

 AUTOMATIONDIRECT.com

**CLICK PLUS**

**Applications Guide**

# CLICK PLUS Applications Guide

---



Please include the Manual Number and the Manual Issue, both shown below, when communicating with Technical Support regarding this publication.

Manual Number: C2-APP-GUIDE  
Issue: 1st Edition, Rev. C  
Issue Date: 11/2025

Publication History		
Issue	Date	Description of Changes
1st Edition	02/2025	Original
1st Edition Rev. A	02/2025	Moved importable project code to github.
1st Edition Rev. B	04/2025	Added link to Node-RED module compatibility utility.
1st Edition Rev. C	11/2025	Added <i>C2-OPCUA Module Connection to UAExpert</i> section

# TABLE OF CONTENTS

Top 10 things to know before starting with CLICK.....	1
Common PLC Programming Patterns.....	2
Flashing Coil .....	2
Single Timer Flash.....	2
Drum Timers for Advanced Timer control .....	3
Input Map.....	3
Debounce .....	3
Analog Input Filter.....	5
State Machine Programming.....	7
Motion Control.....	7
Advanced Math Logic .....	9
Min/Max.....	9
Average .....	9
Convert to Percent .....	10
Node-RED Module Compatibility .....	10
Node-RED Project Examples.....	11
PROJECT: Initiate a Node-RED flow from a CLICK event.....	11
Project Summary:.....	11
Importable Project:.....	12
PROJECT: Find the Max Value.....	13
Importable Project:.....	14
PROJECT: Find your IP Address .....	14
PROJECT: Connect to a Microsoft SQL Azure Database.....	15
SQL Select.....	16
SQL Update .....	17
SQL Insert.....	18
SQL Delete .....	18
PROJECT: Connect to the local SQLite Database .....	20
Pros and Cons of using a local database: .....	20

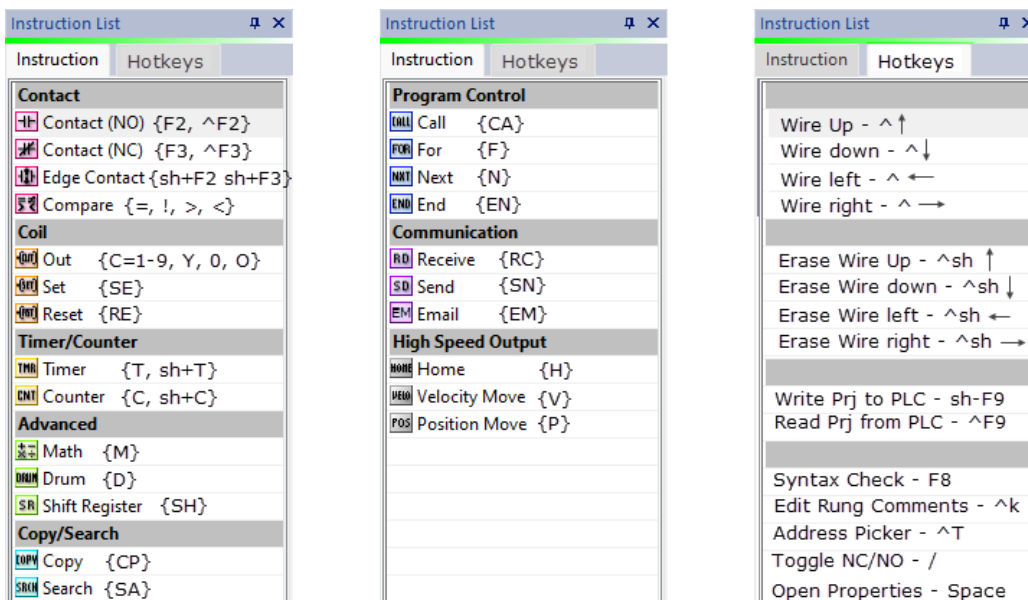
## Table of Contents

---

Frequently Asked Questions: .....	20
Minimum configuration to use SQLite .....	21
<b>PROJECT: Call a Web Service to retrieve data and chart the results.....</b>	<b>23</b>
Project Summary:.....	23
Importable Project.....	24
<b>PROJECT: Local Weather Station .....</b>	<b>25</b>
<b>PROJECT: Create an MQTT Broker.....</b>	<b>28</b>
Importable Project.....	28
<b>PROJECT: Send a Barcode value to Node-RED via HTTP Get, parse it into fields on the dashboard, and perform a SQL Lookup .....</b>	<b>29</b>
Importable Project:.....	30
<b>BONUS FUNCTION: Process Modbus - Four message types using JavaScript to create array of values: .....</b>	<b>31</b>
<b>C2-OPCUA Module Connection to UAExpert.....</b>	<b>32</b>
Setup.....	32

