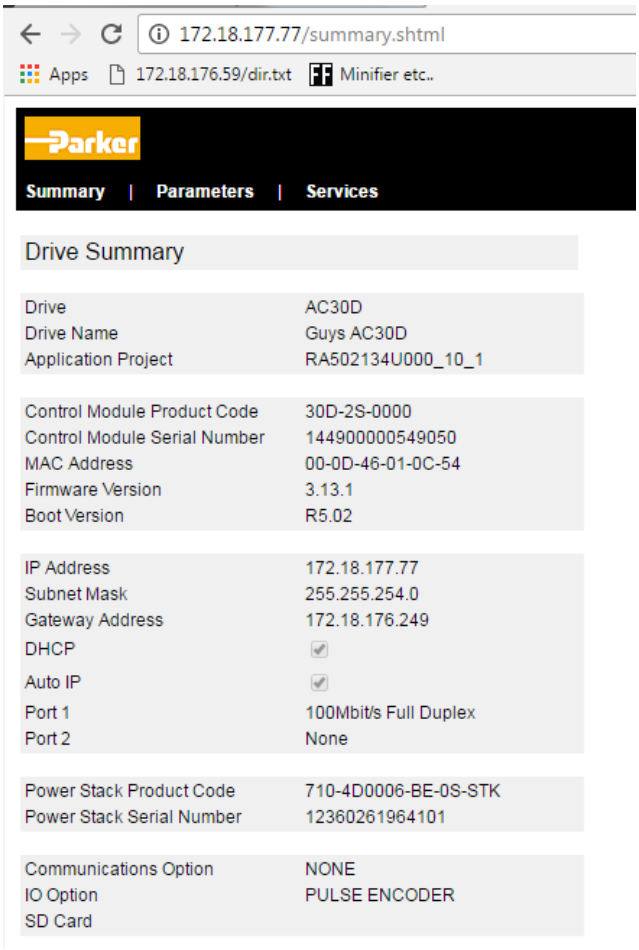
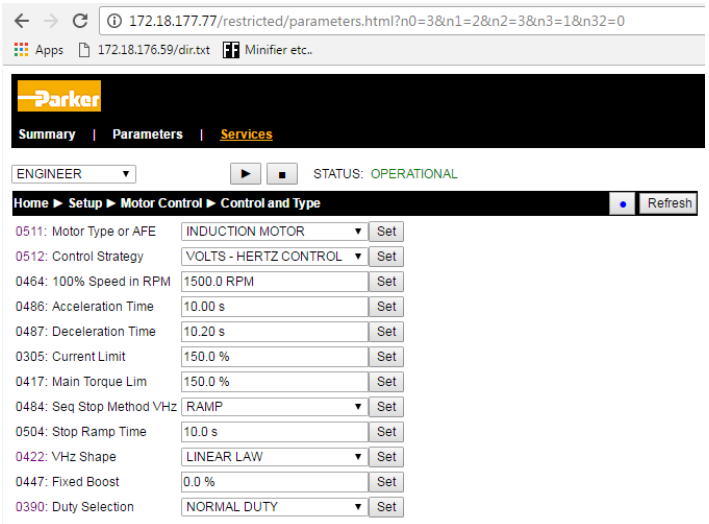
AC30 Custom Webpages

Built-In Webpages

The AC30 Series motor controller hosts a web server which provides quick and easy access to the drive's parameters via easy to use built-in webpages. The built-in webpages are accessed in the browser using the ipaddress of the drive which displays the summary page for the drive as shown below. Also shown below is a sample page from the Parameters link. Parameters can be monitored and edited using these pages.



Summary Page



Sample Parameters Page

Custom Webpages

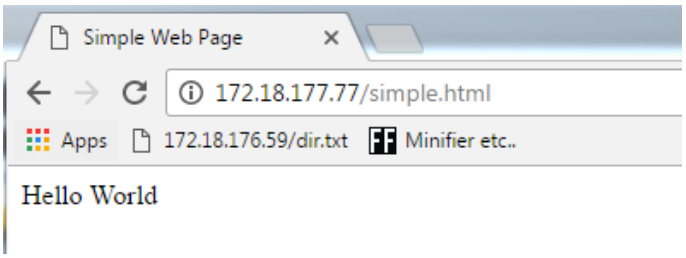
The AC30 can also serve up webpages created by the user. These webpages can be created in a text editor and then saved onto the drive's SD card. The webpages can be written in HTML, javascript and styled using standard css files.

