

i.CanDoIt[®] Guided Tour Modbus TCP Servers



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

*Embedded Web Server
Data Logging & Trending
Time & Date Scheduling
Astronomical Clock
Email Event Notifications
Fill-in-the-blank Alarm Templates
Field Programmable
User Web Pages*



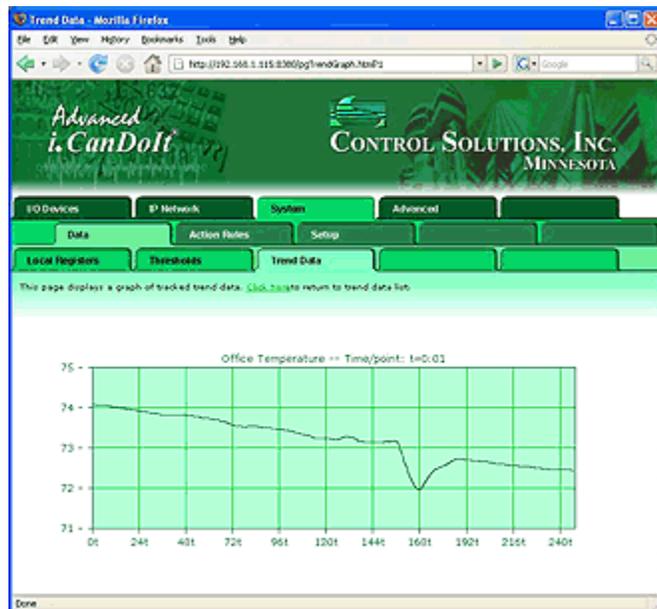
i.CanDoIt[®] Guided Tour Table of Contents

- [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. Time & Date Scheduling](#)
- [12. XML Configuration Files](#)
- [13. PL/i Programming](#)
- [14. Internet Network Configuration](#)
- [15. Modbus/TCP Gateway](#)
- [16. Modbus/RTU Gateway](#)
- [17. SNMP Agent](#)
- [18. User HTML/JavaScript and CGI](#)

i.CanDoIt® Guided Tour, part 1 **Background**

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



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. Multi-protocol support for BACnet, Modbus, and LonWorks are included. The **i.CanDoIt®** is programmable, even though it is rather powerful without programming.

Control Solutions pondered one particular question while doing conceptual design for **i.CanDoIt®**. Why, with the availability of advanced protocols like LonWorks, is Modbus still so popular? The answer is simple: Modbus is simple. While providing support for interface of LonWorks devices, we took a lesson from Modbus and decided that we should keep **i.CanDoIt®** simple.

Conceptually, **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. When data is obtained from LonWorks or BACnet devices, or from local I/O, the data is placed into generic registers that are universally accessible to features of the system.

i.CanDoIt® includes data logging, time & date scheduling, and template or rule based alarm processing with email notification and/or I/O activation in response to events. Data logging simply records the contents of registers on a user specified schedule. Scheduling causes register values to be set according to a schedule. The resulting register manipulation may generate network communication with remote devices, control local I/O, or change setpoints for example.

i.CanDoIt® has two levels of programmability. It runs user programs at the server level on a 32-bit ARM processor, and at the I/O level on a high performance 8-bit RISC processor directly connected to I/O. The programming language is **PL/i**, patterned after PL/1 at the source level, and executing byte code like Java at the execution level. The goal of PI/i was to create a compact compiler with a reasonably safe execution environment, and a syntax with some of the power of C while being simpler than C.

The register oriented spread sheet model of **i.CanDoIt®** makes it possible for users to write PL/i programs that are independent of protocol. The same program can run on a LonWorks, BACnet or Modbus network with no changes when only register references are used in the program.

The **PL/i** compiler tools are full self contained within i.CanDoIt and accessible via a browser. Programs can be compiled and sent over the BACnet, Modbus, or LonWorks network to other AddMe III devices that are not web enabled. PC based support tools for working with larger user programs, including external compiler, are free (download from this web site).