## Top 10 things to know before starting with CLICK

### 1. Use Hotkeys!



^ = ctrl and sh=shift

2. The Drum timer can be used as either a multistage timer (Blinker) or as a state machine controller.
3. Use Input maps.
4. Subroutines leave I/O values in their last known state if the subroutine stops running. An output that was on the last time a subroutine runs will stay on when the subroutine is disabled.
5. You can always find the IP address of your CLICK (Ethernet) in the footer of the programming tool.
6. You can always find the subroutine name in the CLICK Program header.
7. CLICK supports low code programming for High-Speed I/O, PID, MQTT, and Drum timers – review these features before you start programming, they can save you a ton of time.
8. XD/YD data types allow an easy way to set or send up to 16 bits at a time.
9. There is a high-speed programmable limit switch under the High-Speed I/O setup – look for “Presets”.
10. CLICK is great for motion control and can be used for simple motion AND more complex motion like registration moves, velocity control, linear interpolation, and relative or absolute positioning.

## Common PLC Programming Patterns

If you are new to ladder logic, the following are several of the most common patterns used. It's a good idea to make sure you understand these before you write your first program.

### Latching Coil

The most common circuit is a latching circuit or Start/Stop circuit. This can be accomplished with either a single rung or you can use a latching output.



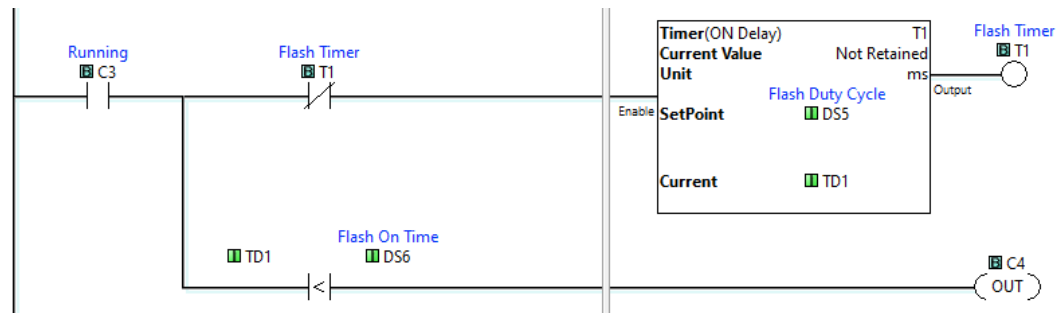
-or-



### Flashing Coil

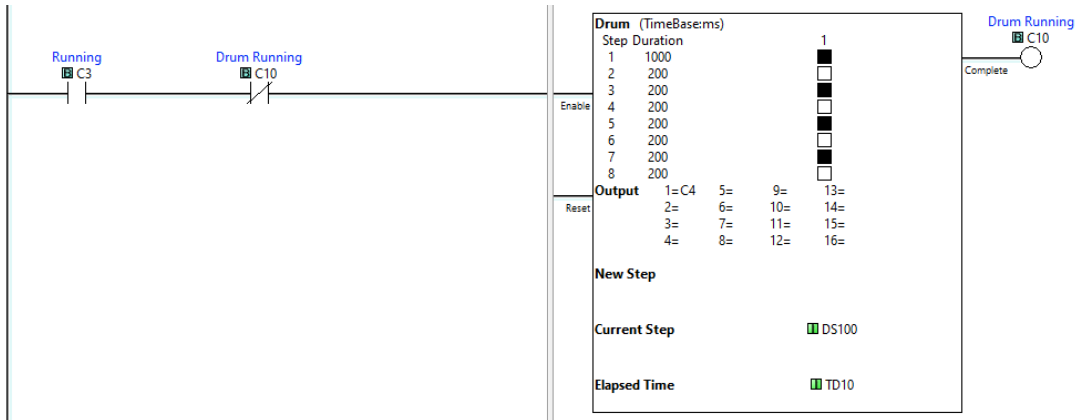
#### Single Timer Flash

Another one of the most common circuits is to create a flasher or on/off cycle. Here is an example of an easy one timer circuit that allows independent control of the on and off times.



In this example, the timer automatically resets when it is done. As long as the timer value is less than the "On" time (DS6), the output C4 will be on, after that C4 will turn off. So if DS5 is 3s, and DS6 is 1s, C4 will be on for 1 second, then off for the remaining 2 seconds in the timer. At the 3 second mark the timer will reset to 0 and C4 will turn back on. That cycle will repeat until C3 turns off.

## Drum Timers for Advanced Timer control



Drum timers make complicated things look easy. The example above shows how to use a drum to create one long blink followed by 3 short blinks. This happens to be the letter “B” in morse code.

