ACCESS 9340 and 9360 Meters Ethernet Communications Card (9340-60-ETHER)

INTRODUCTION

This reference manual covers web page development for the ACCESS 9340 and 9360 Meters Ethernet Communications Card.

For additional information, refer to the following documentation:

- 9340-60-ETHER installation guide PMIM-ETHCC-0208
- 9340-60-ETHER user's guide PMCM-ETHCC-0208

APPLICATIONS FOR THE 9340-60-ETHER

The 9340-60-ETHER functions primarily as an Ethernet gateway to allow Ethernet access to Modbus/Jbus and SY/MAX serial devices. Additionally, the 9340-60-ETHER functions as a web and file server.

Supported Ethernet Protocols

The 9340-60-ETHER supports the following Ethernet protocols:

- **Modbus TCP/IP:** Used to serve compatible Modbus TCP/IP masters such as the WinPM.Net system via TCP port 502.
- **Hypertext Transfer Protocol (HTTP)**: Provides web server functionality via TCP port 80. Remote configuration and the viewing of real-time and historical data is possible using a web browser. Nonvolatile memory in the 9340-60-ETHERNET is used to store the web pages, graphics, documentation, controls, applets, and other files.
- **File Transfer Protocol (FTP)**: Provides the ability to transfer the following file types to and from the 9340-60-ETHERNET via TCP port 21:
  - GIF and JPEG graphics files
  - PDFs
  - Java applets
  - ActiveX controls
  - HTM/HTML
  - XML
  - XSL
  - XSD
  - DTD
  - Txt files
- **Simple Mail Transfer Protocol (SMTP)**: Provides the ability to send e-mail messages using TCP port 25.
- **Simple Network Management Protocol (SNMP):** Based on MIB2 format, SNMP provides the ability to store and send identifying and diagnostic information used for network management purposes via UDP port 161 .
- **Simple Network Time Protocol (SNTP)**: SNTP is a protocol used to synchronize the clocks of networked devices using a SNTP server via UDP port 123.

CREATING CUSTOM WEB PAGES FOR THE 9340-60-ETHER

To create custom web pages for the 9340-60-ETHER, you should have the following:

- A general understanding of the ACCESS Power Monitoring and Control System
- A general understanding of the Internet and the World Wide Web (WWW)
- Basic skills with text editor software
- Working knowledge of HyperText Markup Language (HTML) and JavaScripting.

Hardware, Software, and Logistics Requirements

Before proceeding, ensure that you have the following:

- The 9340-60-ETHER Communication Card installed and assigned an IP address
- Access to the 9340-60-ETHER via a LAN connection
- Web page editor or text editor software

File Storage Considerations

Custom HTML page examples are included on a CD-ROM shipped with the 9340-60-ETHER. The pages are configured to read data from the host device. If you need to delete a custom page from the 9340-60-ETHER, you can restore the deleted page from the CD-ROM.

Components of Custom HTML Pages

In general, each 9340-60-ETHER custom page has two components and each one is developed using different tools.

Static Components: These components include the page layout, static text, color schemes, lines, and tables. This part of the custom page is usually created with a web page editor and customized by adding or modifying HTML tags.

Because the static portion of the web page is dependant on the user, it is left to the web designer to decide how to write the HTML code. Therefore, it is not addressed in this manual.

Dynamic Components: These components include special delimiters known as server side includes (SSI) that tell the 9340-60-ETHER to dynamically get register information from attached devices and display it in the HTML page.

Accessing Devices

In general, the custom pages will be written in HTML with a special tag that tells the 9340-60-ETHER to dynamically get register information from a device. The delimiter at the beginning of a tag is (PL_ _) and the delimiter at the end is (_ _PL). This tells the 9340-60-ETHER to parse this string and dynamically fill it with register data. Table 1 lists the supported ACCESS tags.

Table 1: ACCESS Tags and Usage

Function Name	Function Code	PowerLogic TAG
SyMax Block Read - Registers	SyMax Function Code 0	<DeviceID>^<StartingRegisterAddress>[<NumberOfRegisters>] example tag = PL__1^1003[5]__PL example of data returned = 85,86,84,25,56
SyMax Scattered Read – Registers	SyMax Function Code 4	<DeviceID>^<RegisterAddress1>,<RegisterAddress2>,etc example tag = PL__1^1003,1004,1005,1006,1007__PL example of data returned = 85,86,84,25,56
Modbus Block Read- Coil Status	Modbus Function Code 1	<DeviceID>^C<StartingCoilAddress>[<NumberOfCoils>] example tag = PL__1^C1003[5]__PL example of data returned = 1,0,0,1,1

Table 1: ACCESS Tags and Usage

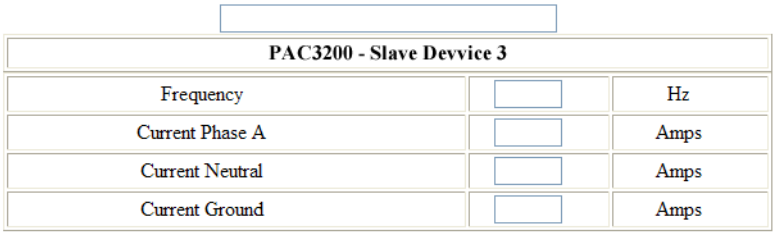
Function Name	Function Code	PowerLogic TAG
Modbus Block Read – Input Status	Modbus Function Code 2	<DeviceID>^D<StartingInputAddress>[<NumberOfInputs>] example tag = PL__1^D1003[5]__PL example of data returned = 1,0,0,1,1
Modbus Block Read – Holding Registers	Modbus Function Code 3	<DeviceID>^H<StartingRegisterAddress>[<NumberOfRegisters>] example tag = PL__1^H1003[5]__PL example of data returned = 85,86,84,25,56
Modbus Block Read – Input Registers	Modbus Function Code 4	<DeviceID>^I<StartingRegisterAddress>[<NumberOfRegisters>] example tag = PL__1^I1003[5]__PL example of data returned = 85,86,84,25,56
Modbus Scattered Read – Holding Registers	Modbus Function Code 100	<DeviceID>^S<RegisterAddress1>,<RegisterAddress2>,etc example tag = PL__1^S1003,1004,1005,1006,1007__PL example of data returned = 85,86,84,25,56
Modbus Block Read – General Reference	Modbus Function Code 20	<DeviceID>^F<StartingRegisterAddress>,[<NumberOfRegisters>]<File> example tag = PL__1^F1003[5]2__PL example of data returned = 85,86,84,25,56

NOTE: When creating custom HTML pages for the 9340-60-ETHER, an asterisk (*) can be used for the device ID. The asterisk acts as a wildcard so when it is parsed by the 9340-60-ETHER, the address of the host circuit monitor is automatically used for the device ID.

EXAMPLE 1—CREATING A NEW HTML CUSTOM PAGE

Example 1 is an exercise in creating the following new HTML page to be transferred to the 9340-60-ETHER. This web page is used to read data from an ACCESS 9360 meter.

Figure 1: Example 1 HTML Page



HTML Source Code for Example 1

The following is the HTML source code for the Example 1 web page. See Table 2 on page 7 for descriptions of the important elements in the source code.

NOTE: Bold type indicates the HTML and JavaScript that are key points to consider when making a custom page. Also note that the line numbers are for reference only; they are not part of the HTML.

Line No. HTML Syntax

```
1.  <html>
2.  <head>
3.  <META HTTP-EQUIV="refresh" CONTENT="5">
4.  <title>ACCESS 9360 - Slave Device 3</title>
5.  </head>
6.  <body>
7.  <form name="view_form">
8.  <p align="center">
```

```
<input type = "text" name = "time_spot" size = "40">
<table border="1" width="600">
<tr>
<td width="600"><p align="center"><font size="4"><b>
ACCESS 9360 - Slave Device 3</b></font></p>
</td>
</tr>
</table>
<table border="1" width="600">
<tr>
<td width="300">
<p align="center">Frequency</p>
</td>
<td align="center" width="90"><p align="center">
<input type="text" size="5" name="frequency"></p>
<td width="100">
<p align="center">Hz</p>
</td>
</tr>
<tr>
<td width="300">
<p align="center">Current Phase A</p>
</td>
<td align="center" width="90"><p align="center">
<input type="text" size="5" name="currentphasea"></
p>
<td width="100">
<p align="center">Amps</p>
</td>
</tr>
<tr>
<td width="300">
<p align="center">Current Neutral</p>
</td>
<td align="center" width="90"><p align="center">
<input type="text" size="5" name="currentneutral"></
p>
<td width="100">
<p align="center">Amps</p>
</td>
</tr>
<tr>
<td width="300">
<p align="center">Current Ground</p>
```

```
51. </td>
52. <td align="center" width="90"><p align="center">
53. <input type="text" size="5" name="currentground"></
    p>
54. <td width="100">
55. <p align="center">Amps</p>
56. </td>
57. </tr>
58. </table>
59. <br><HR SIZE="1" width="66%"><CENTER><font
    face="Times Roman" size="2">© 2008 Siemens. All
    rights reserved.</font></CENTER>
60. </form>
61. <script language="JavaScript">
62. function ShowFreq()
63. {
64. Registers =
    [PL__3^3209,3210,3211,1180,1100,1103,1104__PL];
65. ScaleFactorA = Registers[0];
66. ScaleFactorB = Registers[1];
67. ScaleFactorC = Registers[2];
68. Frequency = Registers[3];
69. CurrentPhaseA = Registers[4];
70. CurrentNeutral = Registers[5];
71. CurrentGround = Registers[6];
72. ScaleFactorAMultiplier = 0;
73. ScaleFactorBMultiplier = 0;
74. ScaleFactorCMultiplier = 0;
75. ScaleFactorFMultiplier = 0.01;
76. TheTime = new Date();
77. switch (ScaleFactorA)
78. {
79. case -2:
80. ScaleFactorAMultiplier = 0.01;
81. break;
82. case -1:
83. ScaleFactorAMultiplier = 0.1;
84. break;
85. case 1:
86. ScaleFactorAMultiplier = 10;
87. break;
88. default:
89. ScaleFactorAMultiplier = 1;
90. break;
```

```
91. }
92. switch (ScaleFactorB)
93. {
94. case -2:
95. ScaleFactorBMultiplier = 0.01;
96. break;
97. case -1:
98. ScaleFactorBMultiplier = 0.1;
99. break;
100.case 1:
101.ScaleFactorBMultiplier = 10;
102.break;
103.default:
104.ScaleFactorBMultiplier = 1;
105.break;
106.}
107.switch (ScaleFactorC)
108.{
109.case -2:
110.ScaleFactorCMultiplier = 0.01;
111.break;
112.case -1:
113.ScaleFactorCMultiplier = 0.1;
114.break;
115.case 1:
116.ScaleFactorCMultiplier = 10;
117.break;
118.default:
119.ScaleFactorCMultiplier = 1;
120.break;
121.}
122.Frequency *= ScaleFactorFMultiplier;
123.CurrentPhaseA *= ScaleFactorAMultiplier;
124.if (CurrentNeutral == -32768)
125.CurrentNeutral = "N/A";
126.else
127.CurrentNeutral *= ScaleFactorBMultiplier;
128.if (CurrentGround == -32768)
129.CurrentGround = "N/A";
130.else
131.CurrentGround *= ScaleFactorCMultiplier;
132.document.view_form.frequency.value = Frequency;
```

```
133. document.view_form.currentphasea.value =
     CurrentPhaseA;
134. document.view_form.currentneutral.value =
     CurrentNeutral;
135. document.view_form.currentground.value =
     CurrentGround;
136. document.view_form.time_spot.value = TheTime;
137. }
138. ShowFreq();
139. </script>
140. </body>
141. </html>
```

Table 2: Description of HTML Source Code for Example 1

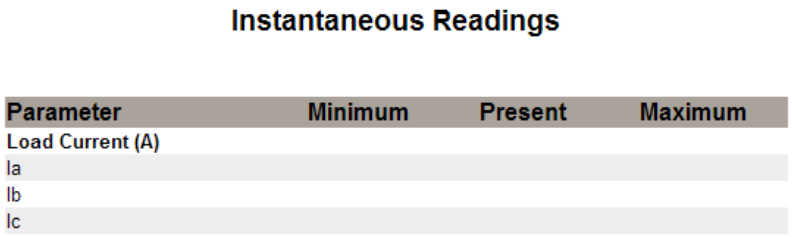
HTML Code Line No.	Description
HTML Source for the Static Elements	
3	HTML tag to set up page refresh cycle in seconds.
4	HTML tag to define page title label. This title appears on the browser title bar and is used in the main links page of the 9340-60-ETHER.
13	HTML syntax to write the title of the table "ACCESS 9360 - Slave Device 3".
20	HTML syntax to write "Frequency" table cell text label.
23	HTML syntax for the input control to be filled by dynamic data.
25	HTML syntax to write "Hz".
30, 33, 35	HTML syntax for displaying Current Phase A.
40, 43, 45	HTML syntax for displaying Current Neutral.
50, 53, 55	HTML syntax for displaying Current Ground.
JavaScripting Code for the Dynamic elements	
64	This line contains the following: "PL" delimiters at the beginning and end to signify to the 9340-60-ETHER to parse this string and dynamically fill it with register data. 3^ to signify the serial slave device address on the daisy chain. 3209,3210, ...,1104 a list of register numbers, which contain necessary ACCESS 9360 data.
65	Register #3209 of the ACCESS 9360 has the Scale Factor A value.
66	Register #3210 of the ACCESS 9360 has the Scale Factor B value.
67	Register #3211 of the ACCESS 9360 has the Scale Factor C value.
68	Register #1180 of the ACCESS 9360 has the Frequency value.
69	Register #1100 of the ACCESS 9360 has the Current Phase A value.
70	Register #1103 of the ACCESS 9360 has the CurrentNeutral value.
71	Register #1104 of the ACCESS 9360 has the Current Ground value.
132-135	JavaScript statements to print variable values into their field.

Once you have created the HTML page, you must transfer this page to the 9340-60-ETHER. For details, see the 9340-60-ETHER user's guide (PMCM-ETHCC-0208).

EXAMPLE 2—CREATING A NEW HTML CUSTOM PAGE

Example 2 is an exercise in creating the following new HTML page to be transferred to the 9340-60-ETHER. This web page is used to read data from a PAC3200.

Figure 2: Example 2 HTML Page



HTML Source Code for Example 2

The following is the HTML source code for the Example 2 web page. See Table 3 on page 12 for descriptions of the important elements in the source code.

NOTE: Bold type indicates the HTML and JavaScript that are key points to consider when making a custom page. Also note that the line numbers are for reference only; they are not part of the HTML.

Line No.	HTML Syntax
1.	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`
2.	`<html xmlns="http://www.w3.org/TR/1999/REC-html-in-xml">`
3.	`<head>`
4.	`<style type="text/css">`
5.	`.gray { font-family:'Arial'; font-size:10pt; color:#000000; background-color:#eeeeee;}`
6.	`.white { font-family:'Arial'; font-size:10pt; color:#000000; background-color:#ffffff;}`
7.	`.title { font-family:'Arial'; font-weight:bold; font-size:14pt; color:#000000; background-color:#ffffff;}`
8.	`.subtitle { font-family:'Arial'; font-weight:bold; color:#000000; font-size:12pt; background-color:#a9a39c;}`
9.	`.subtitlew { font-family:'Arial'; color:#000000; font-size:12pt; background-color:#ffffff;}`
10.	`</style>`
11.	`<title>PAC3200 Instantaneous Readings</title>`
12.	`<script type="text/javascript">`
13.	`if (window.XMLHttpRequest) {`
14.	`  // If IE7, Mozilla, Safari, and so on: Use native object`
15.	`  var xmlhttp= new XMLHttpRequest();`
16.	`}`
17.	`else`
18.	`{`
19.	`  if (window.ActiveXObject) {`
20.	`  // ...otherwise, use the ActiveX control for IE5.x and IE6`
21.	`    var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");`

```
22.   }
23. }
24. var sampleRate = 1000;
25. var postString = "R=PL_"+"_3" + "^14[6]"+"__PL" +
    "PL_"+"_*" + "^88[6]"+"__PL" +
    "PL_"+"_*" + "^158[6]"+"__PL";
26. var Title = "Instantaneous Readings";
27. var STitles = ["Parameter", "Minimum", "Present", "Maximum"];
28. var Labels = ["Load Current (A)", "Ia", "Ib", "Ic"];
29. function window_onload(){
30.   document.getElementById("Lmain").innerHTML  = Title;
31.   document.getElementById("Lsub1").innerHTML  = STitles[0];
32.   document.getElementById("Lsub2").innerHTML  = STitles[1];
33.   document.getElementById("Lsub3").innerHTML  = STitles[2];
34.   document.getElementById("Lsub4").innerHTML  = STitles[3];
35.   document.getElementById("LA").innerHTML     = Labels[0];
36.   document.getElementById("LIa").innerHTML    = Labels[1];
37.   document.getElementById("LIb").innerHTML    = Labels[2];
38.   document.getElementById("LIc").innerHTML    = Labels[3];
39.   startSampling();
40. }
41. function startSampling(){
42.   LoadData(postString);
43. }
44. function LoadData(){
45.   try{
46.     var temp;
47.     var Data = new Array();
48.     xmlhttp.open("POST", "Post__PL__Data", true);
49.     xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
50.     xmlhttp.onreadystatechange  = function() {
51.       if(xmlhttp.readyState == 4){
52.         try{
53.           temp=xmlhttp.responseText;
54.           Data=temp.split(",");
55.         }
56.         catch(exception){
57.           ProcessError(xmlhttp.responseText);
58.           return;
59.         }
60.         if(Data.length > 2){
61.           RefreshData(Data);
62.         }
63.         else{
```

```
64.           ProcessError(Data);
65.         }
66.         if(sampleRate != 0){
67.           TimerID = setTimeout("LoadData()", sampleRate)
68.         }
69.       }
70.     }
71.     xmlhttp.send(postString)
72.   }
73.   catch(exception){
74.     if(sampleRate != 0){
75.       setTimeout("LoadData()", sampleRate);
76.     }
77.   }
78. }
79. function ProcessError(Error){
80.   LoadData();
81. }
82. function ConvertIEE754(msw,lsw){
83.   // set denom=2^23
84.   denom=0x800000;
85.   mantissa=0;
86.   expon=0;
87.   // cycle through lsw
88.   for(i=16;i>0;i--){
89.     if(lsw&1)
90.       mantissa+=(1/denom);
91.     denom=denom>>1;
92.     lsw=lsw>>1;
93.   }
94.   // continue through lower 7 bits of msw
95.   for(i=7;i>0;i--){
96.     if(msw & 1)
97.       mantissa+=(1/denom);
98.     denom=denom>>1;
99.     msw=msw>>1;
100.  }
101.  // get expon
102.  expon=(0xff & msw)-127;
103.  // get sign
104.  if(0x100 & msw)
105.    sign=-1;
106.  else
```

```
107.    sign=1;
108.  // return m*2^e
109.  return(sign * (1+mantissa) * Math.pow(2,expon));
110.}
111.function RefreshData(vData){
112.  currentTime = new Date();
113.  document.getElementById("time").innerHTML = currentTime.toLocaleDateString() + " " +
    currentTime.toLocaleTimeString();
114.  document.getElementById("MinIa").innerHTML  = ConvertIEE754(vData[6],
    vData[7]).toFixed(2);
115.  document.getElementById("PresIa").innerHTML = ConvertIEE754(vData[0],
    vData[1]).toFixed(2);
116.  document.getElementById("MaxIa").innerHTML  = ConvertIEE754(vData[12],
    vData[13]).toFixed(2);
117.  document.getElementById("MinIb").innerHTML = ConvertIEE754(vData[8],
    vData[9]).toFixed(2);
118.  document.getElementById("PresIb").innerHTML  = ConvertIEE754(vData[2],
    vData[3]).toFixed(2);
119.  document.getElementById("MaxIb").innerHTML = ConvertIEE754(vData[14],
    vData[15]).toFixed(2);
120.  document.getElementById("MinIc").innerHTML = ConvertIEE754(vData[10],
    vData[11]).toFixed(2);
121.  document.getElementById("PresIc").innerHTML = ConvertIEE754(vData[4],
    vData[5]).toFixed(2);
122.  document.getElementById("MaxIc").innerHTML = ConvertIEE754(vData[16],
    vData[17]).toFixed(2);
123.}
124.</script>
125.</head>
126.<body onload="window_onload()" style="background-color: #FFFFFF">
127.<table style="margin-right: auto; margin-left: auto; width: 80%" border="0" cellspacing="0">
128.  <tr class="title">
129.    <td colspan="4" id="Lmain" style="text-align: center"></td>
130.  </tr>
131.  <tr class="subtitlew">
132.    <td colspan="4" style="text-align: center"> </td>
133.  </tr>
134.  <tr>
135.    <td> </td>
136.    <td colspan="3" id="time" style="text-align: right" class="white"></td>
137.  </tr>
138.  <tr class="subtitle">
139.    <td id="Lsub1" style="text-align: left"></td>
140.    <td id="Lsub2" style="text-align: center"></td>
141.    <td id="Lsub3" style="text-align: center"></td>
142.    <td id="Lsub4" style="text-align: center"></td>
```

```
143.  </tr>
144.  <tr class="white">
145.    <td style="font-weight: bold" id="LA"></td>
146.    <td colspan="3"> </td>
147.  </tr>
148.  <tr class="gray">
149.    <td id="LIa"></td>
150.    <td id="MinIa" style="text-align: center"></td>
151.    <td id="PresIa" style="text-align: center"></td>
152.    <td id="MaxIa" style="text-align: center"></td>
153.  </tr>
154.  <tr class="white">
155.    <td id="LIb"></td>
156.    <td id="MinIb" style="text-align: center"></td>
157.    <td id="PresIb" style="text-align: center"></td>
158.    <td id="MaxIb" style="text-align: center"></td>
159.  </tr>
160.  <tr class="gray">
161.    <td id="LIc"></td>
162.    <td id="MinIc" style="text-align: center"></td>
163.    <td id="PresIc" style="text-align: center"></td>
164.    <td id="MaxIc" style="text-align: center"></td>
165.  </tr>
166.</table>
167.</body>
168.</html>
```

Table 3: Description of HTML Source Code for Example 1

HTML Code Line No.	Description
HTML Source for the Static Elements	
4 - 10	HTML section that defines styles used to format elements in the page.
11	HTML tag to define the page title label. This title appears on the browser title bar and is used in the main links page of the 9340-60-ETHER.
12 - 124	HTML tag that contains the JavaScript code used to dynamically display data on the page.
126	The “onload” attribute in the <body> tag tells the browser to run the JavaScript function “window_onload()” when the page loads.
129	HTML section that contains the title for the table. The “id” attribute in the <td> tag is used by the JavaScript code to identify where to enter the title text.
139 - 142	HTML section that contains the subtitles for the table. The “id” attributes in the <td> tags are used by the JavaScript code to identify where to enter the subtitle text.
145	HTML section that contains the title for the load current data. The “id” attribute in the <td> tag is used by the JavaScript code to identify where to enter the title text.
149 - 152	HTML section that contains the data for minimum, present, and maximum current A.
155 - 158	HTML section that contains the data for minimum, present, and maximum current B.

Table 3: Description of HTML Source Code for Example 1

HTML Code Line No.	Description
161 - 164	HTML section that contains the data for minimum, present, and maximum current C.
JavaScripting Code for the Dynamic elements	
13 - 23	These lines determine whether to use the native XMLHttpRequest object or the ActiveX control based on the browser used. The object and the control allows the JavaScript to retrieve and update data on the page without refreshing the page.
24	This variable contains how often in milliseconds to retrieve and update the data on the page.
25	This line contains the following: *NOTE: Due to the way JavaScript processes this text string, you must type it exactly as shown in the example code on line 25. Change only the device number and/or the register numbers.* **PL** delimiters at the beginning and end to signify to the 9340-60-ETHER to parse this string and dynamically fill it with register data. **3" + "^** to signify the serial slave device address on the daisy chain, which is 3 in this example. Use an asterisk (*) to read values from the host device. **14[6]**, **88[6]**, and **158[6]** are register number blocks. For example, 14 is the starting register number and the number 6 in brackets indicates there are six registers in the block: 14, 15, 16, 17, 18, and 19. These six registers contain necessary PAC3200 data for I_a. In total, there are 18 registers to retrieve (6 + 6 + 6 = 18).
26	This variable contains the text for the title of the table.
27	This varialbe contains the text array for the subtitles in the table.
28	This variable contains the text array for the labels used to indicate the type of data displayed.
29 - 40	This section displays the title, subtitles, and labels defined on lines 26 - 28. The bold text in the sample JavaScript matches an “id” attribute in the HTML code. When the JavaScript code runs, it looks for the matching “id” in the HTML code and displays the appropriate text.
41 - 81	This section processes and loads the data retrieved from the device. You do not need to change anything in this section.
82 - 110	This section converts the data retrieved from the PAC3200 into data that can be displayed properly in a browser. You do not need to change anything in this section.
111 - 123	This section refreshes the data retrieved from a device, so it can be used by the functions that load the data. The text in the parenthesis for getElementById matches an “id” attribute in the HTML. When the JavaScript code runs, it looks for the matching “id” in the HTML code and displays the appropriate text. The numbers in the brackets for each vData variable represent where an item is located within the vData array. This array contains the data retrieved from the register blocks on line 25. Based on this line, the first six spots in the array, which are 0 to 5, are for present I_a, I_b, and I_c values. Because of the way the PAC3200 stores data, two register numbers are needed to generate a value to display in the browser. For I_a, vData[0] and vData[1] are used; for I_b, vData[2] and vData[3] are used; and for I_c, vData[4] and vData[5] are used.

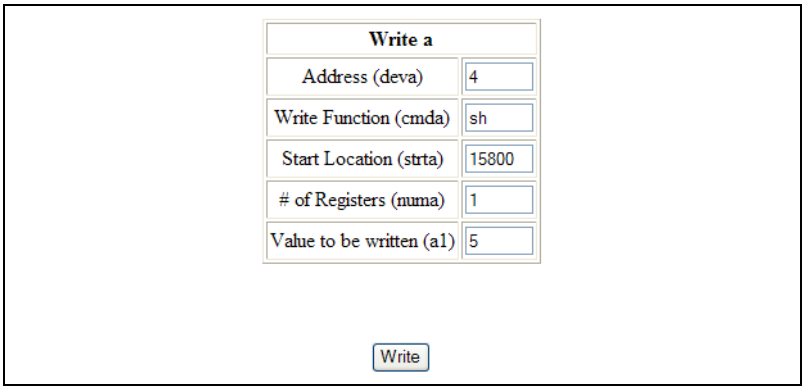
Once you have created the HTML page, you must transfer this page to the 9340-60-ETHER. For details, see the 9340-60-ETHER user’s guide (PMCM-ETHCC-0208).

CREATING A CUSTOM PAGE TO WRITE TO SLAVE DEVICES

The simplest way to create a custom HTML page that allows you to write to slave devices is to copy and modify the sample provided in “Sample Page for Slave Device Writes” on page 15.

We recommend that you set pages capable of doing writes to “Admin-level” password access. If you are using the write capability in a remote control application, be especially careful about assigning passwords and using this feature as writing to registers changes the configuration of a device and can energize and de-energize devices. See the 9340-60-ETHER user’s guide for more about password administration.

The sample results in the following basic page for the 9340-60-ETHER:



Required Format for Posting

Register writes are performed using the HTML “post” function. The form post name must equal “Post*PageName*” where *PageName* is the name of the custom page from which the form post is initiated.

A single custom page can contain multiple form posts as long as each form has a unique name attribute. Within each form post, there can be from one to five write functions with the following maximum number of register or coil data to be written.

Number of writes in individual form post	Number of max locations per write
1	95
2	45
3	25
4	20
5	15

Each write function must be represented by the following input variables:

	Write #1	Write #2	Write #3	Write #4	Write #5
dev*x*	“deva”	“devb”	“devc”	“devd”	“deve”
md*x*	“cmda”	“cmdb”	“cmdc”	“cmdd”	“cmde”
strt*x*	“strta”	“strtb”	“strtc”	“strtd”	“strte”
num*x*	“numa”	“numb”	“numc”	“numd”	“nume”
x#	“a1”	“b1”	“c1”	“d1”	“e1”
x#	“a2”	“b2”	“c2”	“d2”	“e2”
x#	“a3”	“b3”	“c3”	“d3”	“e3”
x#	…	…	…	…	…

Where:

- “dev*x*” is equal to the slave address of the device to be written.
- “cmd*x*” can equal any of the strings in the table below.

- "strt*x*" is equal to the start location (register or coil) to be written.
- "num*x*" is equal to the number of registers or coils to be written.
- "x#" is equal to the value or data to be written.

String	Description	Modbus Write Function
"sh"	Write Single Holding Register	0x06 Preset Single Register
"mh"	Write Multiple Holding Register(s)	0x10 Preset Multiple Registers
"sc"	Write Single Coil	0x05 Force Single Coil
"mc"	Write Multiple Coil(s)	0x0f Force Multiple Coils

Sample Page for Slave Device Writes

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<HTML><HEAD><TITLE>Example Write Register Page</TITLE>

<META http-equiv=Content-Type content="text/html; charset=windows-1252">

<META content="MSHTML 6.00.2800.1170" name=GENERATOR></HEAD>

<BODY><!--

This script creates a <FORM> tag using the URL string Copy and paste this script to the
beginning of your custom page.

-->

<SCRIPT>

/* get string from url box */

UrlString = document.URL;

/* -- create a string that contains the page name and the token (ECC/EGX400 only) -- */

/* get the length of the string */

Length = UrlString.length;

/* find the location where the page name begins */

index_string = UrlString.lastIndexOf("/", Length - 2);

/* remember the page name (including the token if it exists) */

end_string = UrlString.substring(index_string + 1,Length);

/* if this page is the result of a post the "Post" will be prepended to the beginning of the
string.  We do not want this, so delete it */

end_string = end_string.replace("Post","");

/* create the post string for the action attribute */

new_string = "Post" + end_string;

/* If there are multiple forms on a single page and the variable for the posts is accessed
outside of the forms,each form must have a unique name.  If the variables are not accessed
outside the form or if there is only oneform, this is not necessary. It is done here to show
how to use a variable to create a form name. For this pageit is hard-coded but it could be
generated at run time. */

form_name = "my_form";

/* create the string to create the form and send it */

post_string = "<FORM name='"+form_name+"' action='" + new_string + "' method='POST'
align='center'>";

document.write(post_string);
```

```
</SCRIPT>


<TABLE cellPadding=2 align=center border=0>
  <TBODY>
  <TR>
    <TD vAlign=top>
      <TABLE cellPadding=2 align=center border=1>
        <TBODY>
        <TR>
          <TD align=middle colSpan=2><B>Write a</B></TD></TR>
        <TR>
          <TD align=middle><FONT size=3>Address (deva)</FONT></TD>
          <TD align=middle><FONT size=3><INPUT size=5 value=4 name=deva>
            </FONT></TD></TR>
        <TR>
          <TD align=middle><FONT size=3>Write Function (cmda)</FONT></TD>
          <TD align=middle><FONT size=3><INPUT size=5 value=sh
            name=cmda><BR></FONT></TD></TR>
        <TR>
          <TD align=middle><FONT size=3>Start Location (strta)</FONT></TD>
          <TD align=middle><FONT size=3><INPUT size=5 value=15800 name=strta>
            </FONT></TD></TR>
        <TR>
          <TD align=middle><FONT size=3># of Registers (numa)</FONT></TD>
          <TD align=middle><FONT size=3><INPUT size=5 value=1 name=numa>
            </FONT></TD></TR>
        <TR>
          <TD align=middle><FONT size=3>Value to be written (a1)</FONT></TD>
          <TD align=middle><FONT size=3><INPUT size=5 value=5 name=a1>
          </FONT></TD></TR><!-- For four registers, change numa value to 4, and remove the
comment tags below --><!--
        <TR>
             <TD align=middle><FONT size=3>Value to be written (a2)</FONT></TD>
             <TD align=middle><FONT size=3>
             <input type=text name=a2 value=10 size=5>
             </FONT></TD>
        </TR>
        <TR>
             <TD align=middle><FONT size=3>Value to be written (a3)</FONT></TD>
```

```
            <TD align=middle><FONT size=3>
            <input type=text name=a3 value=100 size=5>
            </FONT></TD>
        </TR>
        <TR>
            <TD align=middle><FONT size=3>Value to be written (a4)</FONT></TD>
            <TD align=middle><FONT size=3>
            <input type=text name=a4 value=0 size=5>
            </FONT></TD>
        </TR>
--></TBODY></TABLE></TD></TR></TBODY></TABLE><BR><BR><BR>
<CENTER><INPUT type=submit value=Write></CENTER></FORM>

</BODY></HTML>
```

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Siemens for any consequences arising out of the use of this material.

SIEMENS