

Application Note

C012 User Web Pages

HA502487C012 Issue B

AC30 Vx.18.x onwards

© Copyright 2019 Parker SSD Drives, a division of Parker Hannifin Ltd.

All rights strictly reserved. No part of this document may be stored in a retrieval system, or transmitted in any form or by any means to persons not employed by a Parker SSD Drives company without written permission from Parker SSD Drives, a division of Parker Hannifin Ltd. Although every effort has been taken to ensure the accuracy of this document it may be necessary, without notice, to make amendments or correct omissions. Parker SSD Drives cannot accept responsibility for damage, injury, or expenses resulting therefrom.

WARRANTY

Parker SSD Drives warrants the goods against defects in design, materials and workmanship for the period of 12 months from the date of delivery on the terms detailed in Parker SSD Drives Standard Conditions of Sale IA058393C.

Parker SSD Drives reserves the right to change the content and product specification without notice.



Requirements

Intended Users

This Application Note is to be made available to all persons who are required to install, configure or service equipment described herein, or any other associated operation.

The information given is intended to enable the user to obtain maximum benefit from the equipment.

Application Area

The equipment described is intended for industrial motor speed control utilising AC induction or AC synchronous machines.

Personnel

Installation, operation and maintenance of the equipment should be carried out by qualified personnel. A qualified person is someone who is technically competent and familiar with all safety information and established safety practices; with the installation process, operation and maintenance of this equipment; and with all the hazards involved.

Hazards

Refer to the Safety Information given at the front of the Product Manual supplied with every Parker SSD Drives product.

C012 USER WEB PAGES

Abstract

This application note explains how to create user web pages on an AC30 drive and provides some examples.

Pre-Requisite

The pre-requisites are an AC30 drive, SD card and a text editor such as Notepad++. Some basic knowledge of creating web pages would be needed.

Introduction

An HTTP web server is built into the AC30. Several built-in web pages are available by entering the drive's IP address into a web browser. The web pages including a Summary page and a Parameters page. The Parameters page allows the user to view and modify drive parameters.

A user may also create web pages by placing the required web files onto an SD card and/or, for AC30P and derivatives, onto the built-in flash file system.

An AC30 drive IP address of 192.168.1.10 is used in examples throughout this document.

The File System

A web browser accesses web pages from the AC30 web server in the form of files that are stored on three file systems within the AC30. These are the **SD card**, the **flash file system** (for AC30P and derivatives only) and a built-in **virtual file system**. A browser may access any of these files.

A user may add or remove files to the SD card or flash file system by using the **AC30 Drive Services** feature in PDQ/PDD found under the menu **File** and **Source Upload...** Alternatively, the SD card files may be modified using an SD card reader on a PC. Files may be placed in folders if required. The built-in virtual files cannot be modified.

Volumes

The three file systems are referenced explicitly using the following volumes. If the volume is not reference explicitly then the order the file will be searched is that shown in the Precedence column.

Supported on	Volume Letter	File System	Precedence
AC30V/P	a:	SD Card	1 st
AC30P	c:	Flash file system	2 nd
AC30V/P	z:	Virtual file system	3 rd

File Types

Any type of file can be accessed from the server. However, the server will add extra information to certain files to help the browser render the file.

File extension	File type
html, htm	HTML files
shtml, shtm	HTML files with SSI
css	Cascading style sheet files
txt	Simple text files
jpg, gif, png	Image files
js	Javascript files
json	JSON files
xml	XML files
mp3	MP3 files
gz	GZIP files
act	Action files (virtual file system only) used for form processing
Other	All other files are processed as attachments

Accessing Files

To access a file type from a browser, in the address bar type in the IP address of the drive followed by the filename complete with folder path. For example, to access file **hello.txt** that is in the root folder type:

192.168.1.10/hello.txt

As the filename does not specify the volume, the web server check all volumes until it is found in order of SD card, flash files and virtual files. Assuming the file is not on the SD card or flash file system then the file on the virtual file system will be accessed and the browser will display the contents:

Hello

Alternatively, the file could have been accessed by:

192.168.1.10/z:/hello.txt

In this case, *only* the virtual files are searched. **Note that this is also more efficient.**

Index File

If no file is specified when the IP address is entered into the browser, the file **index.shtml** found in the root folder will be assumed and loaded. All volumes will be checked as specified in the [Volumes](#) section. This allows a user to redirect to web pages on the SD card or flash system rather than displaying the built-in web pages simply by typing in the IP address

Restricted Files

Files on any of the three volumes that are store under the folder **/restricted** are password protected.