## Input Map

This is not really a circuit, but great programming practice to make your system easier to maintain. It also prevents issues from Asynchronous I/O scans (not an issue with CLICK). If you put this in a subroutine, this also gives you a place to apply debounce timers, and create flashers so they don't interfere with the readability of you main program flow.

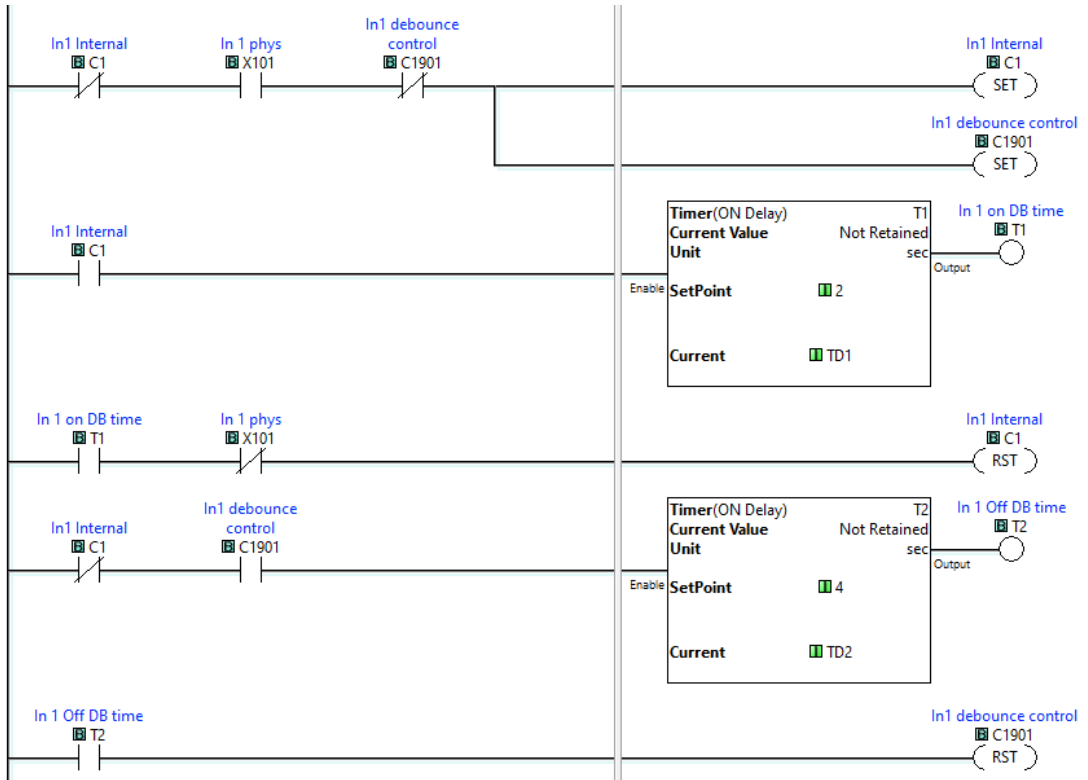
## Debounce

If your circuit uses physical relays or mechanical switches, you may encounter bouncing. This happens because the physical switch is made from a flexible copper wire that bends when the switch or relay changes state. That bending can allow the wire to bounce off the contact point when it opens or closes. Modern PLCs are fast enough to detect the rapid on-off changes that happen when a switch bounces. You may need to handle this in your program by either latching on a circuit so it stays on through the bounce (fast-acting), or delaying the “On” event until the switch has stabilized (slow-acting).

# Common PLC Programming Patterns

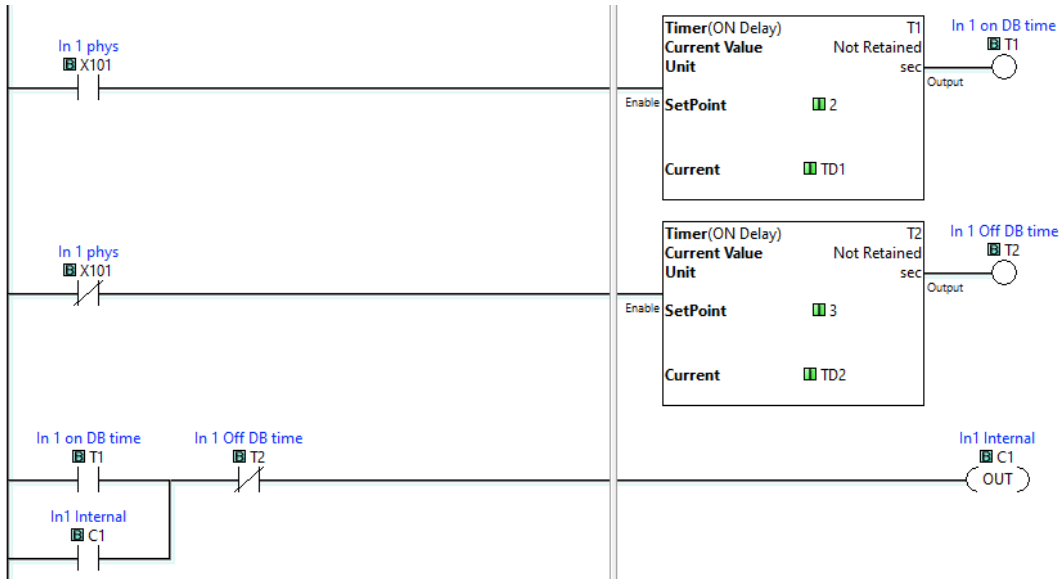
## On-Delay, Off-Delay Debounce (Fast-acting)

