Control Solutions i.CanDoIt, i.Board, and Babel Buster SP
Web Server CGI Overview (2/2008)

User HTML

User HTML may be installed as a “wrapper” around the default web pages. To install user HTML, open an FTP session with the device, change directory to /FLASH0 (that’s flash zero), and send any combination of .html, .txt, .gif, and .jpg files. You can also include .pdf, .xml, and JavaScript .js files. PHP, ASP, etc., are not supported.

The top level user file must be named User.html (case sensitive). If this file is present in /FLASH0, it will be served instead of the default index.html page pre-programmed in the device. Once this page is open, it may link to any other html files in the /FLASH0 directory. All user HTML is filtered as it is served to provide dynamic content.

There are a handful of tricks that must be observed to make user html work. All references to other user pages and to user image files must have the file names preceeded with /UE/ as in /UE/User2.html or /UE/mypicture.gif. The UE stands for “user escape” and is treated as a virtual directory that actually points to /FLASH0.

Dynamic Data – Creating a Form

Dynamic access to register data is provided. Dynamic updates of register contents is also supported via the form post method. The form must be defined using the following tags:

```
<form id="UserForm" action="/UE/icanForm" method="post"
name="UserForm">
```

The submit button causing the post must be defined as:

```
<input type="submit" name="submitChange" value="Change">
```

Any other submit buttons will simply result in a page refresh. Only the submit button named “submitChange” with a value of “Change” will result in parsing of the form data. Only a form with action as named above will be parsed.

If you want to redefine the appearance of the button, you can implement a graphic button by including an image as follows, and then including the JavaScript function as shown:

```
<img src="specialbutton.gif" onclick="sendMeAway();" alt="" height="25"
width="133" border="0">

<script type="text/javascript"><!--
function sendMeAway() {
       document.UserForm.submitChange.value="Change";
      document.UserForm.submit();
}
//--></script>
```

Input Types: <input type=”text”>

Two types of data input are recognized by CGI processing of the user post: Text input and option select. The search string keyed upon for text is **<input type=”text”** and the search string keyed upon for the select option is **<select name=**

A text input should be constructed as follows:

```
<input type="text" name="reg22" value="%d" readonly size="8">
```

The contents of the register number included in the name (“regX”) will be displayed when the page is served, and the data will be taken dynamically from the register at that point in time, and again each time the page is refreshed. The data will be formatted using the C format string found in the value tag. Integer formats (%d, %04d, %x, etc) should be used for integer registers, and floating point (%f, %.2f, etc) should be used for floating point registers. If “readonly” is specified, data will only be displayed in this window. Otherwise the data returned by the post will be parsed, and the result placed back into the register.

The following keywords are recognized as text input “names”:

- regX – references local register number X
- namX – references name of register number X
- site – references the SNMP System Name

All of these data elements may be read, and will be written unless you specify “readonly”. The definition of read means take data from the local register when serving the page, and write means write data to the local register if the form was submitted by the appropriately named submit button (see Form above).

Important: If you change and write the system name, the page refresh will take up to one minute since this change is made to non-volatile Flash memory.

Input Types: <select>

An option select should be constructed as follows:

```
      <select name="reg25" size="1">
            <option selected value="0">OFF</option>
            <option value="1">ON</option>
      </select>
```

The strings corresponding to the values given will be displayed when the register named matches that value, otherwise “---“ will be displayed. When an option is selected and the form posted, the value corresponding to the new selection will be written back into the register. The “selected” tag shown above is not required since it is automatically inserted in the appropriate place (moved around) when the page is served.

Input Types: <input type=”hidden”>

An additional form of input has been added to filtered HTML. Hidden variables may be defined using the following syntax:

```
<input type="hidden" name="reg22" value="%d" readonly>
```

This will be processed the same as “text” input except the value is not displayed. This is useful as a means of providing non-displayed data to a JavaScript function. Hidden data upon return will be parsed and put back into registers unless readonly is specified. Omit readonly if hidden data should be parsed. This provides a means for JavaScript to get data back into registers.

