Making an OPC UA client with MATLAB using the Industrial Communication Toolbox

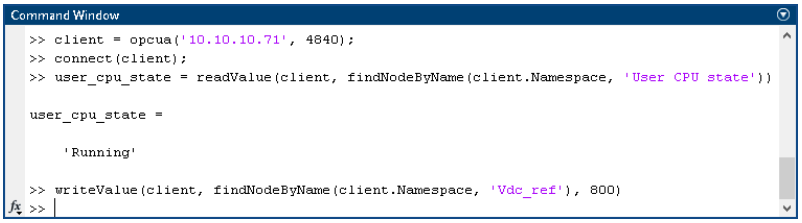
PN178 | Posted on March 30, 2022 | Updated on May 7, 2025

Benoît STEINMANN
Software Team Leader
imperix

Table of Contents

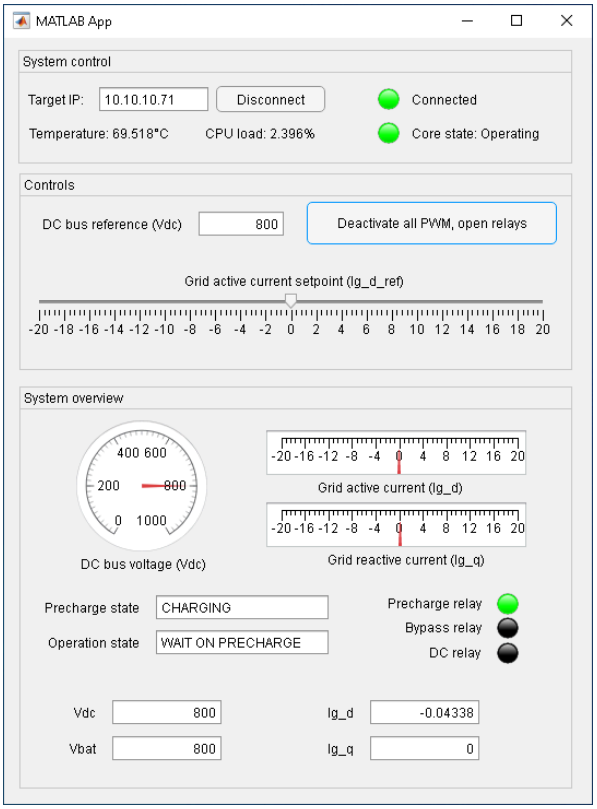
The Industrial Communication Toolbox Add-On for MATLAB and Simulink (formerly OPC Toolbox) allows communicating with an OPC UA server from MATLAB. This page explains how to use the Industrial Communication Toolbox to monitor and control a B-Box digital controller by connecting to its embedded OPC UA server.

The OPC UA variables provided by the imperix controllers are documented on imperix OPC UA server.



Read and write OPC UA variables from MATLAB using the Industrial Communication OPC UA toolbox

The Industrial Communication Toolbox can be used in combination with MATLAB App Designer to easily create an HMI Graphical User Interface (GUI) to remotely control an imperix power electronic controller as shown below.



Example of custom-made GUI to interact with an imperix controller

This page provides an overview of the capabilities of the MATLAB Industrial Communication Toolbox OPC UA client. Please refer to the official MATLAB Unified Architecture documentation for an exhaustive list of available features.