When an input turns on, capture the on state and hold it for a minimum time to allow the input to stabilize. Fast acting can potentially capture transient voltage spikes, so use sparingly.



## On-Delay, Off-Delay Debounce (Slow-acting)

When an input turns on, wait for a minimum time to allow the input to stabilize before enabling the circuit. This approach eliminates potential spikes, but delays the recognition of an on event for a specified time.

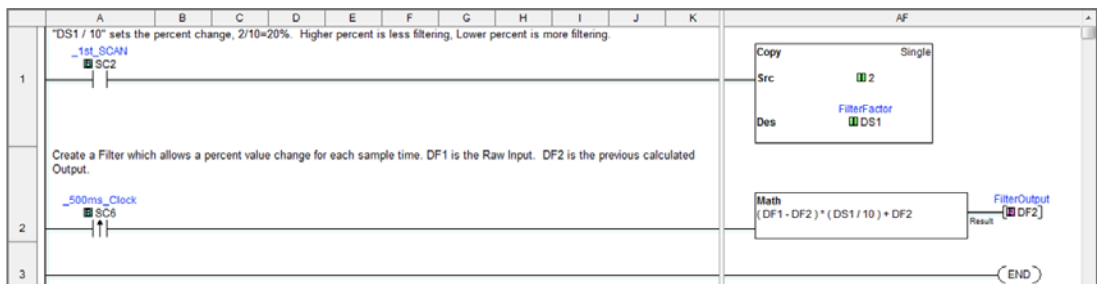


## Analog Input Filter

Two methods of creating an Analog Filter in Ladder Logic are shown here. These can be useful for filtering analog inputs. The following examples provide simple methods to create a filter and average. Be sure to change the addresses to unused memory addresses in your project.

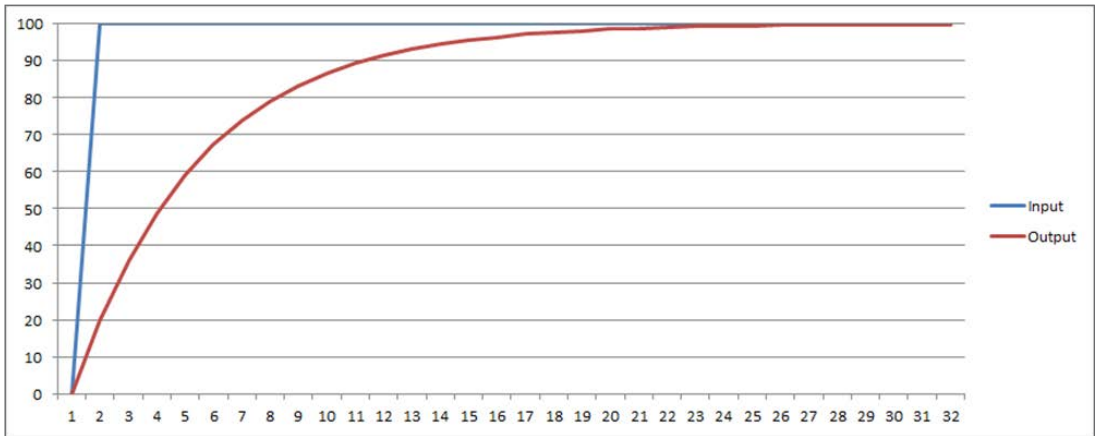
### Filter Example

This method provides flexibility to adjust the sample time and percent change. At each sample time the difference is calculated between the Input and Output, then a percentage of this difference is applied to the new Output. A higher value in DS1 allows more change, or less filtering. A lower value in DS1 allows less change, or more filtering.