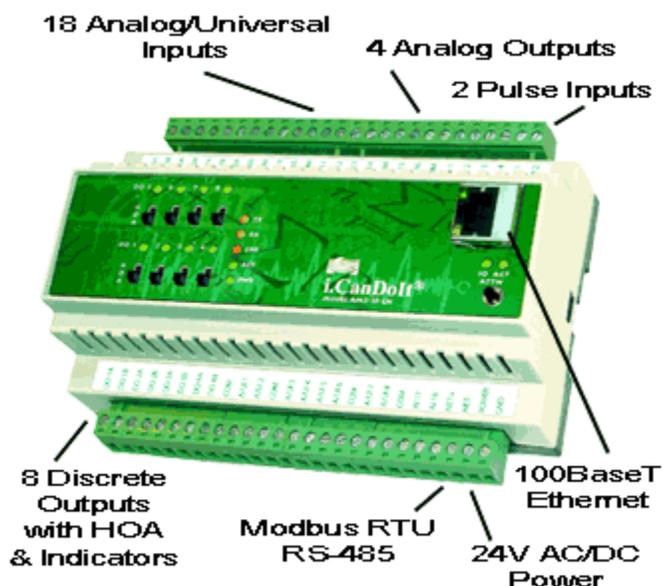
The **i.CanDoIt®** web server includes a large number of predefined web pages for configuration and monitoring. Support is also provided for user defined web pages using HTML and JavaScript. Any of the data registers are accessible to user defined web pages via internal CGI processing of user tags. This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

***i.CanDoIt®* Guided Tour, part 2**

Device Overview

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

The top of the line AddMe III with Advanced *i.CanDoIt®* includes 32 I/O points. AddMe Junior has 14 I/O points. The Model AM3-IP-MB is shown here, and features a Modbus TCP client and server.



***i.CanDoIt®* Features a Dual Processor Architecture**

An ARM7 (32-bit RISC) processor is dedicated to server and supervisory functions, including web server, data logging, scheduling, email notification, etc. A PIC18 processor (high performance 8-bit RISC) is dedicated to hardware interface and control.

Hardware Details

- **18 Analog/universal inputs**
 - 0-10VDC, Thermistor, dry contact
 - Software selectable input type
 - 0.1% reference, up to 16-bit resolution
 - Continuously self-calibrating sigma-delta converter
- **4 Analog outputs**
 - 4-20mA (0-20mA)
 - 8-bit resolution
- **8 Discrete outputs**

- Form A relay, 5A @ 120VAC
- HOA switches & LED status indicators
- **2 Pulse/Discrete inputs**
 - TTL to 24VDC
 - Rate or totalizing count input
- **Modbus RTU Gateway Built In**
- 10/100BaseT Ethernet
- Data and event logger, email & trap notifications
- Real time scheduler including astronomical clock
- Web interface including user HTML wrapper
- User programmable using PL/i
- Powered by 18-30VDC or 24VAC 50/60 Hz
- Power Consumption: 0.3A @ 24VDC max.
- DIN rail mounting, 100mm H x 155mm W x 60mm D
- LED indicators for power, status, Tx/Rx, comm error
- Pluggable screw terminal blocks
- Operating temperature -40°C to +70°C
- Humidity 5% to 90%

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

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



There are no jumpers or physical settings required inside the device. Any configuration switching that needs to be done is handled by solid state switches. Configuration of all I/O is done through web pages found under the I/O Devices setup tabs illustrated above. Configuration of universal analog inputs is illustrated below.

Register	Type	Scaling	Slope	Intercept	Threshold (%) / Filter	Register Name
0001/1001	Analog	0-10V	1.000000	0.000000	0	My Sensor
0002/1003	Analog	10k III (F)	1.000000	0.000000	0	Office Temperature
0003/1005	Analog	0-10V	1.000000	0.000000	0	My Other Sensor
0004/1007	Analog	0-10V	1.000000	0.000000	0	
0005/1009	Discrete	Inverted	1.000000	0.000000	50	A Switch
0006/1011	Discrete	Inverted	1.000000	0.000000	50	Another Switch
0007/1013	Analog	0-10V	1.000000	0.000000	0	
0008/1015	Analog	0-10V	1.000000	0.000000	0	
0009/1017	Fast analog	10k III (F)	1.000000	0.000000	0	
0010/1019	Discrete	10k II (F)	1.000000	0.000000	0	
0011/1021	Dry contact	3k II (F)	1.000000	0.000000	0	
0012/1023	Analog	20k IV (F)	1.000000	0.000000	0	
0013/1025	Analog	10k III (C)	1.000000	0.000000	0	
0014/1027	Analog	10k II (C)	1.000000	0.000000	0	
0015/1029	Analog	3k II (C)	1.000000	0.000000	0	
0016/1031	Analog	20k IV (C)	1.000000	0.000000	0	
		Normal	1.000000	0.000000	0	
		Inverted	1.000000	0.000000	0	
		0-10V	1.000000	0.000000	0	

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.