Connecting to an OPC UA server using MATLAB

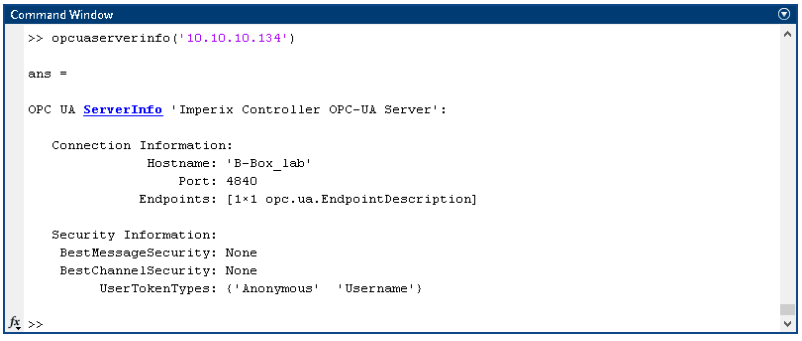
The Industrial Communication Toolbox `opcuaserverinfo` function returns an OPC UA ServerInfo object containing information on the OPC UA server, such as the server description and hostname. The `opcua` function creates an OPC UA client object to which it is possible to `connect` or `disconnect`. The function `isConnected` returns the connection status (0 or 1);

```
info = opcuaserverinfo('10.10.10.134');
client = opcua(info);
connect(client);
isConnected(client);
disconnect(client);Code language: Matlab (matlab)
```

Alternatively, the `opcua` function can directly take an IP and port as parameters.

```
client = opcua('10.10.10.134', 4840);Code language: Matlab (matlab)
```

If the function `opcuaserverinfo` works (it returns the target information as shown in the screenshot below) but `opcua` throws a “not enough input arguments” error, it is probably due to Windows failing to resolve the hostname. You may need to edit the file **C:\Windows\System32\drivers\etc\hosts** and manually add the “hostname/IP” pair. The issue is further discussed in the troubleshooting section below.



Expected output of the MATLAB opcuaserverinfo function

```
Command Window
>> client = opcua('10.10.10.134', 4840)

client =

OPC UA Client:

   Server Information:
                    Name: 'Imperix Controller OPC-UA Server'
                Hostname: 'B-Box_lab'
                    Port: 4840
             EndpointUrl: 'opc.tcp://B-Box_lab:4840/'

   Connection Information:
                 Timeout: 10
                  Status: 'Disconnected'
             ServerState: '<Not connected>'

   Security Information:
      MessageSecurityMode: None
    ChannelSecurityPolicy: None
                Endpoints: [1×1 opc.ua.EndpointDescription]

fx >>
```

Expected output of the MATLAB opcua function

Read and write OPC UA variables using MATLAB

Two steps are required to read or write an OPC UA variable value using the Industrial Communication Toolbox OPC UA client. First, an OPC UA server node object must be created using the `findNodeByName` function. Then simply use the `readValue` of `writeValue` function to read or write the OPC UA variable.

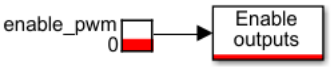
```
% read the B-Box CPU load system variable
cpu_load_node = findNodeByName(client.Namespace, 'CPU load [%]');
readValue(client, cpu_load_node);

% write to a tunable parameter user variable
writeValue(client, findNodeByName(client.Namespace, 'Vdc_ref'));
```
Code language: Matlab (matlab)

```
Command Window
>> cpu_load_node = findNodeByName(client.Namespace, 'CPU load [%]');
>> readValue(client, cpu_load_node)

ans =

  single

    5.3560

>> cpu_state_node = findNodeByName(client.Namespace, 'User CPU state');
>> readValue(client, cpu_state_node)

ans =

    'Running'

fx >>
```

Enabling and disabling PWM outputs using OPC UA

Unfortunately, the MATLAB Industrial Communication Toolbox OPC UA client does not support OPC UA method calls. This means that the imperix OPC UA methods “Enable outputs” and “Disable outputs” can not be directly called. The same functionality can be implemented by using the enable outputs block as shown below.

Enabling the PWM outputs must be done with the utmost precaution to avoid causing serious damage to the converter.