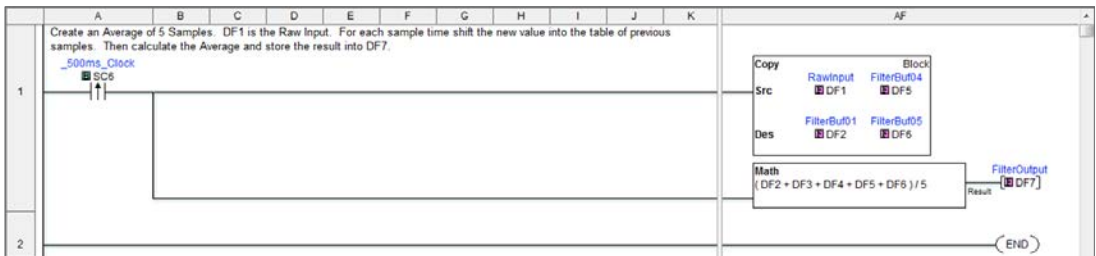
## Common PLC Programming Patterns

In this step response, the Input changes from 0 to 100.

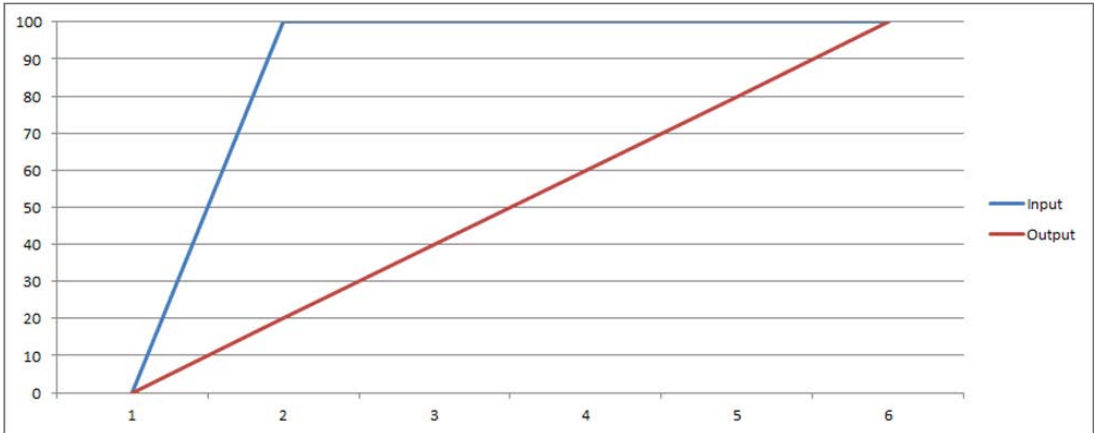


### Average Example

This method enables the flexibility to adjust the sample time and number of samples. At each sample time a new Raw Input is shifted into the group of 5 samples. A new Average is then calculated. In this example, it takes 5 sample times for the Input to reach the Output.



In this step response, the Input changes from 0 to 100.



## State Machine Programming

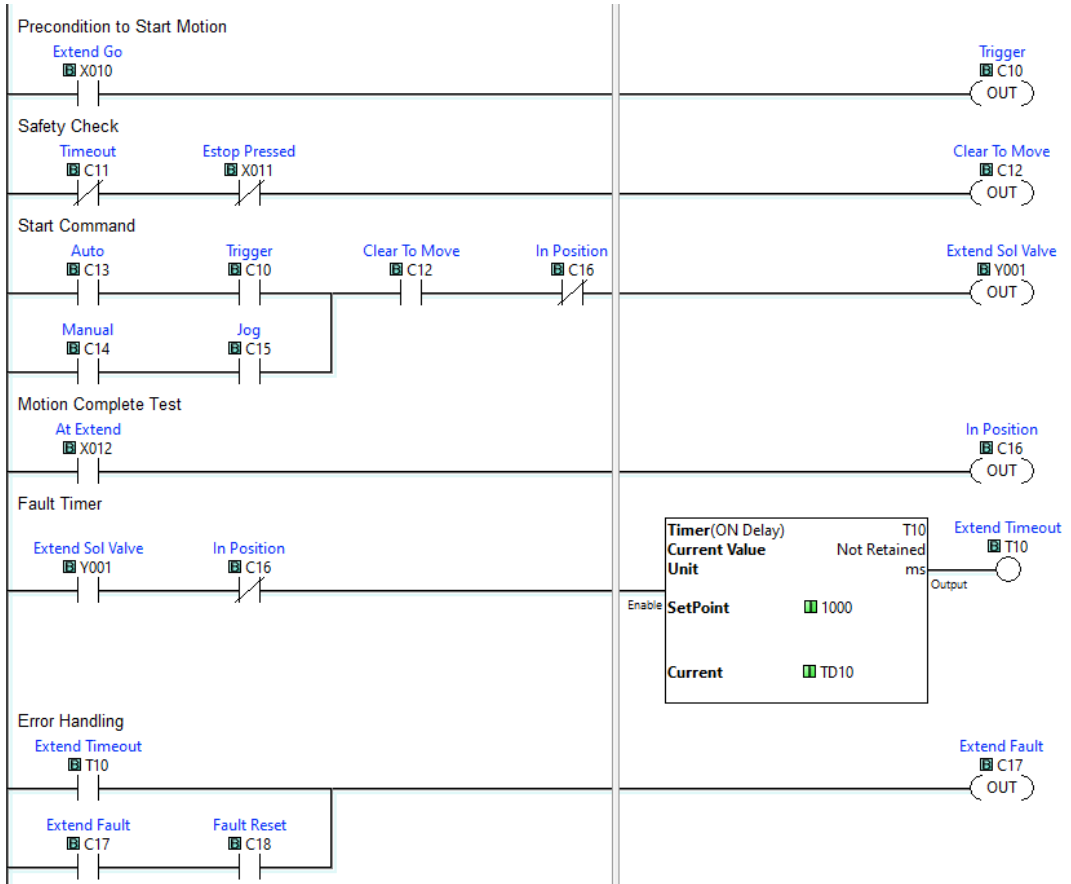
Sometimes it is helpful to control a sequence of actions where you have output events that you want to happen and gates that control when those events are complete and its time to move to the next step.