If the parameter **0944 Web Access** is set to **LIMITED**, then files in the folder cannot be accessed.

If the parameter **0944 Web Access** is set to **FULL** and parameter **0946 Web Password** is set, then the browser will request a case-sensitive username and password. The username is **ac30**.

Note the password protection uses **Basic Authentication** which, is the simplest form of web protection. This will protect against casual access only as the password can easily be intercepted.

Virtual Files

Some of the web files stored on the virtual file system are described below.

Filename	Description
/index.shtml	Index file used when no other is specified in the URL. This will redirect to the “Summary” web page file.
/summary.shtml	“Summary” web page
/restricted/parameters.shtml	“Parameters” web page
/parameters_ro.html	Read-only “Parameters” web page
/services.shtml	“Services” web page
/logo.gif	Company logo image
/hello.txt	Simple text file that displays “Hello”
/dir.txt	Displays the contents of the folder on the SD card or flash file system as a text file. If an optional query string is added of field-value pairs using the g tags, then: <ul style="list-style-type: none"> • g0 specifies the volume of either a or c. If it is not specified, then the SD card is assumed. • g1 specifies the path. If it is not specified, then the root folder is assumed.
/menus.xml	Displays all parameter menus and parameter names and data type in XML format.
/menus.json	Displays all parameter menus and parameter names and data type in JSON format.
/get_params.json	Used to get the current value and other attributes of the parameters specified by a query string of field-value pairs using the n tags (n0, n1, etc). The content is returned in JSON format. The data returned depends on the offset added to the parameter number in the n tag. See the table below.

Returned parameter attributes for **get_params.json** based on the PNO offset passed to the web server.

Offset	PNO	Value	Enum Numeric Value	English Enum String	Name	List of Enum strings	Units	Type	Time-stamp	Write Qualifier
0	✓	✓	✓	✓			✓	✓		
10000	✓	✓	✓	✓					✓	
20000	✓				✓	✓	✓	✓		✓
Other	Reserved									

Parameter Access

All AC30 inverter parameters may be accessed via the web server. Parameter attributes may be read using server side includes (SSI) and parameter values may be written via an HTML form.

The table below describes the tags used to reference parameters and their attributes. Note the tags are case sensitive.

When the tag is used to get the attribute when using the [#echo command](#), the tag is followed by the PNO and an optional dot (.) followed by an index value.

Parameter tag	Get/Set	Attribute description
C	Get	Returns “checked” if the Boolean parameter has a value of TRUE or if the enumerated parameter has a value that matches the index.
E	Get	Returns an error string after attempting to set the parameter to an invalid value. If there is no error an empty string will be returned.
H	Set	Used in a form using a checkbox input as a hidden input when setting Boolean parameters.
N	Get	Enumerated string of an enumerated, Boolean or bit-string parameter referenced by the index.
P	Get	Parameter’s name
S	Get	Returns “selected” if the Boolean or enumerated parameter has a value that matches the
U	Get	Parameter’s units
V	Get/Set	Value of a parameter.

The table below describes special tags do not access parameters.

Special tag	Get/Set	Description
R	Get/Set	Reserved attribute
X	Get	Returns internal AC30 strings
f	Set	Used to tell the server which filename to load after a form has been submitted
g	Set	Used to pass strings into a password ‘form’ or ‘disk’ form as field-value pairs (g0 ... g9)
n	Set	Used to pass numeric values to the web server as field-value pairs (n0 ... n29)

Server Side Includes (SSI)

Server Side Includes (SSI) allows a web server to modify an HTML file, that it has access to, as it is sending it to the client browser. The HTML file has a special filename extension: SHTML or SHTM. If a server supports SSI, then it knows that SHTML files are to be parsed before being sent to the browser.