The first 16 inputs of AddMe III are fully configurable as shown above. Inputs 17 and 18 are restricted to 0-10V only (or 4-20mA with dropping resistor).

Analog Outputs only require a slope and intercept setting. This allows the output to be scaled to engineering units.

Discrete Inputs may be set to function as (1) state input, (2) frequency input, or (3) total count input. When configured for frequency, the slope and intercept will be used to scale an input to engineering units. A common requirement for this is converting tachometer input to RPM. Total count mode creates a totalizing input, and the counts are maintained through power loss.

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.

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.

The screenshot shows a web interface with a navigation menu. The 'I/O Devices' tab is selected, and within it, the 'I/O Data' sub-tab is active. Below the navigation, there are buttons for 'Analog Inputs', 'Analog Outputs', 'Discrete Inputs', and 'Discrete Outputs'. A message states: 'This page displays the input levels on analog inputs.' Below this is a 'Show setup' link, a status indicator 'Showing 1 to 18 of 18', and an 'Update' button. A table displays the following data:

Register	Int Data	Float Data	Register Name
0001/1001	13302	5.014892	Sensor # 1
0002/1003	7490	74.90000	Office Temperature
0003/1005	13301	5.014514	Sensor #3
0004/1007	13299	5.013761	Sensor #4
0005/1009	1	10.00000	Switch # 1

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 32 of each data type are reserved for physical I/O on AddMe III. The number of reserved registers vary 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.

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 example, it means when relay #1 turns on, relays 2 and 3 will also come on automatically.

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 either to a data file or 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.

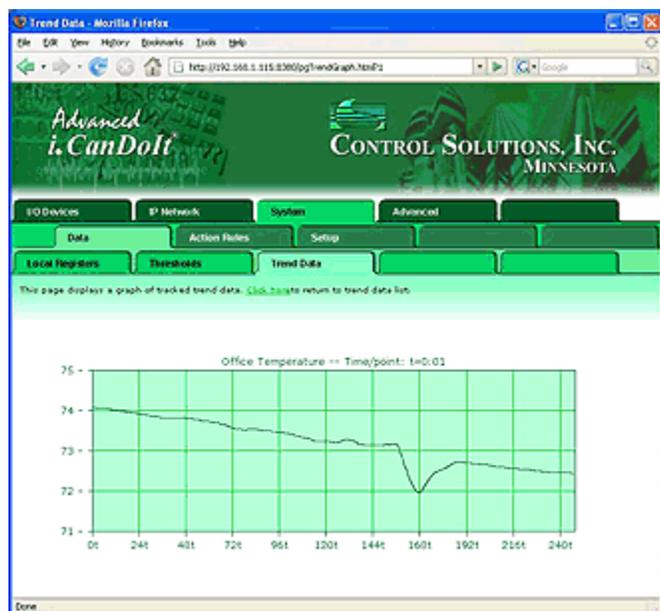
This page displays trend data that has been tracked for the registers indicated.

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. This virtual log file is maintained independent of any other log files you are maintaining via the Data Logging features. If you wish to send this data as a csv file attached to an email, you need to set up a true log file under the Data Logging tab.



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 simply be logged in an event file to simply record the fact that it happened sometime.

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.

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.

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:

How do I change event thresholds based on time of day?

Start by using a register for the test threshold rather than fixed value as shown below:

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:

Then alter the contents of this threshold register based on schedule in the Advanced->Scheduler->Weekly Schedule page as shown below.

#	S...M...T...W...T...F...S	Holidays	On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
1	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	Holidays	8:00:00	17:00:00	83	1000.00	1500.00	High Level Threshold

Using the example setup shown above, the event will be generated any time register 7 exceeds a level of 1000 between 8AM and 5PM, Monday through Friday. The rest of the time, register 7 needs to exceed a level of 1500 before an event will be generated.

