SoftPLC Javascript HMI

Contents

1 Executive Summary

This HMI Interface provides a customizable client-side experience. By using simple HTML, XML, and Javascript, a full HMI experience can be created for any modern browser or mobile device. Combined with the web_tlm module, this HMI system can query and change values in the datatable of SoftPLC devices.

2 Installing the HMI Interface

Normally, your SoftPLC will already have the HMI Interface files installed for you if you purchased the Web Server add-on, Catalog Number SPZ-WEB.

If you are adding the HMI Interface to an existing SoftPLC, installing the HMI is as simple as extracting the files to your SoftPLC.

1. Connect to the SoftPLC by using your favorite SSH, FTP or SCP client like WinSCP or Putty.
2. Unzip the hmi.zip file to a temporary location.
3. Create a new folder /var/www/
4. Using your file transfer client utility program, copy the contents of the files unzipped to the temporary location to the /var/www/ folder
5. Copy any custom images you want to use to /var/www/images

3 Getting to know the HMI

The HMI is made up of only a couple of key files.

1. **production.js** - This contains all of the client-side logic. This should **not** be edited unless you have the development kit version of this file.
2. **templates/main.xml** - This is the primary configuration file for the application. Most of your changes will be made here for accessing the SoftPLC.
3. **index.html** – This is the main html interface file. You will start here. You can customize this file completely. There are no layout restrictions to this file. Feel free to make additional html files and link those in as well.

4 Examining the HTML

Let's take a look at a sample HTML file structure:

```
<!DOCTYPE HTML>
<html lang="en">
	<head>
		<title>HMI Sample</title>
	</head>
	<body refresh="2000">
		<h3>Welcome to the SoftPLC HMI Demo</h3>
		<ul>
			<li>Each control is specified in the template file by class id.<BR>
				<BR>Some Examples:<BR>
				<pre>
					&lt;div id="buttoncontrol"&gt;&lt;/div&gt;
					 - or -
					&lt;span id="buttoncontrol"&gt;&lt;/span&gt;
					 - or -
					&lt;p id="buttoncontrol"&gt;&lt;/p&gt;
			</pre>
			</li>
		<li>Simply specify the control & the rest is inherited from the configuration.</li>
			<li> </li>
		</ul>
		<hr>
		<b>Sample Controls:</b>
		<div id="main">
		<ul>
			<li>Test Button Control: <div id="testbutton1"></div><BR><BR></li>
			<li>Test Value:  <div id="testdata1"></div><BR><BR></li>
			<li>Test Status Light:  <div id="testlight1"></div><BR><BR></li>
		</ul>
		</div>
		<hr>
		<b><a href="http://softplc.com">softplc.com</a></b>

		<script src="steal/steal.production.js"></script>
		<script>
			steal('production.js');
		</script>
	</body>
</html>
```

a. Refresh Tag

First thing to notice is the refresh tag cycle:

```
<body refresh="2000">
```

This tells the HMI that this page should refresh every 2 seconds. Feel free to change this value to anything you like, based on your application, network speed, etc. For example, you may wish to see the data every 10 seconds. In this case, you would use:

```
<body refresh="10000">
```

If you create another HTML page for your HMI, make sure this tag is included if you want a dynamic refresh to occur.

b. Button Control

Let's take a look at a simple button control.

```
<li><div id="testbutton1"></div></li>
```

How do we know this is a button?

The ID gives it away. In this example, we have "id=testbutton1". The name testbutton is completely arbitrary. This id simply serves as a placeholder for the actual button. What determines that this **id** tag is a button is the **main.xml** file. We will look at that in the next section.

We are using an HTML DIV tag as our place holder. You could just as easily use a paragraph or span tag. Feel free to get creative.

Ie.

```
<span id="testbutton1"></span>   --- or ---   <p id="testbutton1"></p>
```

c. Value Placeholder

Just like the button above, we can have placeholders for simple values. This is determined by our main.xml file as well. We will discuss this in our next section. For now, this id is simply a placeholder for where our SoftPLC data will go.

```
<li><div id="testdata1"></div></li>
```

d. Status Light

A status light is simply a button without an option to change values by clicking. We do this by leaving out the <set> tag in the **main.xml**. This is described more in the **main.xml** section.

```
<li>Test Status Light:  <div id="testlight1"></div><BR><BR></li>
```