## Motion Control

This pattern requires five coils and is sometimes called “Five Rung”. It is used whenever an air cylinder or motor will initiate motion in order to ensure safe and complete execution with proper error handling.

1. Trigger to start the move event
2. Safety check
3. Move Command
4. Move Complete Check
5. Fault test

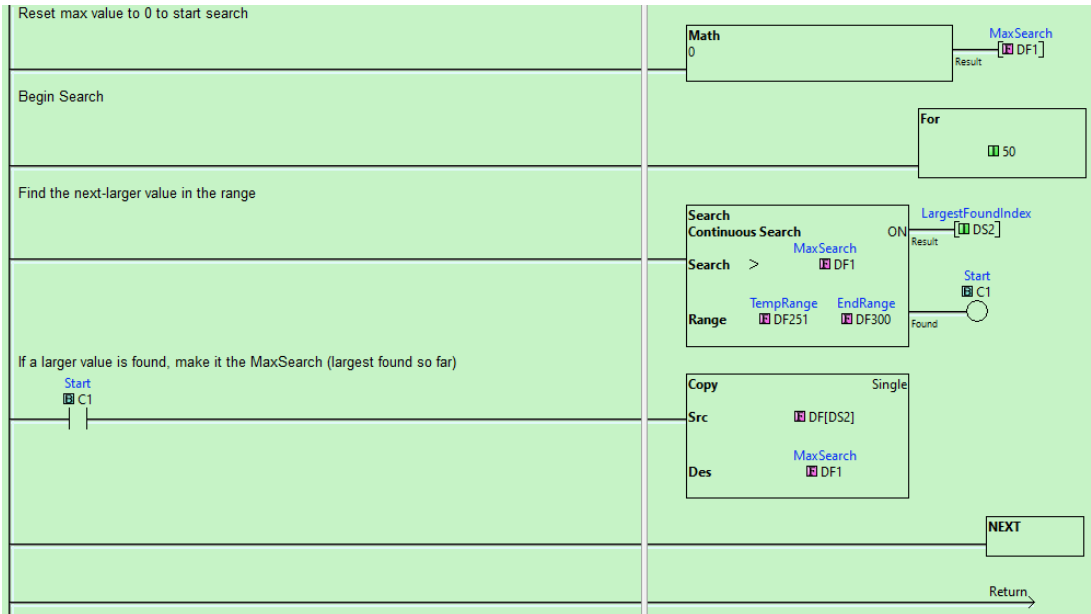
# Common PLC Programming Patterns



## Advanced Math Logic

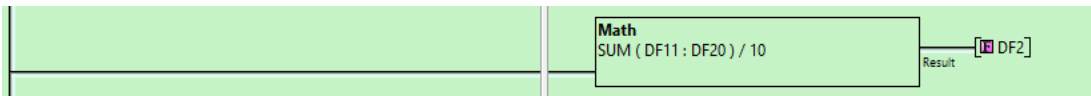
### Min/Max

Find a minimum or maximum value from a range of numbers. This example searches a range of 50 values stored in registers DF251-DF300. Each iteration of the loop will look for the next largest number until it can't find any larger number.



### Average

The Math “SUM” command provides an easy start to calculating the average value of a list of values. This capability translates into methods for filtering (see below).