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

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

Data may be logged to files in the internal memory based file system. These files may be retrieved via FTP, or delivered to you as an attachment to an email. The setup below illustrates sending a daily log file with an email, and also says that after 2 days, the old files should be discarded (to make way for new data).

This page allows you to manage data log files and logging schedules.

Data Log 1 File size: 1K Free space: 1406K/1906K Status: Error 0
File name: amlog200703241F0.txt Update < Prev Next >

File Name: FLASH0 Enable Logging Reset Error

Log Rate: 1:00:00 Log only when register # 0 is greater than zero.

New File: Daily End Time: 7:30:00 Size: 10 KB Life: 2 Days
Upon resources exhausted: Discard oldest Stop logging

Recipient: support@csimn.com
Subject: AddMeIII Data Log
Message: Daily log file from Jim's test system.

[Click here for log file format information](#)

Once a file is set up, you need to tell the system what data points to log. Any register in the system may be selected for logging, and multiple points may be logged to a single file. They are recorded in CSV format for easy import into spread sheet software. Up to three different log files may be active at the same time allowing different log rates for different points. Simply click the check box in the file column for the registers that are to be logged in that file. The Report box refers to reporting this point by HTTP Get to a remote server set up to receive data logging via Get requests.

Register #	Register Name	Log to File 1	Log to File 2	Log to File 3	Report
01001	Temp on AddMe III	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

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

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

The event log file is set up in the same manner as data log files, and it may also be delivered via email. The two main differences with the event log is the file format (still CSV but with different data), and the source of the data logged. Data logging allows you to check off a list of registers to log. Event logging allows you to check off any of the defined threshold rules for logging.

Rule #	Event Name	Log on True	Log on False	Report	
1	Low Temp Alarm	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input 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>.

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) 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 ***Time & Date Scheduling***

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

Weekly Scheduling using Time/Date or Astronomical Clocks

Suppose you want the lights in the parking lot to come on half hour before sunset, and go off half hour after sunrise. One way to do that is use a time clock and reset it a few times a year. The better way to do that with ***i.CanDoIt®*** is use the astronomical clock. Instead of on and off times, you simply say "dawn" and "dusk" with an optional offset.

#	S...M...T...W...T...F...S	Holidays	On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
1	<input checked="" type="checkbox"/>	Holidays	DUSK-30	DAWN+30	11	1.00000	0.00000	Relay 3
2	<input checked="" type="checkbox"/>	Holidays	21:00:00	7:30:00	12	1.00000	0.00000	Relay 4

In this example, we set register #11 half hour before sunset, and clear it half hour after sunrise. Our hardware configuration determined that register #11 is allocated to Relay 3, which we have wired to control the parking lot lights. Register #12, Relay 4, shows how to do it the old way using time of day.

The above example shows two scheduled events that will happen every day of the week because all days are checked. If you wish the weekly event to only occur on Saturday, for example, you would only check the "S" box at the end of the string of boxes. For weekdays only, you would check M through F (Monday through Friday).

The astronomical clock requires knowing where in the world you are. The clock setting page found at System->Setup->Time & Date includes a section for setting location.

Astronomical Clock

Latitude: Longitude: Time Zone:

Sunrise: **07:09** Sunset: **19:29**

Once the internal battery backed clock and calendar have been set, and you have entered latitude, longitude, and time zone, the clock setting page will show you when the system has calculated sunrise and sunset to be at this location.

On Demand Scheduling using Time/Date or Astronomical Clocks

Sometimes it is desirable to schedule something to happen just once at a specific time and date. This is done with "on demand" scheduling. You simply specify a starting time and date, and ending time and date, and the given register will be set accordingly during that time.