Page Links

To create a link on the user page to get into the default preprogrammed pages, define a link to “../index.html”, for example:

```
<a href="../index.html">Log In</a>
```

To link to another user page in the FLASH0 directory, use a link such as:

```
<a href="/UE/pwUserP3.html">Room #1</a>
```

Links to graphic images you want shown on the page are created in similar fashion:

```
<img src="/UE/isotech.gif" alt="" height="448" width="804" border="0">
```

Note that you preface the page name with “/UE/” when the file is located in the FLASH0 directory, but preface the name with “../” when accessing a preprogrammed page.

Password Protection

There are 3 levels of password protection: Restricted, Maintenance, Administrator (root is a special form of administrator). User pages may be password protected at the “Restricted” level. To password protect a user page, simply insert the letters “pw” in front of the name. Therefore, if *User2.html* is a page you wish to protect, rename it *pwUser2.html*.

The top level page for User HTML <u>must</u> still be named *User.html* (and <u>not</u> *user.html* or not *USER.html*). If you don’t want anything useful to be completely unrestricted, simply put a plain dumb page in User.html with a link that says “log in” and link it to pwUser.html.

Note that “restricted” level of password protection means the user can access any “pw” user pages, but cannot access any pages beyond index in the pre-programmed page set.

Other Input Types

Radio buttons and other forms of input are not supported at this time. The HTML will be passed through, but not filtered and associated with register data. Therefore, you can use radio buttons, etc., with JavaScript, but you must explicitly associate the resulting data with hidden input variables in order to return the data to registers.

Naked Pages

You can embed any of the fixed pre-programmed pages into your User HTML using an iframe. However, once you give access to the page, the user is free to navigate through any of the tabs and links found on that page. A small subset of pages is provided in “naked” form meaning they have no color, no logo, and no links or tabs. These pages allow the user to do things (in user HTML pages) that are not accessible via data registers.

To embed a “naked page” in your user HTML page, use an inline frame as follows:

```
<p>
<iframe name="MyNetwork" src="../pgNkNetwork.html" frameborder="0"
height="500" width="820"></iframe>
</p>
```

Most naked pages do not require parameters; however, the trend graph page does require a parameter telling the server which trend graph you want. The following example shows an inline frame request for graph #1:

```
<p>
<iframe src="../pgNkTrendGraph.html?1" height="340"
width="758"></iframe>
</p>
```

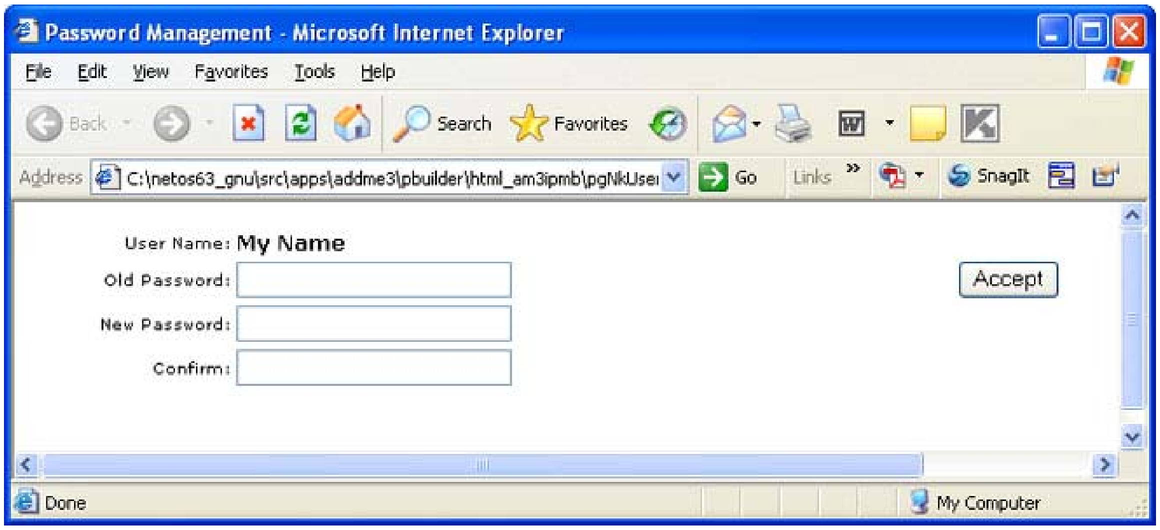
Be sure to create a frame big enough to hold the content of the embedded page. It should also be noted that the same password protection found in the internal pages is imposed on the naked pages exposed via the user pages.