5 Examining the Heart of the HMI – MAIN.XML

Let's take a look at our sample **main.xml** file.

```
<?xml version="1.0" encoding="utf-8" ?>
<hmi>
	<item>
		<id class="button">testbutton1</id>
		<name>Test Button 1</name>
		<type>button</type>
		<img>images/toggleswitch_horizontal_on.png</img>
		<on value="1">images/toggleswitch_horizontal_on.png</on>
		<off value="0">images/toggleswitch_horizontal_off.png</off>
		<read ad='my_toggle_bit'></read>
		<set ad='my_toggle_bit'></set>
	</item>
	<item>
		<id class="formatdata">testdata1</id>
		<name>Test Display Data</name>
		<type>value</type>
		<format>numeric</format>
		<read ad='my_test_int'></read>
	</item>
	<item>
		<id class="light">testlight1</id>
		<name>Test Light 1</name>
		<label class="labelclass"> Light K19</label>
		<type>button</type>
		<img>images/pb_square_3dgreen.png</img>
		<on value="1">images/pb_square_3dgreen.png</on>
		<off value="0">images/pb_square_3dred.png</off>
		<read ad='my_status_bit'></read>
	</item>
</hmi>
```

This is actually quite simple. We need to look at each item as an individual element on the HTML page.

Each control (button, light, or otherwise) is defined with the <item> tag in the **main.xml.** Each tag contains one control. A couple of important points to note:

- Just because the control is defined in the **main.xml** file does not necessarily mean that you have to use the control on a webpage.
- The **main.xml** is only loaded the first time the HTML page is visited. This means that if you make changes to the file, you will need to refresh the page. This can be done with SHIFT-F5 on most browsers.
- Paths in the **main.xml** file are relative to the HTML page in which the control is displayed (unless a full path from the document root is used, ex: <img>/images/toggleswitch_horizontal_on.png</img>).
- In order to add a control to a page, you simply add the placeholder.
- It is highly recommended that you create tags for any addresses you wish to reference with the HMI in the SoftPLC's descriptor table using TOPDOC NexGen. The HMI does allow use of addresses instead of tags, but for readability and reference purposes it is easier to use tags.

a. Dissecting a Button

```
<item>
        <id class="button">testbutton1</id>
        <name>Test Button 1</name>
        <label class="labelclass"> Pushbutton PB23</label>
        <type>button</type>
        <img>images/toggleswitch_horizontal_on.png</img>
        <on value="1">images/toggleswitch_horizontal_on.png</on>
        <off value="0">images/toggleswitch_horizontal_off.png</off>
        <read ad='my_toggle_bit'></read>
        <set ad='my_toggle_bit'></set>
</item>
```

The id tag is required in order to define the placeholder and the elements on the HTML page. **The entire HMI application will fail if there is not an id tag for each element. You can include a class= attribute to add CSS elements to the button design if you so choose.

The name tag is optional. This simply provides an easy way for you to reference the controls you create. Think of this as your own personal name for the control. This is not used by the HMI application at this time.

The label tag is optional and can be used in ALL controls. Specifying the label tag will simply include a user-defined Label for the control in the HTML file immediately following the control. This can be formatted using CSS by simply specifying a class= attribute as shown above.

The type tag is required in order to tell the HMI application what sort of control this is. The following types are available:

- button – This indicates a clickable control. This type is also used for status lights.
- value – this simply indicates an output value direct from the PLC. Used for readouts and displays.

The img tag is required and is the default image to display for the button when the page first loads.

The on tag is required and is the image used with the button is in its ON state. If you will see in our example, we have a value= parameter included as well. The value parameter tells the HMI what value should be read from the PLC in order to display this image.

The off tag is required and is the image used with the button is in its OFF state. If you will see in our example, we have a value= parameter included just like the on tag. The value parameter tells the HMI what value should be read from the PLC in order to display this image.

The read tag is required and defines what address the data for the state of the button is read from. In our example, the address tag 'my_toggle_bit' references an actual address in the PLC (tags are configured using TOPDOC NexGen). For most buttons, a bit address (ex: B12/1) is appropriate since we only need two values: 0 or 1. If the PLC returns a 0 at that address, then our toggleswitch is off. If the PLC returns a 1 at that address, then our toggleswitch is on. This example uses SoftPLC descriptor tags rather than addresses.

The set tag is optional and defines what address is used to change the PLC. If the set tag is not specified the button will not be clickable. This is useful for a status light. Simply omitting the set tag allows for the creation of a status light.

In its current form, when a button is clicked, the opposite value is sent to the PLC as specified in the on and off tags.

For example, if the toggle switch is in its on state [value="1"] and the button is clicked. The HMI will send a 0 to the address referenced by 'my_toggle_bit' as specified here:

```
<set ad='my_toggle_bit'></set>
```

Once clicked, the HMI will know that the next state of the button needs to be 1. The HMI will send a 1 to address 'my_toggle_bit' as specified in the set tag.

b. Single State Buttons

If you would like to have a simple button that doesn't change state and always sends the same value, simply set the two on and off tags to be the same.