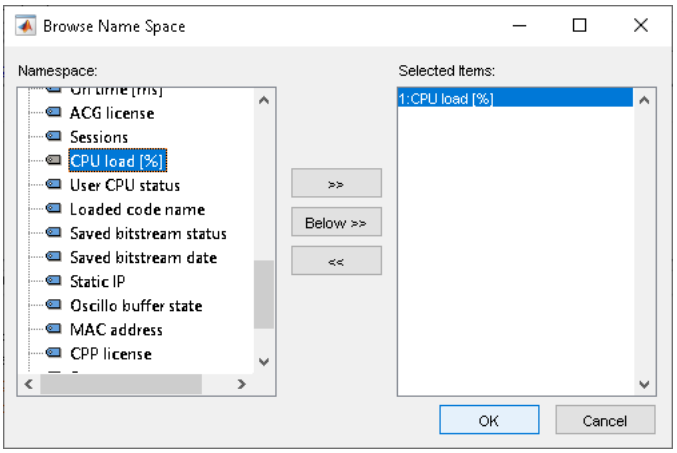


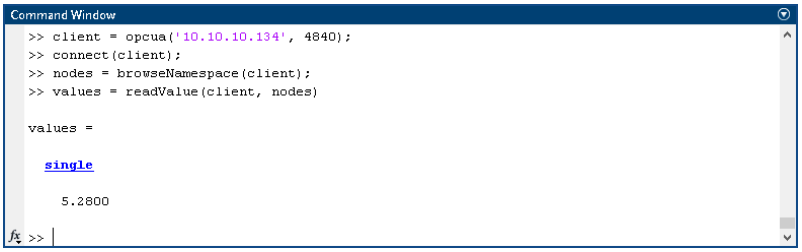

```
Command Window
>> readValue(findNodeByName(client.Namespace, 'Core State'))

ans =

    'Blocked'

>> writeValue(findNodeByName(client.Namespace, 'enable_pwm') , 1)
>> readValue(findNodeByName(client.Namespace, 'Core State'))

ans =

    'Operating'

>> writeValue(findNodeByName(client.Namespace, 'enable_pwm') , 0)
>> readValue(findNodeByName(client.Namespace, 'Core State'))

ans =

    'Blocked'

fx >>
```

Browsing OPC UA variables using MATLAB OPC Toolbox

Instead of explicitly providing the OPC UA variable name with `findNodeByName`, the function `browseNamespace` opens the Browse Name Space containing the list of available OPC UA nodes. When the OK button is clicked, `browseNamespace` returns the OPC UA server node object that can be passed as a parameter to the `readValue` or `writeValue` functions.

```
nodes = browseNamespace(client);
values = readValue(client, nodes);
```





App Designer GUI as an OPC UA client

An App Designer GUI can easily be used as an OPC UA client thanks to the Industrial Communication Toolbox. This section presents the main structure of a GUI used as an OPC UA client. For a more detailed tutorial on how to create graphical user interfaces with MATLAB App Designer, please refer to the page Graphical User Interface with MATLAB App Designer.

In the code view of App Designer, the first step for the creation of an OPC UA client is to declare a `client` as a property so that it is accessible to all functions and callbacks within the app.

```
    properties (Access = private)
        client ;
    endCode language: Matlab (matlab)
```

Then, in the GUI's startup function (`StartupFcn`), assign an OPC UA client as an `opc.ua.client` object to the previously declared property.

```
        % Code that executes after component creation
        function startupFcn(app)
            app.client = opc.ua.Client;
        endCode language: Matlab (matlab)
```

Then, a push button with the `connect` and `disconnect` function in its callback can be used to connect and disconnect to/from the OPC UA server. The server's IP address can be entered inside an `EditField` UI for instance.

```
% Button pushed function: ConnectButton
        function ConnectButtonPushed(app, event)

            if isConnected(app.client) == 1
                disconnect(app.client);
                app.client = opc.ua.Client;
            else
                ip = app.TargetIPEditField.Value;
                try
                    info = opcuaserverinfo(ip);
                    hostname = info.Hostname;
                    try
```

```
                    app.client = opcua(info);
                    connect(app.client);
                catch
                    msgbox({['Target ' ip ' (' hostname ') found but failed to connect. '];''; ...
                        ['Note: due to a limitation of the Industrial Communication Toolbox, you may need to e
                end
            catch
                msgbox(['Failed to connect to ' ip '.']);
            end
        end
    endCode language: Matlab (matlab)
```

Finally, the aforementioned read/write methods can be used in the callbacks of timers or UI components to interact with the available OPC UA variable. The code below gives an example of how these methods can be used in the callback of a push button.