#	On Time	On Date	Off Time	Off Date	Register Number	"On" Value	"Off" Value	Register Name
1	8:00:00	03/24/2007	17:00:00	03/24/2007	25	1.00000	0.0	Parking Lot Gate
2	8:00:00	03/31/2007	17:00:00	03/31/2007	25	1.00000	0.0	Parking Lot Gate

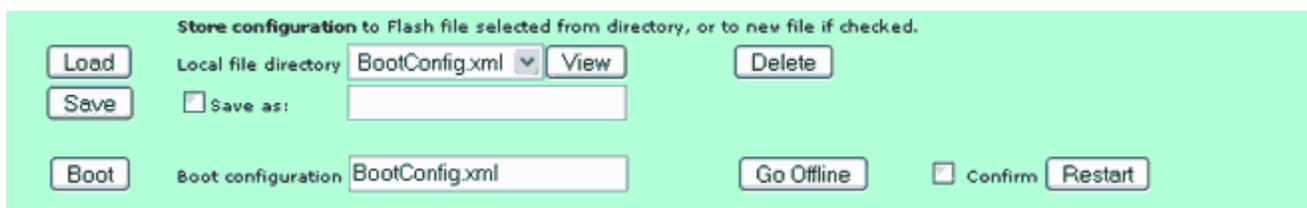
In this example, we are opening the parking lot gate on two consecutive dates. Our parking lot is normally closed on weekends, but we are sponsoring a special event requiring the use of our parking lot on two consecutive Saturdays. Fill in your own favorite example here.

i.CanDoIt® Guided Tour, part 12 ***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.



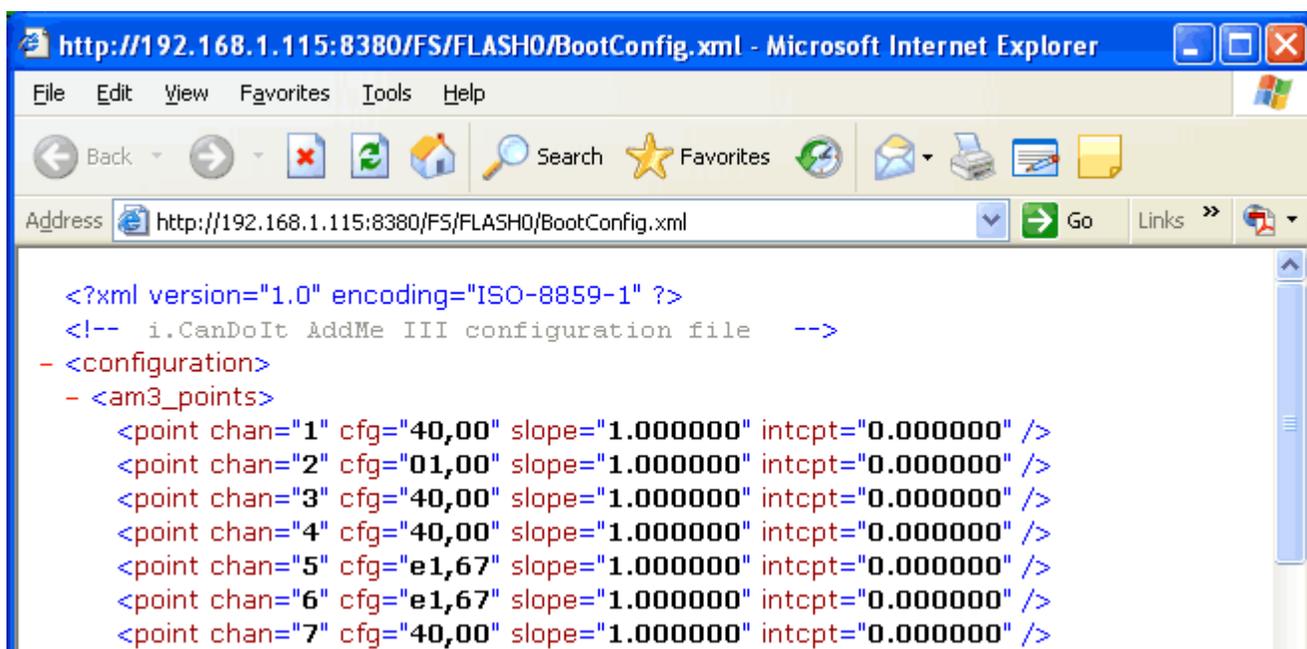
Store configuration to Flash file selected from directory, or to new file if checked.

