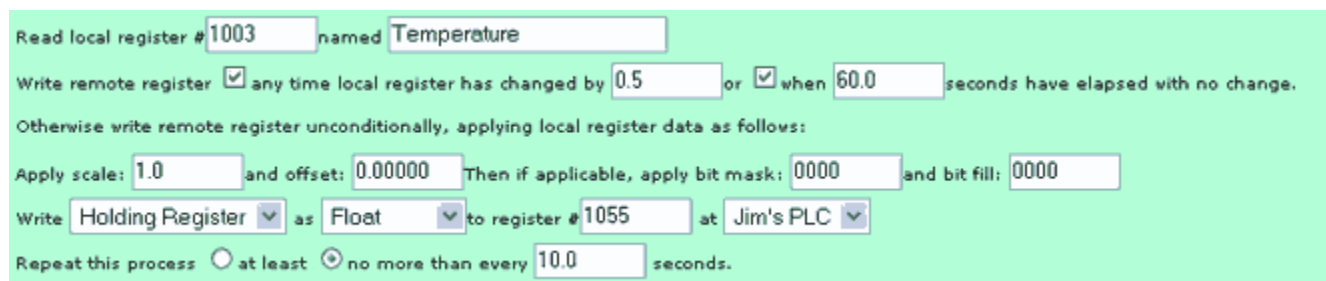
Read as from register # at

Apply bit mask if applicable: then apply scale: and offset:

Save in local register # named Repeat this process every seconds.

Apply this default value: after read failure(s).

The remote device "write rule" allows setting up a "send on delta" to only write the register to the remote device when the local data has changed. You can also send periodically, with optional data scaling on the way out.



Read local register # named

Write remote register any time local register has changed by or when seconds have elapsed with no change.

Otherwise write remote register unconditionally, applying local register data as follows:

Apply scale: and offset: Then if applicable, apply bit mask: and bit fill:

Write as to register # at

Repeat this process at least no more than every seconds.

The server map allows you to effectively relocate registers as seen by an outside Modbus TCP client (master). This allows the i.Board to replace existing Modbus TCP devices without rewriting the front end software that wants to access the data. If this server map is "exclusive", this means the external client can only see registers listed in this map. If non-exclusive, this map is checked first, but if the requested register is not found, a second check is made to see if it exists in the set of registers already known to i.Board locally.

Showing to 1 of 1

Map #	Mapped Register #	Mapped Register Format	Local Register #	Scale Factor	Offset	Bit Field	Fill	Name
1	<input type="text" value="3005"/>	<input type="text" value="Floating Point"/>	<input type="text" value="1003"/>	<input type="text" value="1.00000"/>	<input type="text" value="0.00000"/>	<input type="text" value="0000"/>	<input type="text" value="0000"/>	<input type="text" value="Temperature"/>

Custom Registers Enabled: User Map Enabled Map is Exclusive
 Swap Double Registers Zero fill null registers

Modbus Server is Enabled on Port 502 (see [Network Configuration](#))

i.CanDoIt® Guided Tour, part 15 ***SNMP Agent***

[\(return to index\)](#)

The i.Board is equipped with an SNMP agent. This agent allows accessing data points using SNMP Get and Set requests. The OIDs assigned to each Modbus register may be seen on the MIB View page under System -> Data. The agent also generates SNMP v2c traps based on events defined by the threshold rules.

There may be several trap destinations. Each device may be identified by static IP address, or by host name if DNS access is provided. Each device may be assigned to any of three groups. Each event may direct a trap at any of the three groups of devices. If trap on true is checked and a non-zero repeat time is given, the traps will continue to repeat at that rate as long as the event is active or "true". The initial report will provide the data as of the event first being triggered, but subsequent reports will contain real time data which may be different then the value that first triggered the event.

Rule #	Event Name	Trap on True	Trap on False	Repeat Count	Repeat Time	Enable Group 1	Enable Group 2	Enable Group 3
1	High Temperature	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0	0.00000	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How do I set up an SNMP trap?

I/O point data is placed in "registers" by easily configured I/O drivers. Each register is accessible by standard SNMP Get and Set requests. Threshold "rules" determine when traps are sent. The image below is an actual screen shot of a threshold rule that will result in a trap when the sensor on input #7 exceeds a level of 1000. Data values are scaled to any units you decide. You find the threshold rules in the System->Action Rules->Thresholds page.

Read local source register # for this event named

Event is TRUE if the value is this value: this local register:

Qualified by this hysteresis value: this minimum On Time: this minimum Off Time:

The SNMP Devices page allows setting the IP address or Host Name of one or more SNMP managers that v2c traps are to be sent to. Traps may be repeated periodically for as long as the rule tests true. Trap data includes the register name and number, data value that caused the trap, the event name, test type, and test threshold.

Devices Trap Enable Identity

This page sets up the network address and group membership for a remote SNMP manager which traps will be sent to.