SSI consists of several commands, starting with the # character, that are enclosed within an HTML comment so that if a web server does not support SSI, the browser will simply not show what is within the comment.

SSI commands detected within the file are replaced with new content depending on the command. For the AC30 two commands are available: #echo and #include. Currently #include is not supported for SD card or flash files.

#echo command

The **#echo** command is used on the AC30 only to display parameter attributes and some other special attributes described in the tables above. Standard environment variables are not supported.

An example of using the echo commands within an SHTML file is shown below. On load of the web page, the commands will be replaced with the parameter name (**P961**) and the parameter value (**V961**) for parameter **0961 Drive Name**. Note the format must be exactly as shown with a single space between echo and var, and a single space between the last quote (“) and the closing comment (-->).

```
<html>
  <body>
    <b><!--#echo var="P961" --> is <!--#echo var="V961" --></b><br>
  </body>
</html>
```

Browser Output:

Drive Name is Drive 1

Forms

Several built-in actions may be used by an HTML form to process data. They allow tags to be modified as describe in the section [Parameter Access](#).

These actions are files within the virtual file system. The file should be specified in the form **action** attribute.

Action file	Description
/restricted/parameters.act	Allows AC30 parameters to be modified. All settable tags are allowed.
/password.act	Allows a password to be changed.
/disk.act	Disk actions including file delete, file rename and make directory.
/generic.act	Only allows f , g and n tags to be set.
All other filenames	Any other filename used will be processed in the same way as generic.act . After a form has been submitted the page will be refreshed using this filename. This will be overridden if the f tag has been included in the form.

The form **method** attribute can be **POST** or **GET**, but POST is normal for forms.

To refresh the web page after a form is submitted, the HTML **input** element of type **hidden** may be used with the **name** attribute set to the **f** tag. The **value** attribute is set to the refresh filename – the **full path** must be specified. Note if a refresh filename is used that is in the **/restricted** folder then either the action or the HTML file in which the form is held must also be in the **/restricted** folder.

Changing Parameters

AC30 parameters may be changed using an HTML form.

The form **action** attribute should be set to the built-in action **parameters.act** stored in the folder **/restricted** in the virtual file system. As this action is in the **/restricted** folder it can be password protected.

Example 1 – Changing a numeric or string parameter

The example file (**form1.shtml**) is given below mixed with SSI. This example can be used with most parameters.

- The **input** element with attribute **type** set to **text** allows the user to enter text. The **name** attribute is set to the parameter to be changed using the **V** tag. The **value** attribute can be set to the initial value to be shown in the input box. This is done by using the **#echo** commands to read the current value and units of the parameter.
- **E486** will show an error if the last data entered is not valid for the parameter.

```
<html>
  <body>
    <b>Parameter 0486: <!--#echo var="P486" --></b><br>
    <form action="/restricted/parameters.act" method="post">
      <input type="text" name="V486" value="<!--#echo var="V486" --> <!--#echo var="U486" -->"/>
      <input type="hidden" name="f" value="/form1.shtml"/>
      <input type="submit" value="set"/><span style="color:red"> <!--#echo var="E486" --></span>
    </form>
  </body>
</html>
```

Browser Output:

Parameter 486: Acceleration Time
 Out of range

Example 2 – Changing a Boolean parameter using a checkbox

The example file (**form2.shtml**) is given below with mixed with SSI. This allows a Boolean-type parameter to be changed using a checkbox.

- The **input** element with **type** attribute set to **checkbox** must have the **value** attribute set to **1**. The **#echo** command with attribute **C** is included within the input element so that when the page is loaded the current state of the parameter is reflected by the checkbox.
- Due to the characteristic of a checkbox, if the checkbox is not set then no value is submitted. To overcome this an input element with the **type** attribute set to **hidden** can be used with the **name** attribute set to the **H** tag and the **value** attribute set to **0**. This hidden input **must** occur before the checkbox input.

```
<html>
<body>
<b>Parameter 0023: <!--#echo var="P23" --></b><br>
<form action="/restricted/parameters.act" method="post">
  <input type="hidden" name="H23" value="0"/>
  <input type="checkbox" name="V23" value="1" <!--#echo var="C23" -->/>
  <input type="hidden" name="f" value="/form2.shtml"/>
  <input type="submit" value="set"/>
</form>
</body>
</html>
```

Browser Output:

Parameter 23: Digout 01

☒

Example 3 – Changing an enumerated parameter

The example file (**form3.shtml**) is given below with mixed with SSI. This allows enumerated-type parameters to be changed from a drop-down list.