Simple Webpage

The following example shows how this works by creating an html file called simple.html and saving it onto the SD card and then navigating to this using a browser.

```
<html>
<head>
   <title>Simple Web Page</title>
</head>
<body>
Hello World
</body>
</html>
```

File saved as simple.html to SD card



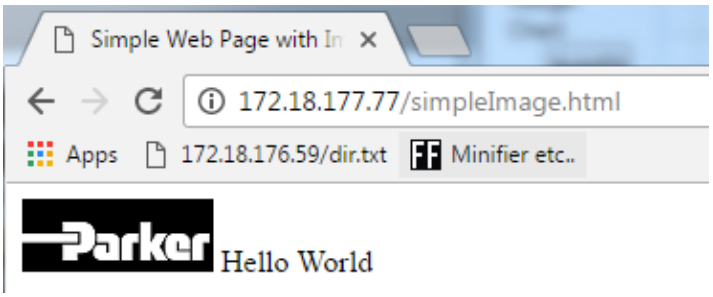
Webpage as seen in browser

Adding an Image

Images can be added to the webpage using the img html tag and copying the image to the SD card. In this example the image Parker_Hannifin.png has been saved to the folder 'assets/img/'. The file simpleImage.html is saved to the root folder.

```
<html>
<head>
   <title>Simple Web Page with Image</title>
</head>
<body>
<img src='assets/img/Parker_Hannifin.png' ></img>
Hello World
</body>
</html>
```

File saved as simpleImage.html to SD card



Webpage as seen in browser

Webpage Access to AC30 Parameters

The AC30 provides a number of service calls to access parameter information which are detailed in Appendix A. It is possible to write your own javascript to use these services directly but to simplify the process, a library of custom html elements has been written. As an example the following custom html element will read the value of parameter tag 696 or the *Drive Name.*

```
<param-label-value tag=”961”></param-label-value>
```

In order for custom elements to work, the assets folder and its contents should be copied to the root folder of the SD card.

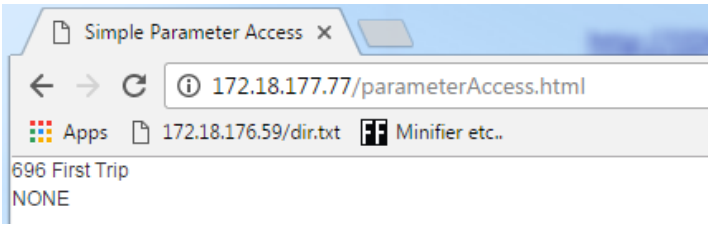
In addition to the assets folder the html page must also contain links to files in the assets folders. An example webpage using this custom element is shown below :-

```
<!DOCTYPE html>
<html ng-app="ac30WebApp">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Simple Parameter Access</title>
  <link rel="stylesheet" href="assets/bootstrap/css/bootstrap.min.css">
  <link rel="stylesheet" href="assets/css/styles.min.css">
</head>

<body ng-controller="driveSystemController">
  <param-label-value tag="696"></param-label-value>
</body>

<script src="assets/js/jquery.min.js"></script>
<script src="assets/bootstrap/js/bootstrap.min.js"></script>
<script src="assets/js/script.min.js"></script>
</html>
```

File saved as parameterAccess.html to SD card



Webpage as seen in browser

The custom elements also have custom attributes to change their behaviour and appearance.

For this custom element the following custom attributes are defined :-

The Fodefine

It is possible to change the appearance of the custom element by using the style attribute

reference files inside the asset folder.

<param-label-value tag='696' hidename="true"></param-label-value> tycontain links to theAs an example the following custom html tag will read the value of parameter 961 (Drive Name).

Appendix A

AC30 Web Services

The AC30 Series motor controller hosts a web server which provides quick and easy access to the drive's parameters via easy to use built-in webpages. In addition to this the drive also supports a number of HTTP web service calls which allow the user to access parameters information for use in their own custom web pages.

Drive FW: AC30V: 1.12 AC30P/D: 2.12

All examples below assume an AC30 with an ipaddress of 172.18.177.77

Single Parameter Tag Readers

In the following examples the parameter is Tag=1, *Analog Input Type*
3 modes:

1. Parameter Value and Type Information, Type is returned as an integer (eg 13 = Enum):

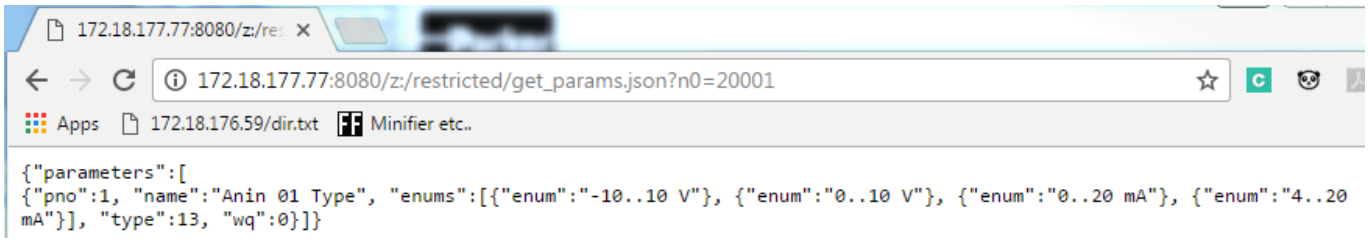
   http://{{DRIVEIPADDRESS}}:8080/z:/restricted/get_params.json?n0={{TAGNO}}