Device #

IP Address Local Name:

Domain Name

Group Membership Group 1 Group 2 Group 3

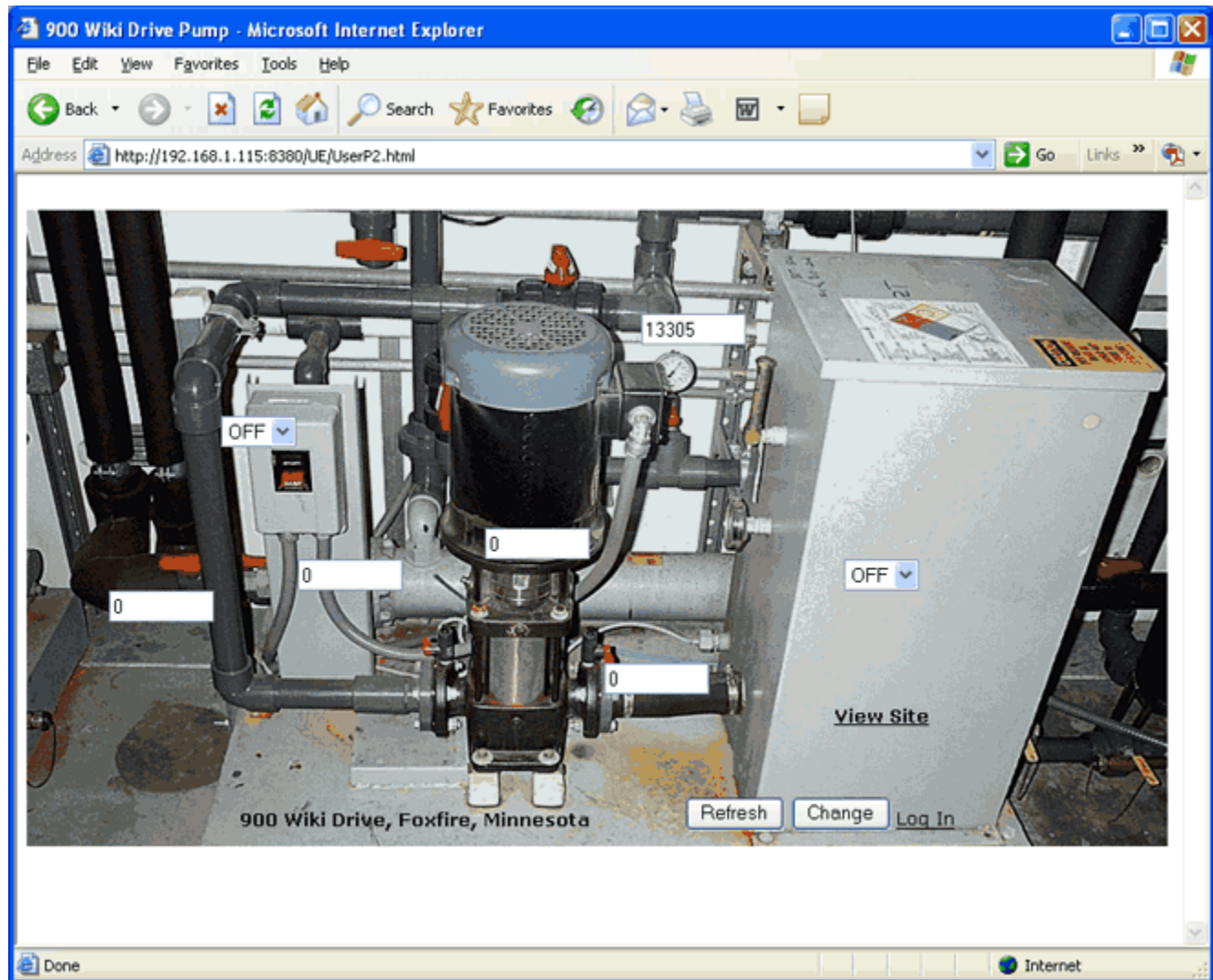
Once you have the Threshold Rule set up, you need to go to the IP Network->SNMP Setup->Trap Enable page and check off whether to trap on true, false, or both, and also which group(s) to send the trap to. The following is what you would see for the above rule:

Rule #	Event Name	Trap on True	Trap on False	Repeat Count	Repeat Time	Enable Group 1	Enable Group 2	Enable Group 3
1	High Level	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0	0.00000	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

i.CanDoIt® Guided Tour, part 16 ***User HTML***

[\(return to index\)](#)

You have the ability to put user friendly web pages in front of the configuration pages. If user HTML has been loaded, logging into the device will pull up the user pages instead of the default index page. There may be multiple linked user pages. An example user page is shown here.



Data types supported by the CGI processor include text and select option inputs. Dynamic, live access to data is by way of register references used as the name of the input object. Input windows can be defined as read-only. If they are not restricted, any new data entered by the user will be written into the registers.