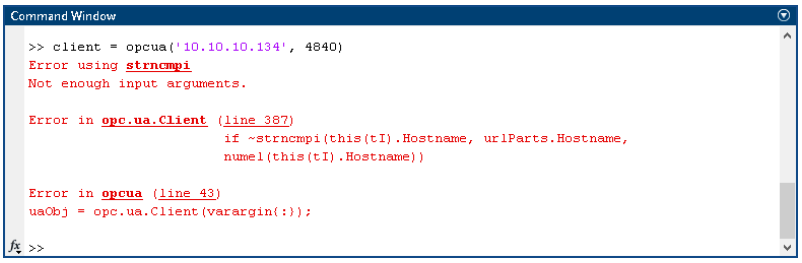
```
    % Button pushed function: PrechargebusandactivatePWMButton
    function PrechargebusandactivatePWMButtonPushed(app, event)
        if UserCodeIsRunning(app)
            activate_cmd_value = readValue(app.client, findNodeByName(app.client.Namespace, 'activate'));
            if activate_cmd_value == 0
                writeValue(app.client, findNodeByName(app.client.Namespace, 'activate'), 1);
                app.PrechargebusandactivatePWMButton.Text = 'Deactivate all PWM, open relays';
            else
                writeValue(app.client, findNodeByName(app.client.Namespace, 'activate'), 0);
                app.PrechargebusandactivatePWMButton.Text = 'Precharge bus and activate PWM';
            end
        end
    endCode language: VHDL (vhdl)
```

MATLAB Industrial Communication Toolbox troubleshooting

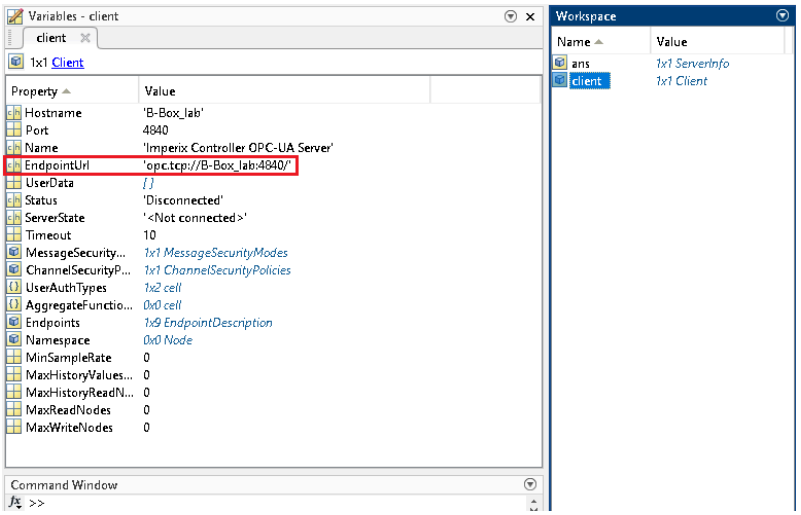
MATLAB opcua function “not enough input arguments” error

If the function `opcuaserverinfo` works but `opcua` throws the “not enough input arguments” error shown below, even when using an IP address, it is probably due to Windows failing to resolve the hostname.

```
>> client = opcua('10.10.10.134', 4840)
Error using strncmpi
Not enough input arguments.

Error in opc.ua.Client (line 387)
                        if ~strncmpi(this(tI).Hostname, urlParts.Hostname,
                        numel(this(tI).Hostname))

Error in opcua (line 43)
uaObj = opc.ua.Client(varargin{:});Code language: Matlab (matlab)
```