The following pages are included as “naked pages”:

pgNkUser.html – allows user to change their password
pgNkNetwork.html – allows changing IP address, SNMP community & ports
pgNkNetwork2.html – same as pgNkNetwork with DDNS added
pgNkSNMPsetup.html – allows setting SNMP manager IP address, host name, etc.
pgNkSNMPident.html – allows setting system name, location, contact
pgNkTrendGraph.html?X – shows trend graph #X

These are illustrated below:

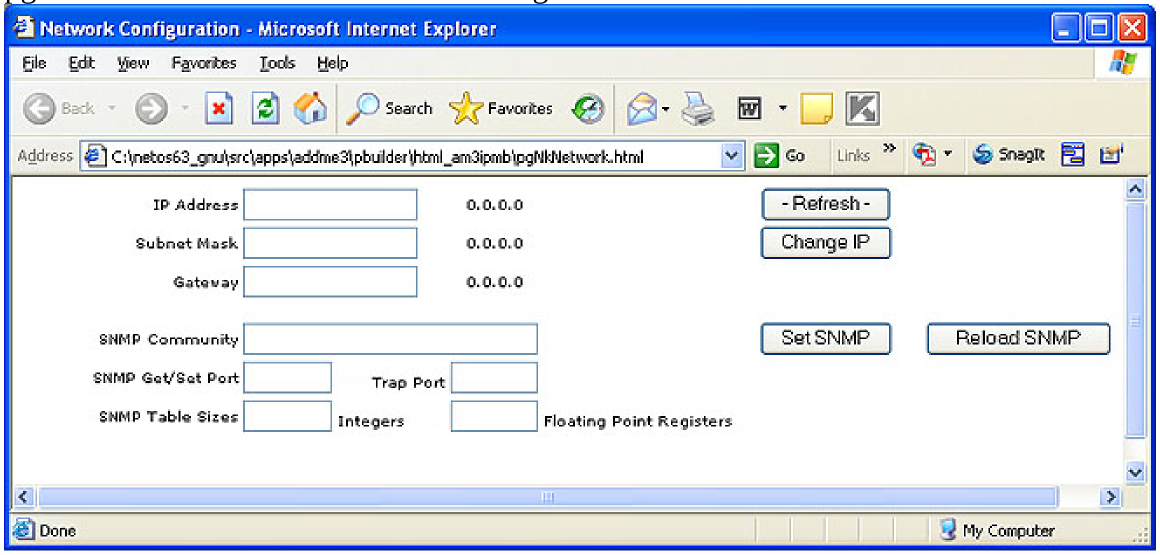
pgNkUser.html – allows user to change their password


Password Management - Microsoft Internet Explorer

Address: C:\netos63_gnu\src\apps\addme3\pbuilder\html_am3ipmb\pgNkUse

User Name:	My Name
Old Password:	
New Password:	
Confirm:	