- The **select** element is used with the **option** elements.
- The **#echo** command with the **S** tag is included within the **option** element so that when the page is loaded the current value of the parameter is reflected in the selection. The **#echo** command **N** tag is to provide the enumerated string for each option.

```
<html>
<body>
<b>Parameter 0511: <!--#echo var="P511" --></b><br>
<form action="/restricted/parameters.act" method="post">
  <select name="V511">
    <option value="0" <!--#echo var="S511.0" --><!--#echo var="N511.0" --></option>
    <option value="1" <!--#echo var="S511.1" --><!--#echo var="N511.1" --></option>
    <option value="2" <!--#echo var="S511.2" --><!--#echo var="N511.2" --></option>
  </select>
  <input type="hidden" name="f" value="/form3.shtml"/>
  <input type="submit" value="set"/>
</form>
</body>
</html>
```

Browser Output:

Parameter 0511: Motor Type or AFE

PMAC MOTOR	<input type="button" value="set"/>
INDUCTION MOTOR	
PMAC MOTOR	
AFE	

Changing Passwords

AC30 password parameters may be changed using an HTML form. Currently only string password parameters can be changed.

The form **action** attribute should be set to the built-in action **password.act** stored in the root in the virtual file system.

The following **g** tags are used to pass information to the web server:

- g0 – the parameter number of the password parameter
- g1 – the old password requested from the user
- g2 – the new password requested from the user
- g3 – the confirmed password requested from the user

The following **E** tags return status / error information after the form is submitted:

- E5001 – Wrong password
- E5002 – String too long
- E5003 – Mismatched password
- E5004 – Password unchanged
- E5005 – Password changed

Example

The example file (**password.shtml**) is given below with mixed with SSI.

```
<html>
<body>
  <b>Parameter 0946: <!--#echo var="P946" --></b>
  <form action="/password.act" method="post">
    <input type="hidden" name="f" value="/password.shtml"/>
    <input type="hidden" name="g0" value="946"/>
    <table>
      <tr>
        <td>Old password: </td>
        <td><input type="password" name="g1"/></td>
        <td style="color: red"><!--#echo var="E5001" --></td>
      </tr>
      <tr>
        <td>New password: </td>
        <td><input type="password" name="g2"/></td>
        <td style="color: red"><!--#echo var="E5002" --><!--#echo var="E5004" --></td>
      </tr>
      <tr>
        <td>Confirm password: </td>
        <td><input type="password" name="g3"/></td>
        <td style="color: red"><!--#echo var="E5003" --></td>
      </tr>
      <tr>
        <td></td>
        <td align="right"><input type="submit" value="Change"/></td>
        <td style="color: green"><!--#echo var="E5005" --></td>
      </tr>
    </table>
  </form>
</body>
</html>
```

Browser Output:

Parameter 0946: Web Password

Old password:

New password:

Confirm password:

Password Changed

Disk Services

Disk services are available using the action **disk.act**. Note that this is only available on TCP port 8080 and is not actually intended to be used within forms. However, it could be used with AJAX.

The following **g** tags are used as field-value pairs to pass information to the web server:

Disk volume	Command	First filename	Second filename
g0	g1	g2	g3
a (sd card) c (flash file system)	ren (rename)	old filename	new filename
	del (delete)	filename	-
	md (make directory)	directory name	-
c (flash file system)	format	-	-

The response content from the server is either “DONE” or “FAILED”.

Style Sheets, JavaScript and AJAX

Style sheets and JavaScript may be inline, embedded within the HTML file or included as an external file.

Style sheet

An embedded style sheet is defined within the `<style>` element within the `<head>` section.