Load Local file directory: View Delete

Save Save as:

Boot Boot configuration: Go Offline Confirm Restart

The first few lines of XML viewed in the browser look like this...



```
<?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 13 ***PL/i Programming***

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

The PL/i programming language is a descendant of PL/1 at the source code level. At the execution level, AddMe III runs on byte code similar to Java. The goal of PL/i was to create a compact compiler with a reasonably safe execution environment and a syntax with some of the power of C while being simpler than C.

The AddMe III architecture is register based. Therefore a carefully written program will be portable to systems running on Modbus, LonWorks or BACnet networks with equivalent results.

A summary of PL/i syntax and grammar can be found in the help pages in the device's internal web site under the Advanced->PL/i Programming tabs. [A more in depth tutorial is available here.](#)

What does PL/i look like?

Most of any given program will look a lot like Basic. Referring to subroutines as "procedures" is borrowed from older languages like Pascal. The block structure is common to many languages, and setting the blocks apart with "begin..end" instead of {} is also borrowed from Pascal. In fact, if you are a Pascal programmer, you may recognize several similarities. We hope you won't miss ":=".

The simple test program shown here will ramp an analog output on AddMe III up and down from zero to near full scale. It got named "dimmer" because if we connect an LED across the output, we have an LED dimmer. This makes a good "my first program" exercise.

```
program dimmer

declare
    myVar: int;

procedure turnon (setting: int)
declare
    level: int;
begin
    level = setting * 10;
    seti (19, level);
    delay (5);
end;