## Node-RED Module Compatibility

---

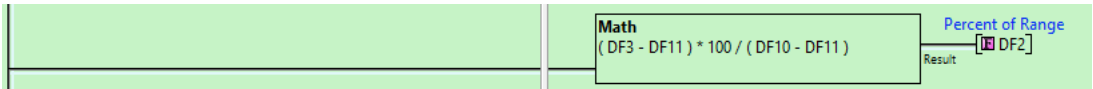
### Convert to Percent

This is useful if you want to graph data in C-more with more than one Y-Axis or with Dynamic Y-Axis ranges.

DF3 is the measured value.

DF10 is the Range max value.

DF11 is the Range minimum value.



To use Temperature as an example, if measured is 72°F, Max is 212 and min is 32, the percent of range is 22.2% (Coincidentally, that is also the conversion from Fahrenheit to Celsius). This makes it easy to create a C-more chart with dual Y-axes by setting up a chart with a range from 0–100% and then creating a range of values on the right starting at 32 and going up to 212.

## Node-RED Module Compatibility

Use the link below to identify and generate a link to the latest compatible version of any Node-RED module:

<https://automationdirect.github.io/CLICK-PLC/Node-RED/C2-NREDModuleVersionCheck/CompatililtyCheck.html>

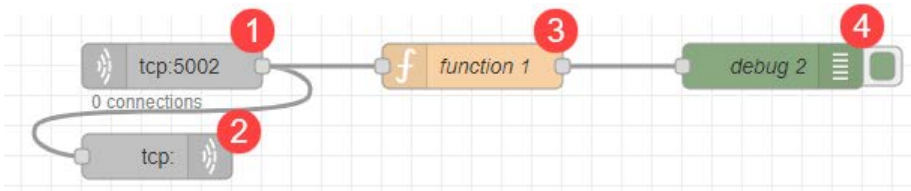
This utility lets you enter a Node-RED module name and it will generate a download link for the best version compatible with the C2-NRED.

## Node-RED Project Examples

### PROJECT: Initiate a Node-RED flow from a CLICK event

#### Project Summary:

This script and corresponding ladder rung initiates a flow in Node-RED and sends a value from the CLICK to that flow. That value will be 0 or 1 for a boolean value, or a signed int between -32767 and 32767 for an integer.

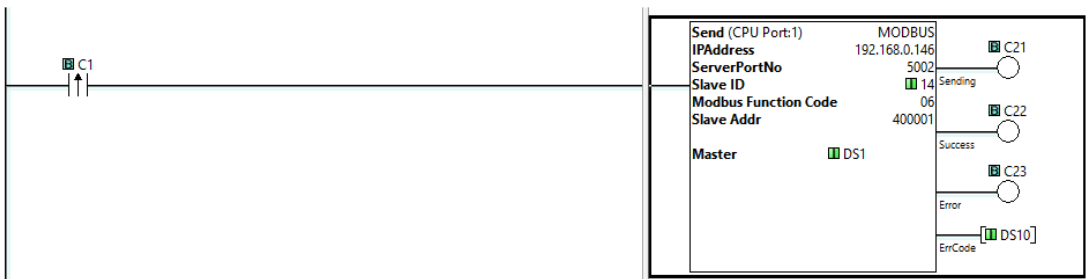


**NOTE:** The CLICK CPU and the C2-NRED module must be connected to the same Ethernet network in order to communicate via Modbus TCP.

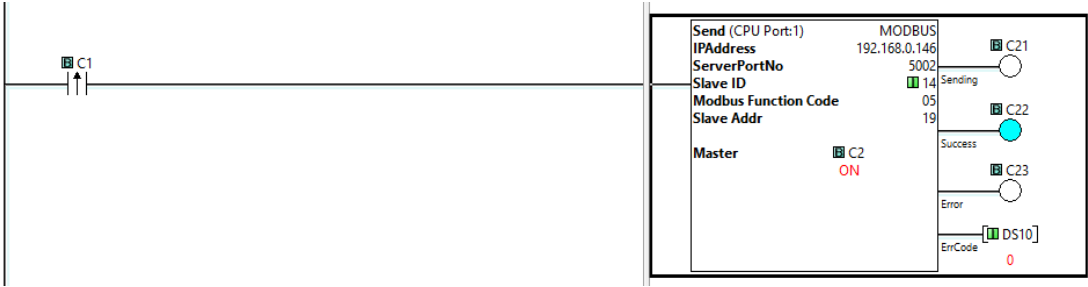


**NOTE:** Node 3 is optional and only necessary if you want to pass data from the CLICK CPU to Node-RED as part of this function. If the CPU is only trying to initiate a flow, only nodes 1 and 2 are needed.