External style sheet files are included using the `<link>` element within the `<head>` section:

```
<head>
  <link rel="stylesheet" type="text/css" href="my_style_sheet.css">
</head>
```

JavaScript

Embedded JavaScript is defined within the `<script>` element within the `<head>` or `<body>` section.

External JavaScript files are included using the `<script>` element within the `<head>` or `<body>` section

```
<script src="my_javascript.js"></script>
```

AJAX

Modern browsers have the built-in **XMLHttpRequest** JavaScript object that allows data to be requested by a browser from a server asynchronously. This allows a web page to be updated without having to reload the page.

The JSON file **get_params.json** has been provided in the AC30 [virtual file system](#) that returns the value and other attributes of requested parameters and can be used with the XMLHttpRequest object.

Cross-Origin Resource Sharing (CORS) is possible. For example, the web files may be placed on one AC30 (or a PC) and the HTTP request made to a different AC30.

Example

The following example demonstrates the use of the XMLHttpRequest object. It consists of an HTML and a JavaScript file. The value of two AC30 parameters (**1139 Control Board Up Time** and **1733 Time Since Power-On**) are displayed in the browser and continually update at a rate of 250ms.

For the XMLHttpRequest, as the GET method is used in the **open()** function, the parameter PNOs are passed via the URL. However the POST method could be used and the PNOs placed in the content of the **send()** function. Either way, the parameters are added as a query string composed of field-value pairs separated by **&**, i.e. **n0=1139&n1=1733**.

The returned data is in JSON format. The **JSON.parse()** function creates an array of all the parameters requested. In this case as there are two parameters the array size is 2. The attributes of the parameters are available within each element of the array, such as the value attribute.

Timeout functions are used to repeat the request every 250ms, to check for a loss of communications and attempt a re-request if communications are lost.

If the file are placed, for example, on a PC using cross-origin resource sharing then the **open()** function should specify the full path, i.e. **http://192.168.1.10/z/get_params.json?...**

HTML file (ajax.html)

```
<!-- HTML page demonstrating AJAX
      Parameter 1139 Control Board Up Time
      Parameter 1733 Time Since Power-On -->
<html>
  <head>
    <script src="ajax.js"></script>
  </head>
  <body onLoad="loadFunc(1139, 1733)">
    <div id="param1"></div><br>
    <div id="param2"></div><br>
  </body>
</html>
```

JavaScript file (ajax.js)

```
/*
  AJAX Example.
  Asynchronous request of AC30 parameters at a rate of 250ms.
  Values returned from the server are used to modify the HTML page.
  Includes a timeout and retry function if communications are lost.
*/

/* Global variables */
var xmlhttp;
var t;
var pno1, pno2;

/* Page load function called in ajax.html */
function loadFunc(p1, p2){
  if (window.XMLHttpRequest) {
    xmlhttp=new XMLHttpRequest();
    xmlhttp.abort();
    pno1 = p1;
    pno2 = p2;
    httpRequest();
  }
} // loadFunc()

/* HTTP request and response function */
function httpRequest(){
  xmlhttp.onreadystatechange=function() {
    if (xmlhttp.readyState==4 && xmlhttp.status==200) {
      var obj=JSON.parse(xmlhttp.responseText);
      var value1 = obj.parameters[0].value;
      var value2 = obj.parameters[1].value;
      setElements(value1, value2)
      setTimeout("httpRequest()", 250);
      clearTimeout(t);
    }
  }

  xmlhttp.open('GET', "/z:/get_params.json?n0="+pno1+"&n1="+pno2, true);
  xmlhttp.send(null);
  t = setTimeout("timedOut()", 4000);
} // httpRequest()

/* Timeout function if HTTP request fails to respond */
function timedOut(){
  setElements("---", "---")
  xmlhttp.abort();
  setTimeout("httpRequest()", 3000);
} // timedOut

/* Modifies the elements in ajax.html */
function setElements(value1, value2) {
  document.getElementById("param1").innerHTML = "PNO " + pno1 + " is " + value1;
  document.getElementById("param2").innerHTML = "PNO " + pno2 + " is " + value2;
} // setElements()
```

Browser Output:

PNO 1139 is 3701119

PNO 1733 is 7:07:58.826