2. Parameter Value and Timestamp: Add 10000 to Parameter Tag

   http://{{DRIVEIPADDRESS}}:8080/z:/restricted/get_params.json?n0={{TAGNO+10000}}

3. Parameter Name and Enum Information: Add 20000 to Parameter Tag

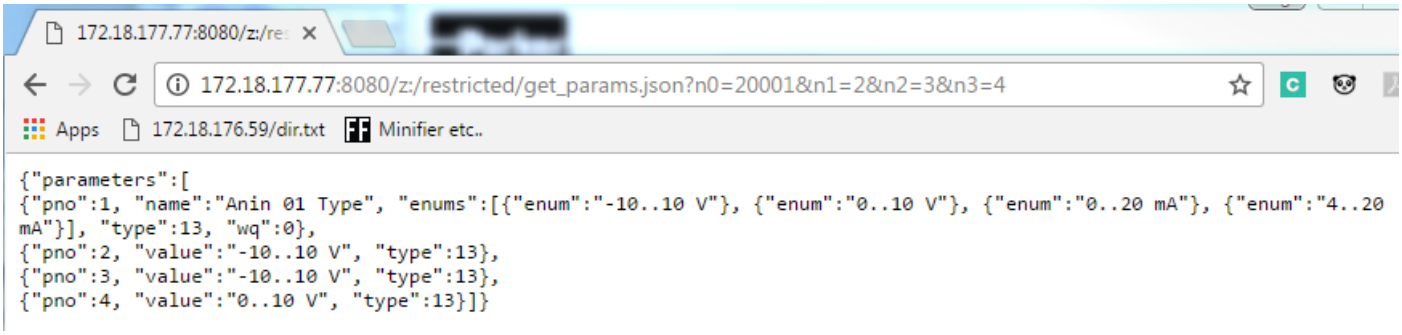
   http://{{DRIVEIPADDRESS}}:8080/z:/restricted/get_params.json?n0={{TAGNO+20000}}



Multiple Parameter Tag Readers

To read multiple parameters in one request add tags to the end of the URL in the following format

http://{{DRIVEIPADDRESS}}:8080/z:/restricted/get_params.json?n0=20001&n1=20002

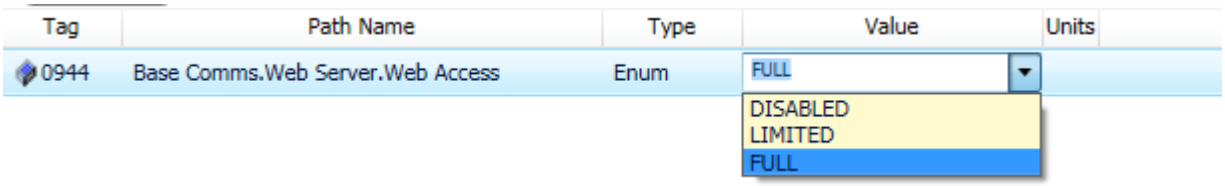


Note: Tag Readers do NOT require Web Access (Tag 944) since reads are made using 'always open' port 8080

Tag Writes detailed in the next section do require Web Access (Tag 944) set to FULL.

Parameter Tag Writers

To write parameter values to the drive the Web Access parameter, tag 944, must be set to FULL

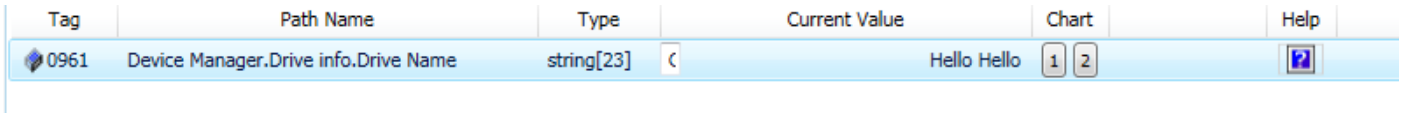


http: ://{{DRIVEIPADDRESS}}:8080/z:/restricted/parameters.act?V1=0

This example will set the value of Tag 1 to 0, which is the enum index 0 which is -10..10V

http: ://{{DRIVEIPADDRESS}}:8080/z:/restricted/parameters.act?V961=Hello%20Hello

This example sets the drive name to Hello Hello



Note strings must be URL Encoded as in this example where the space character is replaced by a %20

https://www.w3schools.com/tags/ref_urlencode.asp