begin
    while TRUE do
        begin
```

```
    for myVar = 0 to 20 turnon(myVar);  
    for myVar = 20 down to 0 turnon(myVar);  
end;  
end
```

i.CanDoIt® Guided Tour, part 14 ***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.)

This page allows you to change this device's IP address, and optionally set other network parameters including DNS.

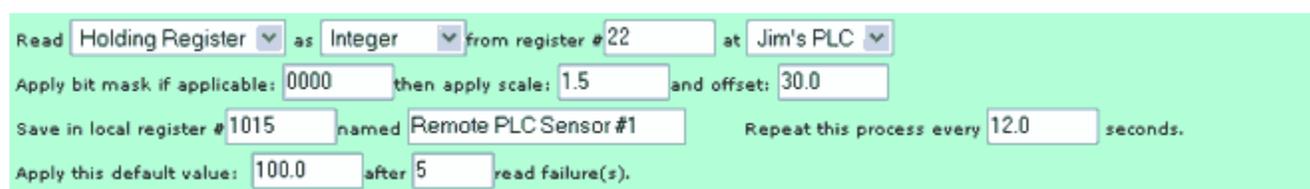
IP Address	<input type="text" value="192.168.1.115"/>	192.168.1.115	<input type="button" value="Review IP"/>	Email Server
Subnet Mask	<input type="text" value="255.255.255.0"/>	255.255.255.0	<input type="button" value="Change IP"/>	
Gateway	<input type="text" value="192.168.1.1"/>	192.168.1.1		
Static DNS1	<input type="text" value="68.87.77.130"/>	68.87.77.130	<input type="button" value="Apply DNS"/>	<input type="button" value="Reset DNS"/>
Static DNS2	<input type="text" value="68.87.72.130"/>	68.87.72.130		
Dynamic DNS Service	<input type="text" value="Custom DynDNS"/>	71.195.57.89	DDNS status: No update needed.	
Host Name	<input type="text" value="icandoit.deviceisp.net"/>			
DDNS User Name	<input type="text" value="icandoit"/>	Password	<input type="password" value="*****"/>	
HTTP Port	<input type="text" value="8380"/> (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="192.168.1.160"/>	<input checked="" type="checkbox"/> Enable	<input type="button" value="Update"/>	
Log Page	<input type="text" value="/icandoit/log"/>			
Log Parameters	<input type="text" value="reg100=999&reg101=888"/> (optional)			
Configuration Page	<input type="text" value="/config"/>			
Update Timeout	<input type="text" value="60"/>	Client ID	<input type="text" value="1992"/>	

i.CanDoIt® Guided Tour, part 15 ***Modbus TCP 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 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.



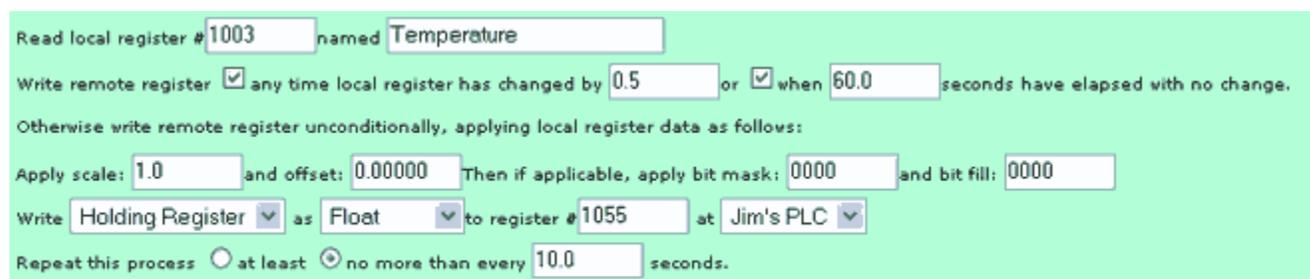
Read **Holding Register** as **Integer** from register # **22** at **Jim's PLC**

Apply bit mask if applicable: **0000** then apply scale: **1.5** and offset: **30.0**

Save in local register # **1015** named **Remote PLC Sensor #1** Repeat this process every **12.0** seconds.

Apply this default value: **100.0** after **5** 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 # **1003** named **Temperature**

Write remote register any time local register has changed by **0.5** or when **60.0** seconds have elapsed with no change.

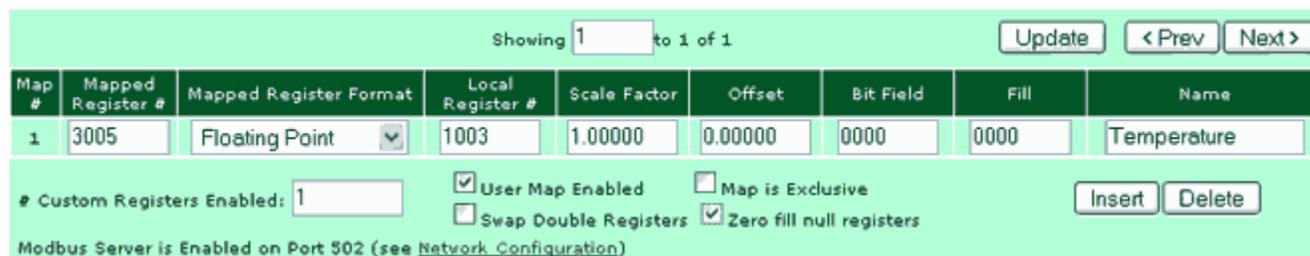
Otherwise write remote register unconditionally, applying local register data as follows:

Apply scale: **1.0** and offset: **0.00000** Then if applicable, apply bit mask: **0000** and bit fill: **0000**