pgNkNetwork.html – allows user to change IP address


Network Configuration - Microsoft Internet Explorer

Address: C:\netos63_gnu\src\apps\addme3\pbuilder\html_am3ipmb\pgNkNetwork.html

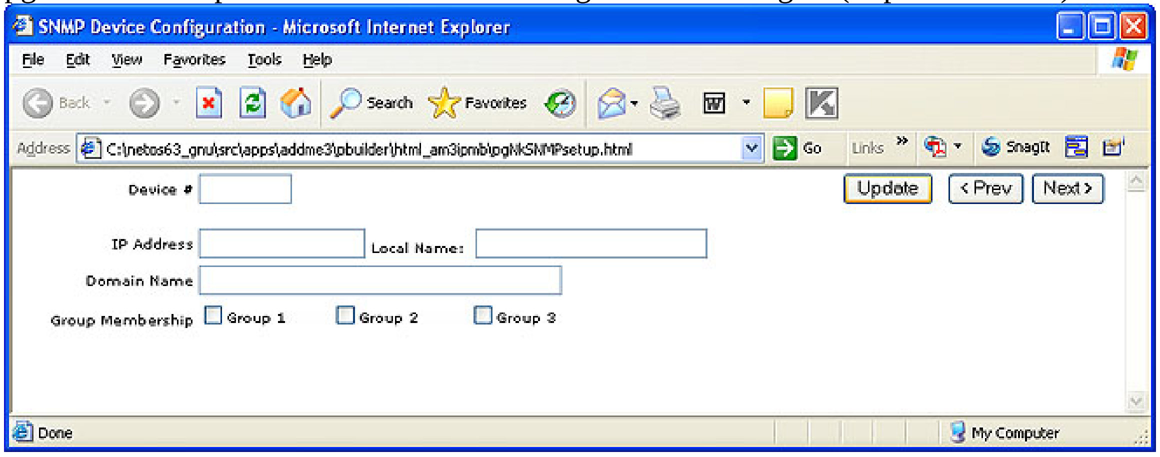
IP Address		0.0.0.0
Subnet Mask		0.0.0.0
Gateway		0.0.0.0

SNMP Community

SNMP Get/Set Port — Trap Port

SNMP Table Sizes — Integers — Floating Point Registers

pgNkSNMPsetup.html – allows user to change SNMP managers (trap destinations)


SNMP Device Configuration - Microsoft Internet Explorer

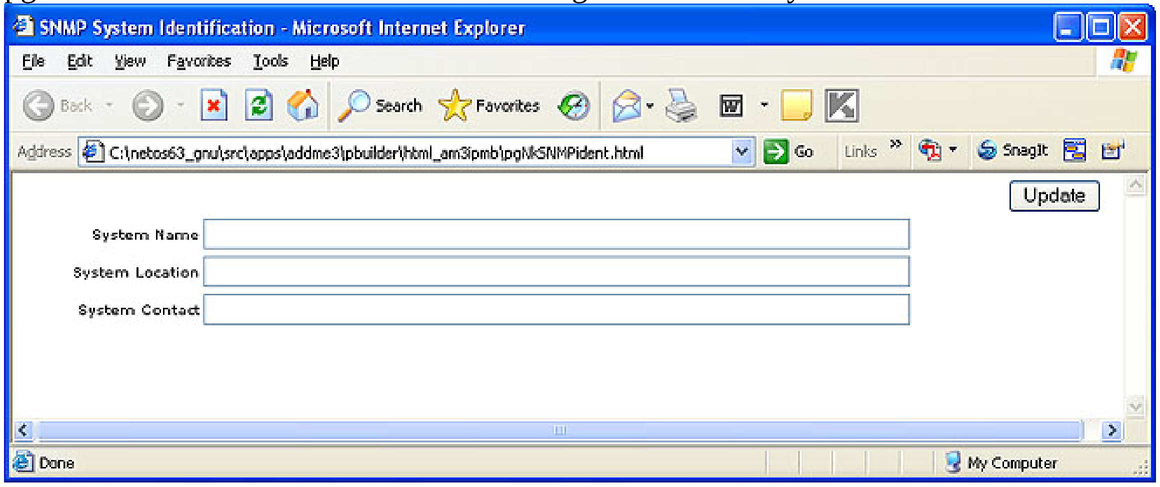
Address: C:\netos63_gnu\src\apps\addme3\pbuilder\html_am3ipmb\pgNkSNMPsetup.html

Device #

IP Address — Local Name:

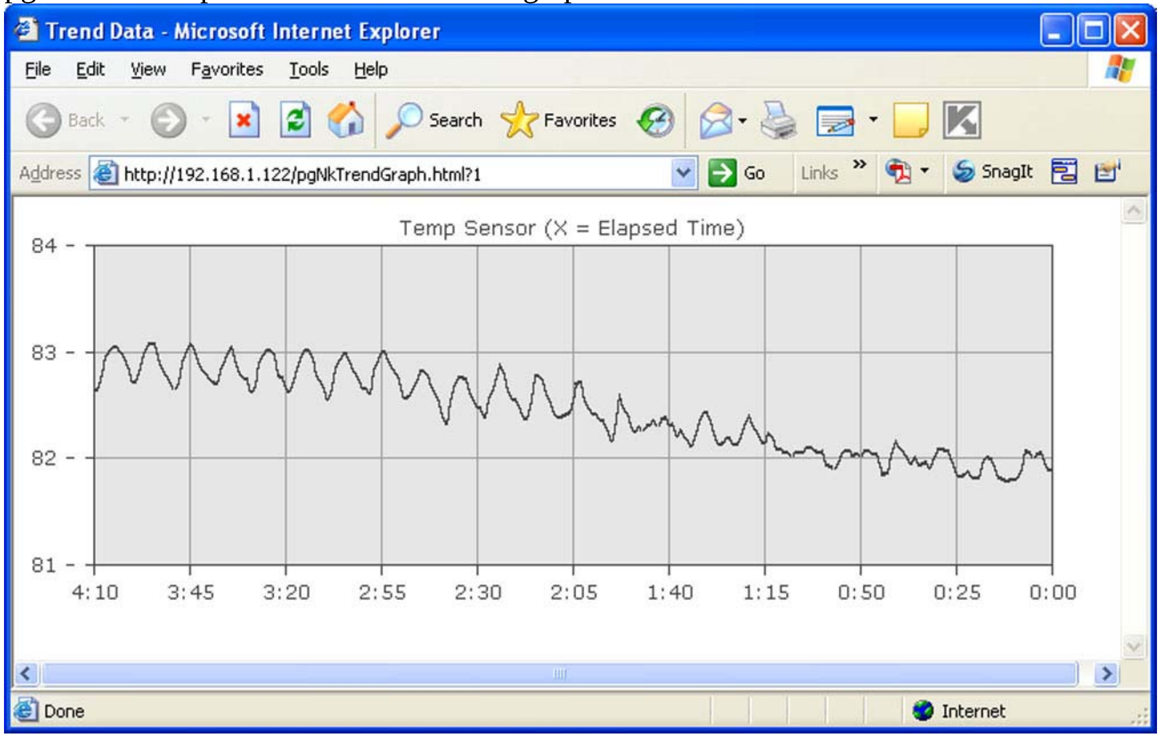
Domain Name

Group Membership ☐ Group 1 ☐ Group 2 ☐ Group 3 pgNkSNMPident.html – allows user to change SNMP identity


SNMP System Identification - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\netos63_gnu\src\apps\addme3\pbuilder\html_am3ipmb\pgNkSNMPident.html
Go Links SnagIt
Update
Done
My Computer

System Name
System Location
System Contact

pgNkTrendGraph.html?1 – shows trend graphs

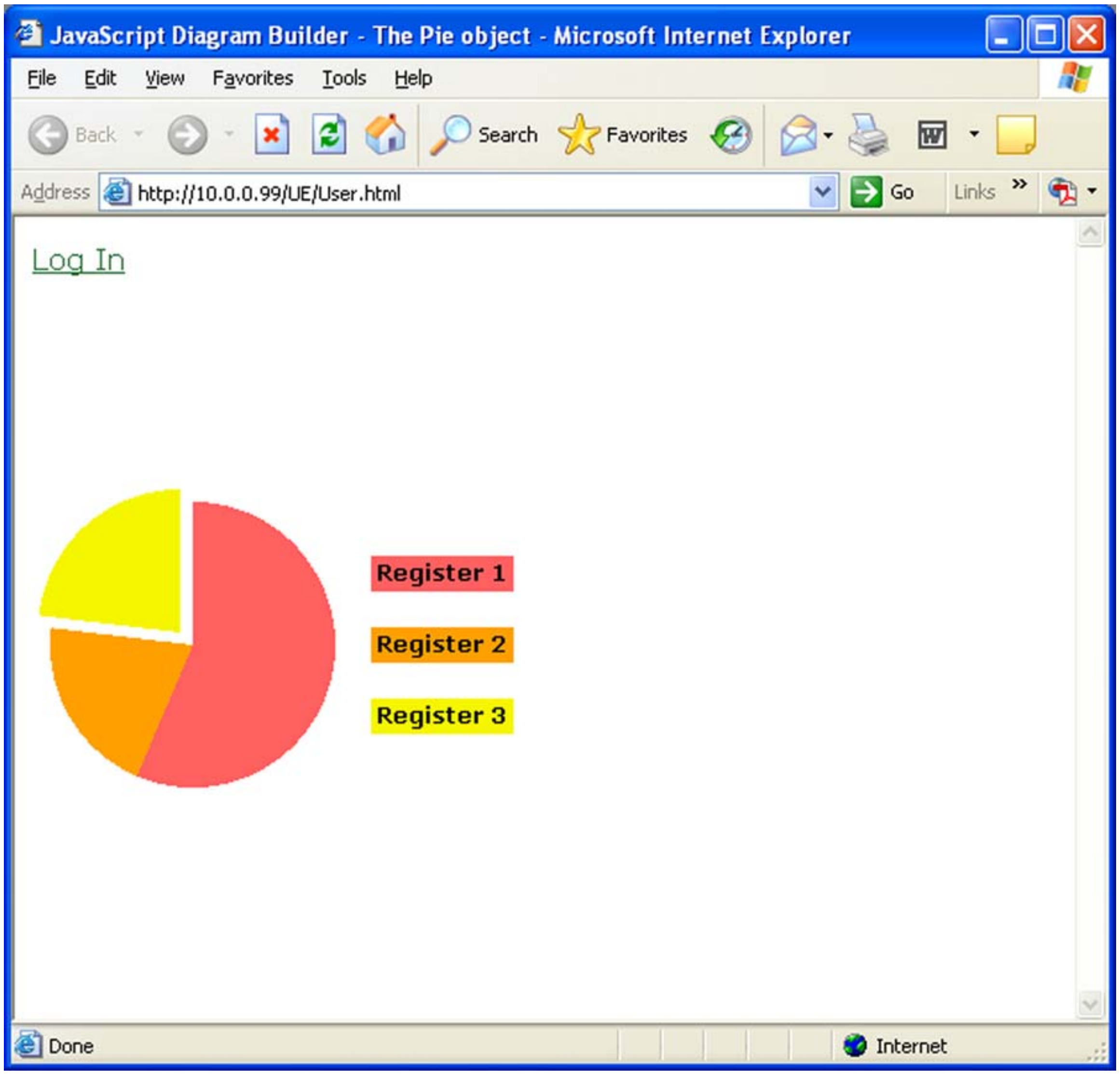

Trend Data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://192.168.1.122/pgNkTrendGraph.html?1
Go Links SnagIt
Temp Sensor (X = Elapsed Time)
84
83
82
81
4:10
3:45
3:20
2:55
2:30
2:05
1:40
1:15
0:50
0:25
0:00
Done
Internet

<u>JavaScript</u>

An external JavaScript library can be specified. The JS file should be loaded into the /FLASH0 directory along with User.html, etc. In the HTML file, the script file is referenced as (using diagram.js as an example):

```
<SCRIPT Language="JavaScript" src="/FS/FLASH0/diagram.js"></SCRIPT>
```

The following illustration is an example of a JavaScript library being used to generate a pie chart based on the contents of registers. This example even includes a mouse-over that pulls out a pie slice when the mouse is over its respective label. The proportional size of each slice will change based on the contents of registers 1 through 3 as viewed from the System->Data->Local Registers page.


JavaScript Diagram Builder - The Pie object - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://10.0.0.99/UE/User.html
Go
Links
Log In
Register 1
Register 2
Register 3
Done
Internet

The freeware JavaScript library diagram.js can be found by simply doing a Google search for “diagram.js”. Many types of graphics are supported by this library.

The library example was modified as shown in the User.html page whose source code is given here:

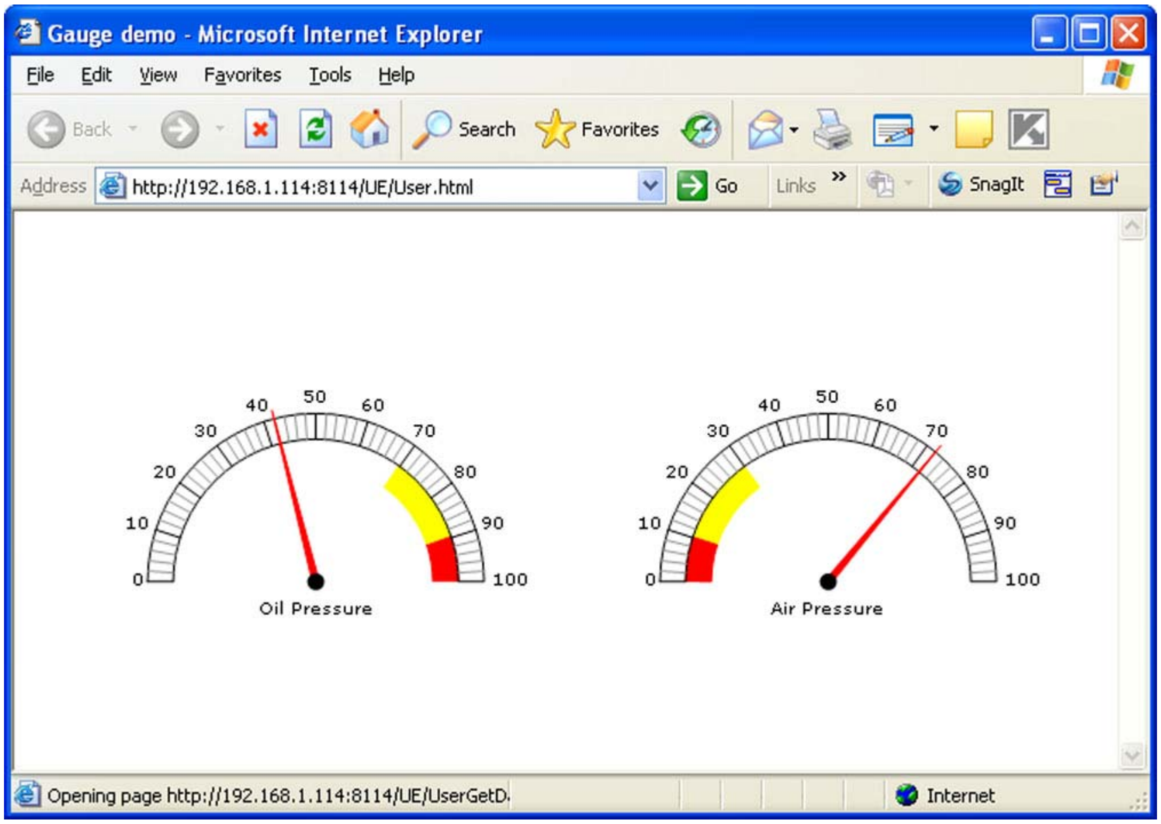
```
<html>
<head>
<title>JavaScript Diagram Builder - The Pie object</title>
<SCRIPT Language="JavaScript" src="/FS/FLASH0/diagram.js"></SCRIPT>
</head>
<body link="#226c30" vlink="#009900" alink="lime">
           <form method="post" name="myForm">
<input type="hidden" name="reg1" value="%d" readonly>
<input type="hidden" name="reg2" value="%d" readonly>
<input type="hidden" name="reg3" value="%d" readonly>
           </form>
           <script language="JavaScript"><!--
P=new Array();
document.open();
var ireg1 = parseInt(document.myForm.reg1.value);
var ireg2 = parseInt(document.myForm.reg2.value);
var ireg3 = parseInt(document.myForm.reg3.value);
var total = ireg1 + ireg2 + ireg3;
if (total == 0) {
     total = 3; ireg1 = 1; ireg2 = 1; ireg3 = 1;
     }
var pct1 = ireg1*100/total;
var pct2 = ireg2*100/total + pct1;
var pct3 = ireg3*100/total + pct2;
P[0]=new Pie(100,240,10,80,0*3.6,pct1*3.6,"#ff6060");
P[1]=new Pie(100,240,0,80,pct1*3.6,pct2*3.6,"#ffa000");
P[2]=new Pie(100,240,0,80,pct2*3.6,pct3*3.6,"#f6f600");
new Bar(200,190,280,210,"#ff6060","Register 1","#000000","","void(0)","MouseOver(0)","MouseOut(0)");
new Bar(200,230,280,250,"#ffa000","Register 2","#000000","","void(0)","MouseOver(1)","MouseOut(1)");
new Bar(200,270,280,290,"#f6f600","Register 3","#000000","","void(0)","MouseOver(2)","MouseOut(2)");
document.close();
function MouseOver(i) { P[i].MoveTo("","",10); }
function MouseOut(i) { P[i].MoveTo("","",0); }
-->
</script>
     <p><font face="Verdana, Arial, Helvetica, sans-serif"><a href="../index.html">Log In</a></font>
     </p>
     </body>
</html>
```

Advanced JavaScript using purchased library

You may purchase (for a rather reasonable price) a JavaScript library that supports more advanced graphical objects like live gauges as illustrated below. This is definitely not for the novice, but if you are up to the challenge, the combination of hidden variables, automatically refreshing iframes, and this library, can create a completely animated dash board.

You will find this library at *www.ejschart.com*. Their advanced analog gauge sample is a recreation of a complete analog automotive dash board.

The following example is an actual live page being served by an AddMe Jr. (Yes, the gauges move.)


Gauge demo - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Address
http://192.168.1.114:8114/UE/User.html
Go
Links
SnagIt
0
10
20
30
40
50
60
70
80
90
100
Oil Pressure
0
10
20
30
40
50
60
70
80
90
100
Air Pressure
Opening page http://192.168.1.114:8114/UE/UserGetD
Internet

<u>CGI Queries</u>

The CGI support included in i.CanDoIt allows direct query of registers using an HTTP Get which would be structured as follows if typed into the address window of a browser:

```
http://192.168.1.113/UE/QueryCGI?reg1001&reg1003&reg4
```

where 192.168.1.113 is the IP address of the device. The /UE/QueryCGI? may be followed by one or more "regN" instances separated by the ampersand. The N may be any valid local register number.

The "page" or reply that comes back in the above example would be displayed in the browser as:

```
reg1001=-1.000000&reg1003=0.000000&reg4=0
```

and its raw HTML is:

```
<html><head><title>i.CanDoIt Query</title></head><body>reg1001=-
1.000000&reg1003=0.000000&reg4=0</body></html>
```

Note: IE 6 incorrectly displays ® instead of &reg when the reply comes back. Firefox displays &reg correctly. The trademark symbol should have a semicolon as such: &reg**;** before being displayed as the trademark symbol. Since this form of query is primarily intended for M2M anyway, this is not much of a problem.

You can also set registers using a similar query. To set a register, the regN becomes regN=X where N is the register number, and X is any valid numeric string. For example, the following query:

```
http://192.168.1.113/UE/QueryCGI?reg1=99
```

returns the following:

```
<html><head><title>i.CanDoIt
Query</title></head><body>reg1=99</body></html>
```

<u>Support</u>

Please use the support ticket system at www.csimn.com for inquiries about User HTML. While we do not object to visiting on the phone, matters of this level of complexity are better documented via email with time allowed for our investigation of a reply.