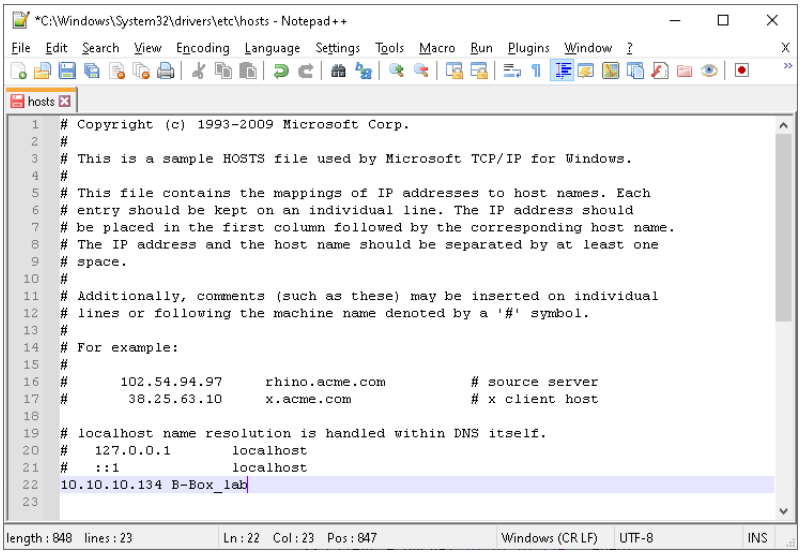


It comes from a limitation of the Industrial Communication Toolbox OPC UA client. As can be seen in the screenshot below, even when an IP address is directly provided to the `opcua` function, the toolbox will try to connect using the hostname instead.



If opcua throws an error, it probably means that Windows failed to resolve the hostname. As shown in the image below, the solution is to edit the file **C:\Windows\System32\drivers\etc\hosts** and manually add the “hostname/IP” pair.

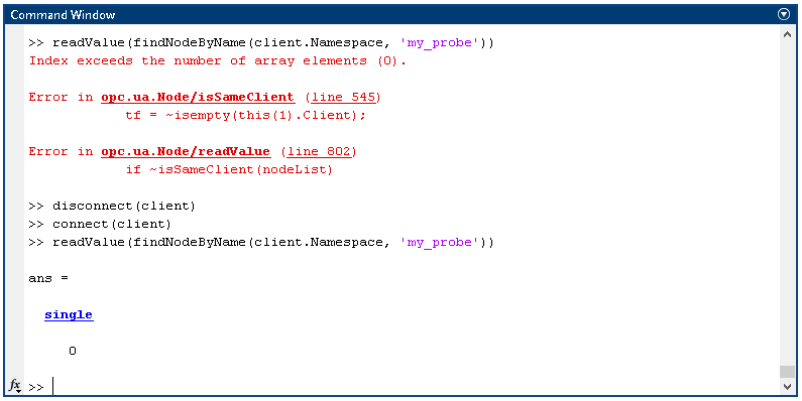
You need administrator rights to edit **C:\Windows\System32\drivers\etc\hosts**.



MATLAB readValue “index exceeds the number of array elements (0)” error

If the readValue function generates a “index exceeds the number of array elements (0)” error, but you now that the OPC UA node exists, it may be because the OPC UA client is not up to date and needs to rebrowse the OPC UA nodes. A solution is to `disconnect` and `connect` again as shown in the screenshot below.

```
>> readValue(findNodeByName(client.Namespace, 'my_probe'))
Index exceeds the number of array elements (0).

Error in opc.ua.Node/isSameClient (line 545)
            tf = ~isempty(this(1).Client);

Error in opc.ua.Node/readValue (line 802)
            if ~isSameClient(nodeList)Code language: VHDL (vhdl)
```


Command Window
>> readValue(findNodeByName(client.Namespace, 'my_probe'))
Index exceeds the number of array elements (0).
Error in opc.ua.Node/isSameClient (line 545)
tf = ~isempty(this(1).Client);
Error in opc.ua.Node/readValue (line 802)
if ~isSameClient(nodeList)
>> disconnect(client)
>> connect(client)
>> readValue(findNodeByName(client.Namespace, 'my_probe'))
ans =
single
0
>>