Write **Holding Register** as **Float** to register # **1055** at **Jim's PLC**

Repeat this process at least no more than every **10.0** seconds.

The server map allows you to effectively relocate registers as seen by an outside Modbus TCP client (master). This allows the AddMe III 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 AddMe III locally.



Showing **1** to 1 of 1 Update < Prev Next >

Map #	Mapped Register #	Mapped Register Format	Local Register #	Scale Factor	Offset	Bit Field	Fill	Name
1	3005	Floating Point	1003	1.00000	0.00000	0000	0000	Temperature

Custom Registers Enabled: **1** User Map Enabled Map is Exclusive Insert Delete

Swap Double Registers Zero fill null registers

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

i.CanDoIt[®] Guided Tour, part 16 ***Modbus RTU Gateway***

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

The Modbus RTU gateway is a Modbus master, and can poll up to 50 slave devices. Its remote read and write capabilities are virtually identical to the TCP gateway.

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 **Holding Register** as **Integer** from register # **22** at **Jim's PLC**

Apply bit mask if applicable: **0000** then apply scale: **1.5** and offset: **30.0**

Save in local register # **1015** named **Remote PLC Sensor #1** Repeat this process every **12.0** seconds.

Apply this default value: **100.0** after **5** 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 # **1003** named **Temperature**

Write remote register any time local register has changed by **0.5** or when **60.0** seconds have elapsed with no change.

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

Apply scale: **1.0** and offset: **0.00000** Then if applicable, apply bit mask: **0000** and bit fill: **0000**

Write **Holding Register** as **Float** to register # **1055** at **Jim's PLC**

Repeat this process at least no more than every **10.0** seconds.

i.CanDoIt® Guided Tour, part 17 ***SNMP Agent***

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

Some versions of i.CanDoIt are 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

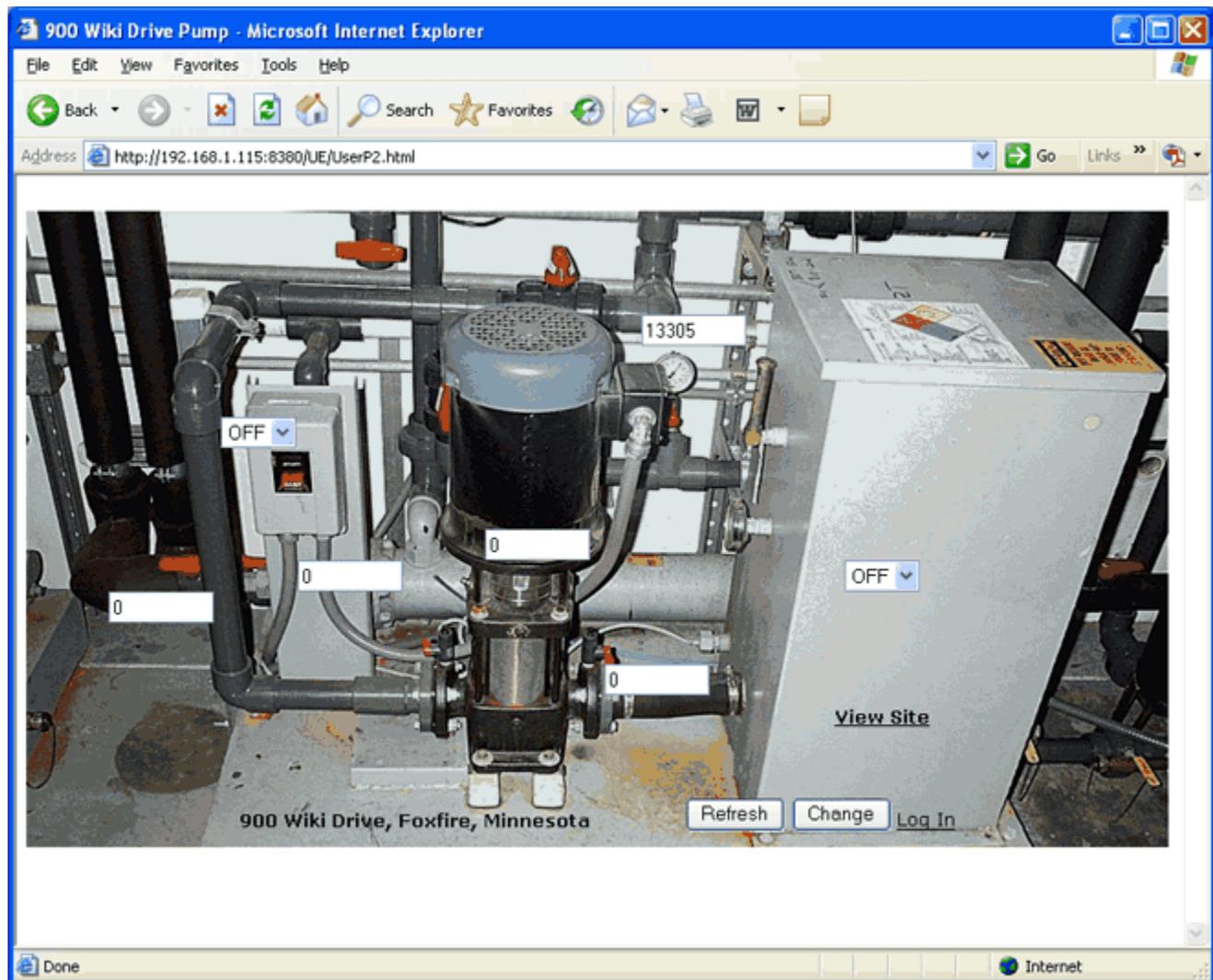
Once you have the Threshold Rule set up, you need to go to the IP Network->SNMP Setup->Trap Enable page and check off whether to trap on true, false, or both, and also which group(s) to send the trap to. The following is what you would see for the above rule:

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 18 ***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.