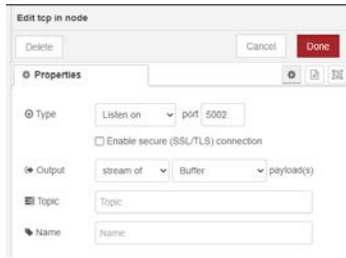
**Send an integer** (DS register). Configure the Modbus send as function 05 (Send single Register) using the same port number defined in the Node-RED TCP In node. The Slave ID and slave address are ignored. The data value sent will be the “Starting Master Address”



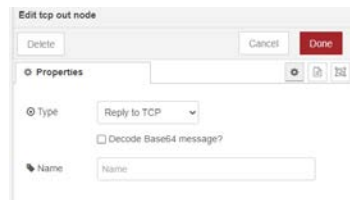
**Send a bit** (X, Y, C Register) Configure the Modbus send as function 05 (Send single Register) using the same port number defined in the Node-RED TCP In node. The Slave ID and slave address are ignored. The data value sent will be the “Starting Master Address”



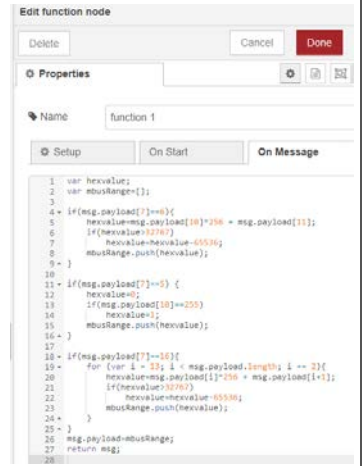
1: TCP IN—Set a port other than the default modbus port. This must be greater than 1024.



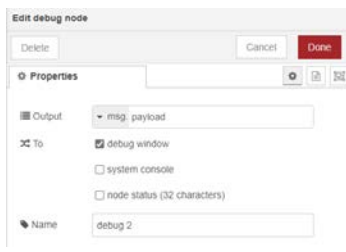
2: TCP OUT—This node will only Echo the TCP message back to the CLICK. This is default the Modbus expectation.



3: Function—Convert the Hex data sent in the Modbus packet to an array of bits or integers.



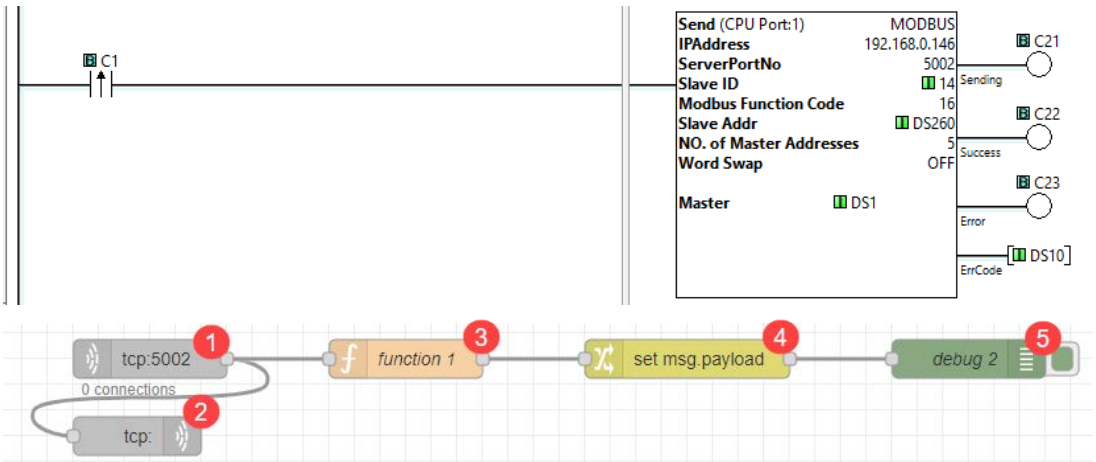
4: Display results.



## Importable Project:

See our repository at <https://github.com/AutomationDirect/CLICK-PLC/tree/main/Node-RED> for an importable copy of this project.

## PROJECT: Find the Max Value

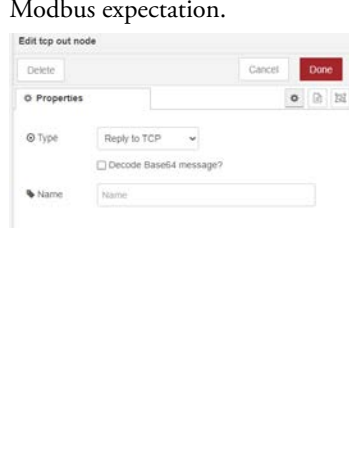


You can pass in a range of numbers using the Modbus Function Code 16 and the following function to convert those values to an array of signed integers. Then use a single node to find the max value in that array. