63230-319-230A1
03/2011

PowerLogic™ Ethernet Gateway EGX300
Reference Guide

Table of Contents

Schneider Electric

Introduction

The EGX300 functions primarily as an Ethernet gateway to allow Ethernet access to Modbus/Jbus and PowerLogic (SY/MAX) serial devices. Additionally, the EGX300 functions as a web and file server.

This reference manual covers custom web page development for the PowerLogic™ Ethernet Gateway EGX300.

For additional information, refer to the following documentation:

- EGX300 Installation Guide 63230-319-212A1
- EGX300 User Guide 63230-319-216A1

To download the documentation, go to www.powerlogic.com, select your country > Library, then navigate to the EGX300 and select the manual you want to download. Follow the instructions on the website for obtaining a password to access online literature.

Prerequisites

To create custom web pages for the EGX300, the following prerequisites are necessary:

- A working knowledge of Modbus, JBUS, and/or SY/MAX
- A general understanding of the Internet and the World Wide Web (WWW)
- A working knowledge of HyperText Markup Language (HTML) and JavaScript.

Components of Custom HTML Pages

Each EGX300 custom page has two components, standard HTML and JavaScript, and custom PowerLogic Tags.

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

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

Custom Components: Special delimiters known as PowerLogic Tags that tell the EGX300 how to dynamically get register information from its devices.

PowerLogic Tags

The EGX300 creates a request from its devices using information contained in a PowerLogic Tag. These tags indicate what type of request is desired (i.e. Read Holding Register), the device ID, and data supporting the request. The delimiter at the beginning of a tag is (PL_ _) and the delimiter at the end is (_ _PL). This tells the EGX300 to parse this string and dynamically fill it with register data. Table 1 lists the supported PowerLogic tags.

Table 1: PowerLogic 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
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 Write – Force Single Coil	Modbus Function Code 5	<DeviceID>^**X**<CoilAddress>[<data1>] example tag = PL__1^X8000[1]__PL example of data returned = pass/fail
Modbus Write – Preset Single Register	Modbus Function Code 6	<DeviceID>^**W**<RegisterAddress>[<data1>] example tag = PL__1^X8000[4110]__PL example of data returned = pass/fail
Modbus Write – Force Multiple Coils	Modbus Function Code 15	<DeviceID>^**Z**<StartCoilAddress>[<data1>,<data2>,<data3>.etc]] example tag = PL__1^X8000[1,1,1,1,1]__PL example of data returned = pass/fail
Modbus Write – Preset Multiple Registers	Modbus Function Code 16	<DeviceID>^**Y**<RegisterAddress>[<data1>,<data2>,<data3>.etc]] example tag = PL__1^X8000[46728,12,5121,1,13107,13107,59]__PL example of data returned = pass/fail
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
Modbus Write – General Reference	Modbus Function Code 21	<DeviceID>^**V**<RegisterAddress>[<data1>,<data2>,<data3>.etc]] <File> example tag = PL__1^X8000[46728,12,5121,1,13107,13107,59]2__PL example of data returned = pass/fail
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

Table 2: PowerLogic Tag Opcode Identifier

Function Name	Identifier
Modbus Block Read – Coil Status	C
Modbus Block Read – Input Status	D
Modbus Block Read – Holding Registers	H
Modbus Block Read – Input Registers	I
Modbus Scattered Read – Holding Registers	S
Modbus Block Read – General Reference	F
Modbus Write – Preset Single Register	W
Modbus Write – Force Single Coil	X
Modbus Write– Preset Multiple Registers	Y
Modbus Write – Force Multiple Coils	Z
Modbus Write – General Reference	V

Accessing Devices

There are two methods for reading data from the EGX300 devices described in the following sections:

Static PowerLogic Tags: Every time the data is refreshed the entire web page must be refreshed and parsed by the EGX300. The web page must have a .shtml extension.

Dynamic XMLHTTPRequest: The data is refreshed without refreshing the entire web page.

Static PowerLogic Tags

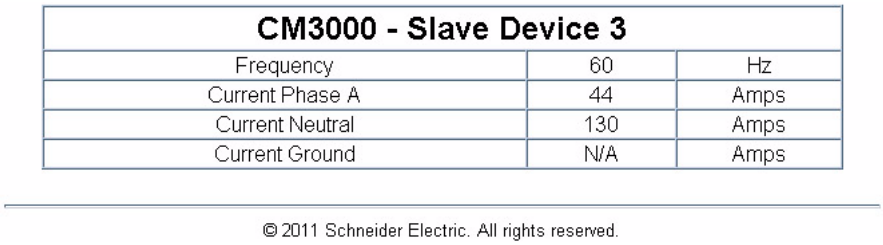
The custom pages are written in HTML and include static PowerLogic Tags telling the EGX300 to get register information from a device. The custom pages must be saved with the file extension .shtml. When the EGX300 receives a request from a browser for a page with a .shtml extension, it scans the file for PowerLogic Tags. When a PowerLogic Tag is found, the EGX300 creates the request for data, waits for a response from the slave device, then replaces the PowerLogic Tag with the data received. The EGX300 continues to scan the rest of the file, repeating the procedure whenever a PowerLogic Tag is found in the file. Once the entire file is scanned, the EGX300 serves the modified file to the web browser.

NOTE: If there are multiple tags, file scanning can be delayed due to slow or non-responding devices. To avoid delays, minimize the number of tags and consider the limitations of the Modbus protocol. For example, the maximum number of registers that can be read from a device using the Modbus function “Read Multiple Registers” is 125.

Creating a New Custom HTML Page

Use the following “PowerLogic Tags Custom HTML Page Source Code” or “XMLHTTPRequest Custom HTML Page Source Code” to create the custom HTML page in Figure 1.

Figure 1: Custom HTML Page



CM3000 - Slave Device 3		
Frequency	60	Hz
Current Phase A	44	Amps
Current Neutral	130	Amps
Current Ground	N/A	Amps

PowerLogic Tags Custom HTML Page Source Code

The HTML and JavaScript in bold type are key points to consider when creating a custom HTML page. See Table 3 on page 8 for source code descriptions.

NOTE: Line numbers are included for reference only and are not part of the HTML.

Line No. HTML Syntax

```
1.  <html>
2.  <head>
3.  <meta http-equiv="refresh" content="5">
4.  <title>CM3000 - Slave Device 3</title>
5.  </head>
6.  <body style="font-family:Arial" onLoad="updateData();">
7.  <form name="view_form">
8.  <div id="time_spot" style="text-align: center"></div>
9.  <table border="1" cellspacing="0" style="width: 600px; margin-
    left:auto; margin-right:auto">
10. <tr>
11. <td colspan="3" style="width: 600px; text-align: center; font-
    weight: bold; font-size: x-large">CM3000 - Slave Device 3</td>
12. </tr>
13. <tr>
14. <td style="width: 300px; text-align: center">Frequency</td>
15. <td style="width: 90px; text-align: center"
    id="frequency"></td>
16. <td style="width: 100px; text-align: center">Hz</td>
17. </tr>
18. <tr>
19. <td style="width: 300px; text-align: center">Current Phase
    A</td>
20. <td style="width: 90px; text-align: center"
    id="currentphasea"></td>
```

```
21. <td style="width: 100px; text-align: center">Amps</td>
22. </tr>
23. <tr>
24. <td style="width: 300px; text-align: center">Current
    Neutral</td>
25. <td style="width: 90px; text-align: center"
    id="currentneutral"></td>
26. <td style="width: 100px; text-align: center">Amps</td>
27. </tr>
28. <tr>
29. <td style="width: 300px; text-align: center">Current
    Ground</td>
30. <td style="width: 90px; text-align: center"
    id="currentground"></td>
31. <td style="width: 100px; text-align: center">Amps</td>
32. </tr>
33. </table>
34. <br /><hr style="width:66%; height:1px" /><div style="text-
    align: center; font-size: small">&copy; 2011 Schneider
    Electric. All rights reserved.</div>
35. </form>
36. <script type="text/javascript">
37. function updateData()
38. {
39. // Read Registers (this is actually done when the page is
    served by the EGX300)
40. Registers =
    [PL__3^3209,3210,3211,3208,1180,1100,1103,1104__PL];
41. // Assign Scale Factors
42. ScaleFactorA = Registers[0];  // Current Scale Factor
43. ScaleFactorB = Registers[1];  // Neutral Current Scale Factor
44. ScaleFactorC = Registers[2];  // Ground Current Scale Factor
45. // Assign Nominal Frequency
46. NominalFrequency = Registers[3];
47. // Assign Data Values
48. Frequency      = Registers[4];
49. CurrentPhaseA  = Registers[5];
50. CurrentNeutral = Registers[6];
51. CurrentGround  = Registers[7];
52. // Get the current date and time
53. TheTime = new Date();
54. // Scale Frequency
55. if(NominalFrequency != 400)
56. {
```

```
57. Frequency *= 0.01;
58. }
59. else
60. {
61. Frequency *= 0.10;
62. }
63. // Scale Phase A Current
64. CurrentPhaseA *= Math.pow(10, ScaleFactorA);
65. // Scale Neutral Current
66. if (CurrentNeutral == 32768)
67. {
68. CurrentNeutral = "N/A";
69. }
70. else
71. {
72. CurrentNeutral *= Math.pow(10, ScaleFactorB);
73. }
74. // Scale Ground Current
75. if (CurrentGround == 32768)
76. {
77. CurrentGround = "N/A";
78. }
79. else
80. {
81. CurrentGround *= Math.pow(10, ScaleFactorC);
82. }
83. // Put data on the page
84. document.getElementById("frequency").innerHTML = Frequency;
85. document.getElementById("currentphasea").innerHTML =
    CurrentPhaseA;
86. document.getElementById("currentneutral").innerHTML =
    CurrentNeutral;
87. document.getElementById("currentground").innerHTML =
    CurrentGround;
88. document.getElementById("time_spot").innerHTML = TheTime;
89. }
90. </script>
91. </body>
92. </html>
```

Table 3: Description of PowerLogic Tags HTML Source Code

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 EGX300.
11	HTML syntax to write the title of the table "CM3000 - Slave Device 3".
14	HTML syntax to write "Frequency" table cell text label.
15	HTML syntax for the input control to be filled by dynamic data.
16	HTML syntax to write "Hz".
19, 20, 21	HTML syntax for displaying Current Phase A.
24, 25, 26	HTML syntax for displaying Current Neutral.
29, 30, 31	HTML syntax for displaying Current Ground.
JavaScripting Code for the Dynamic elements	
40	This line contains the following: "PL" delimiters at the beginning and end to signify to the EGX300 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 CM3000 data.
42	Register #3209 of the CM3000 has the Scale Factor A value.
43	Register #3210 of the CM3000 has the Scale Factor B value.
44	Register #3211 of the CM3000 has the Scale Factor C value.
46	Register #3208 of the CM3000 has the Nominal Frequency value.
48	Register #1180 of the CM3000 has the Frequency value.
49	Register #1100 of the CM3000 has the Current Phase A value.
50	Register #1103 of the CM3000 has the CurrentNeutral value.
51	Register #1104 of the CM3000 has the Current Ground value.
84-87	JavaScript statements to print variable values into their field.

Once you have created the HTML page, you must transfer this page to the EGX300. For details, see the EGX300 User Guide (63230-319-216A1).

Dynamic XMLHTTPRequest

The custom pages are written in HTML and use the DOM XMLHTTPRequest to dynamically get register information from a device. The custom pages should be saved with the file extension .html, or .htm. When the EGX300 receives a request from a browser for a page with a .html or .htm extension, it serves the file to the web browser. The web page dynamically creates PowerLogic Tags and sends them to the EGX300, via XMLHTTPRequest. The EGX300 creates a request for data, waits for a response from the slave device, then returns the data to the web page in response to the XMLHTTPRequest.

The readyState property holds the status of the server's response. Each time the readyState property changes, the onreadystatechange function will be executed. The readyState property uses the following values:

0	The request is not initialized
1	The request has been set up
2	The request has been sent
3	The request is in process
4	The request is complete

XMLHTTPRequest Custom HTML Page Source Code

The HTML and JavaScript in bold type are key points to consider when creating a custom HTML page. See Table 4 on page 13 for source code descriptions.

NOTE: Line numbers are included for reference only and are not part of the HTML.

Line No. HTML Syntax

```
1.  <html>
2.  <head>
3.  <title>CM3000 - Slave Device 3 - XMLHTTP</title>
4.  </head>
5.  <body style="font-family:Arial" onload="getData();">
6.  <form name="view_form">
7.  <div id="time_spot" style="text-align: center"></div>
8.  <table border="1" cellspacing="0" style="width: 600px; margin-
    left:auto; margin-right:auto">
9.  <tr>
10. <td colspan="3" style="width: 600px; text-align: center; font-
    weight: bold; font-size: x-large">CM3000 - Slave Device 3</td>
11. </tr>
12. <tr>
13. <td style="width: 300px; text-align: center">Frequency</td>
14. <td style="width: 90px; text-align: center"
    id="frequency"></td>
15. <td style="width: 100px; text-align: center">Hz</td>
16. </tr>
17. <tr>
18. <td style="width: 300px; text-align: center">Current Phase
    A</td>
19. <td style="width: 90px; text-align: center"
    id="currentphasea"></td>
20. <td style="width: 100px; text-align: center">Amps</td>
21. </tr>
22. <tr>
23. <td style="width: 300px; text-align: center">Current
    Neutral</td>
24. <td style="width: 90px; text-align: center"
    id="currentneutral"></td>
25. <td style="width: 100px; text-align: center">Amps</td>
26. </tr>
27. <tr>
28. <td style="width: 300px; text-align: center">Current
    Ground</td>
29. <td style="width: 90px; text-align: center"
    id="currentground"></td>
```

```
30. <td style="width: 100px; text-align: center">Amps</td>
31. </tr>
32. </table>
33. <br /><hr style="width:66%; height:1px" /><div style="text-
    align: center; font-size: small">&copy; 2011 Schneider
    Electric. All rights reserved.</div>
34. </form>
35. <script type="text/javascript">
36. if (window.XMLHttpRequest) {
37. // If IE7, Mozilla, Safari, and so on: Use native object
38. var xmlhttp= new XMLHttpRequest();
39. }
40. else
41. {
42. if (window.ActiveXObject) {
43. // ...otherwise, use the ActiveX control for IE5.x and IE6
44. var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
45. }
46. }
47. function getData() {
48. postString = "PL_" + "_3" +
    "^3209,3210,3211,3208,1180,1100,1103,1104" + "__PL";
49. doRead(xmlhttp, updateData, postString);
50. }
51. function doRead(xmlhttpObj, functionName, postString) {
52. var sampleRate = 5*1000;
53. try{
54. var temp;
55. var Data = new Array();
56. xmlhttpObj.open("POST", "Post__PL__Data", true);
57. xmlhttpObj.setRequestHeader("Content-Type", "application/x-
    www-form-urlencoded");
58. xmlhttpObj.onreadystatechange  = function() {
59. if(xmlhttpObj.readyState == 4){
60. try{
61. temp=xmlhttpObj.responseText;
62. Data=temp.split(",");
63. }
64. catch(exception){
65. ProcessError(xmlhttpObj.responseText);
66. return;
67. }
68. if(Data.length > 2){
```

```
69. functionName(Data);
70. setTimeout("getData()", sampleRate)
71. }
72. else{
73. ProcessError(Data);
74. }
75. }
76. }
77. xmlhttpObj.send(postString)
78. }
79. catch(exception){
80. ProcessError(exception);
81. }
82. }
83. function ProcessError(errTxt) {
84. alert(errTxt);
85. }
86. function updateData(Registers)
87. {
88. // Assign Scale Factors
89. ScaleFactorA = Registers[0];  // Current Scale Factor
90. ScaleFactorB = Registers[1];  // Neutral Current Scale Factor
91. ScaleFactorC = Registers[2];  // Ground Current Scale Factor
92. // Assign Nominal Frequency
93. NominalFrequency = Registers[3];
94. // Assign Data Values
95. Frequency      = Registers[4];
96. CurrentPhaseA  = Registers[5];
97. CurrentNeutral = Registers[6];
98. CurrentGround  = Registers[7];
99. // Get the current date and time
100.TheTime = new Date();
101.// Scale Frequency
102.if(NominalFrequency != 400)
103.{
104.Frequency *= 0.01;
105.}
106.else
107.{
108.Frequency *= 0.10;
109.}
110.// Scale Phase A Current
```

```
111.CurrentPhaseA *= Math.pow(10, ScaleFactorA);
112.// Scale Neutral Current
113.if (CurrentNeutral == 32768)
114.{
115.CurrentNeutral = "N/A";
116.}
117.else
118.{
119.CurrentNeutral *= Math.pow(10, ScaleFactorB);
120.}
121.// Scale Ground Current
122.if (CurrentGround == 32768)
123.{
124.CurrentGround = "N/A";
125.}
126.else
127.{
128.CurrentGround *= Math.pow(10, ScaleFactorC);
129.}
130.// Put data on the page
131.document.getElementById("frequency").innerHTML = Frequency;
132.document.getElementById("currentphasea").innerHTML =
    CurrentPhaseA;
133.document.getElementById("currentneutral").innerHTML =
    CurrentNeutral;
134.document.getElementById("currentground").innerHTML =
    CurrentGround;
135.document.getElementById("time_spot").innerHTML = TheTime;
136.}
137.</script>
138.</body>
139.</html>
```