For example:

```
<on value="1">images/toggleswitch_horizontal_on.png</on>
<off value="1">images/toggleswitch_horizontal_on.png</off>
```

In the above example, whenever you click the button, you will always have the same "ON" image. Every click of the button will also send a 1 to the PLC.

c. Dissecting a Value Control

```
<item>
        <id class="formatdata">testdata1</id>
        <name>Test Display Data</name>
        <label class="labelclass"> Engine Speed </label>
        <type>value</type>
        <format>numeric</format>
        <read ad='my_test_int'></read>
        <set ad='my_test_int'></read>
</item>
```

All of the tags in the value control are exactly the same as in the button control. However, value controls only require the following:

```
<id></id>
<name> </name>
<type> </type>
<format> </format>
<read ad='my_test_int'></read>
```

The rest of the button tags would simply be ignored if included.

The type must be set to the word value (all lowercase)

The value is read and displayed from the PLC from the address specified in:

<read ad='my_test_int'></read>. For example, 'my_test_int' could reference an integer file value (N14:33).

<format> is not currently implemented and reserved for future formatting of the value.

If the set tag is included, the value is placed into a form input box. Changing the value will push the value to the PLC in the address specified in set ad= attribute. (ex: <set ad='my_test_int'></set>)

d. Dissecting a Status Light

```
<item>
        <id class="light">testlight1</id>
        <name>Test Light 1</name>
        <label class="labelclass"> Light K19</label>
        <type>button</type>
        <img>images/pb_square_3dgreen.png</img>
        <on value="1">images/pb_square_3dgreen.png</on>
        <off value="0">images/pb_square_3dred.png</off>
        <read ad='my_status_bit'></read>
</item>
```

Read ***Dissecting a Button* before reading this section.**

A status light is simply a button without a set tag.

The label tag is optional and can be used in ALL controls. Specifying the label tag will simply include a user-defined Label for the control in the HTML file immediately following the control. This can be formatted using CSS by simply specifying a class= attribute as shown above.

e. Bar Control

```
<item>
        <id>mybar1</id>
        <name>Bar 1</name>
        <type>bar</type>
        <color alpha="1.0">aqua</color>
        <min>0</min>
        <max>100</max>
        <thresholds>25|#ffd700 | 10|rgb(255, 0, 0)</thresholds>
        <read ad='my_bar_value'></read>
</item>
```

A bar control is used to display a value as a vertical bar from a min to max range (with labels to the left of the bar along the y-axis to indicate the range). The format is similar to the other control types, but several different tags are used to define the look and details of the bar.

The following tags function the same as they do for the other controls and are the only tags required:

```
<id></id>
<name> </name>
<type> </type>
<read ad='my_bar_value'></read>
```

The color tag is optional and defines the color of the bar itself. Colors are accepted in multiple formats:

- Simple named colors, e.g. red, lightred, darkred, magenta
- As RGB values (0-255), e.g. rgb(255, 0, 0)
- As hex values: #ffd700

The alpha attribute of the color tag is optional and defines the transparency of the bar. The value can range from 0.0 to 1.0, where 0.0 is fully transparent and 1.0 is opaque. The default color is aqua and the default alpha is 1.0.

The min tag is optional and defines the expected minimum value to be read from the read ad. If the tag is omitted, 0 is used.

The max tag is optional and defines the expected maximum value to be read from the read ad. If the tag is omitted, 100 is used.

The thresholds tag is optional and is used to change the bar color based on the current read value. The format is threshold|color where the threshold is a number (in the min-max range) below which the bar will change to the color specified. The color can be defined in the same formats as the color tag. To specify multiple thresholds, simply add another | to separate the definitions (as shown in the example above). The default thresholds are 25|gold | 10|red. To "clear" these just set a single threshold to 0|white (or substitute your minimum value for 0).

6 Notes on Performance

Remember to watch your refresh cycles. Setting an extremely low refresh cycle on the HMI could create race conditions and adversely impact the SoftPLC.

The “best” refresh rate will be a function of the type of SoftPLC hardware, the type of network connection, the number of connected browser clients, the number of points being monitored by the HMI, the type and quantity of I/O, other communication loads on the SoftPLC and other factors.

We recommend no less than 1000ms refresh cycle for the HMI.

7 Other HMI Functions

Additional controls are planned and can be added upon request. Contact SoftPLC.