Table 4: Description of XMLHTTPRequest HTML Source Code

HTML Code Line No.	Description
HTML Source for the Static Elements	
3	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 EGX300.
10	HTML syntax to write the title of the table "CM3000 - Slave Device 3".
13	HTML syntax to write "Frequency" table cell text label.
14	HTML syntax for the input control to be filled by dynamic data.
15	HTML syntax to write "Hz".
18, 19, 20	HTML syntax for displaying Current Phase A.
23, 24, 25	HTML syntax for displaying Current Neutral.
28, 29, 30	HTML syntax for displaying Current Ground.
JavaScripting Code for the Dynamic elements	
48	This line contains the following: “PL” delimiters at the beginning and end to signify to the EGX300 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 CM3000 data.
56	This line must have these arguments ("POST", "Post__PL__Data", true) for the open() function.
57	This line must have these arguments ("Content-Type", "application/xwww-form-urlencoded") for the setRequestHeader() function.
58-76	These lines define what to do “onreadystatechange”. They define a function that checks to see if the readyState is 4 (Request Complete). If so, it puts the data in an array and calls a function to process the data.
89	Register #3209 of the CM3000 has the Scale Factor A value.
90	Register #3210 of the CM3000 has the Scale Factor B value.
91	Register #3211 of the CM3000 has the Scale Factor C value.
93	Register #3208 of the CM3000 has the Nominal Frequency value.
95	Register #1180 of the CM3000 has the Frequency value.
96	Register #1100 of the CM3000 has the Current Phase A value.
97	Register #1103 of the CM3000 has the CurrentNeutral value.
98	Register #1104 of the CM3000 has the Current Ground value.
131-134	JavaScript statements to print variable values into their field.

Once you have created the HTML page, you must transfer this page to the EGX300. For details, see the EGX300 User Guide (63230-319-216A1).

Modifying an Existing Custom HTML Page

The following example modifies the CM3000 web page to a CM4000. The CM4000 has a device address of 1 on the EGX300 daisy chain. For this example, we will keep the web page layout and colors exactly the same. However, the real-time data tags must be changed to get information from the CM4000 rather than the CM3000. To do that, open the HTML source file using any web page or text editor and make the following changes:

HTML Modification

Replace line 4:

```
<title>CM3000 - Slave Device 3</title>
```

with:

```
<title>CM4000 - Slave Device 1</title>
```

Replace line 11:

```
CM3000 - Slave Device 3</b></font></p>
```

with:

```
CM4000 - Slave Device 1</b></font></p>
```

JavaScripting Source Code Modification

To receive the dynamic data from the CM4000 rather than the CM3000, change the CM3000 device address and register numbers to the CM4000 device address and its corresponding register numbers. Make the following changes:

Replace line 40/48:

```
Registers = [PL__3^3209,3210,3211,1180,1100,1103,1104__PL];
postString = "PL_" + "_3" +
"^3209,3210,3211,3208,1180,1100,1103,1104" + "_PL";
```

with:

```
Registers = [PL__1^3209,3210,3211,1180,1100,1103,1104__PL];
postString = "PL_" + "_1" +
"^3209,3210,3211,3208,1180,1100,1103,1104" + "_PL";
```

Once complete, save the file and transfer this page to the EGX300. See Figure 2.

Figure 2: Modified HTML page

CM4000 - Slave Device 1		
Frequency	60	Hz
Current Phase A	5	Amps
Current Neutral	4	Amps
Current Ground	N/A	Amps

Schneider Electric
Street Address
City, State Zip Country

For technical support:
Global-PMC-Tech-support@schneider-electric.com
(00) + 1 250 544 3010

Contact your local Schneider Electric sales representative for assistance or go to www.schneider-electric.us

63230-319-230A1 03/2011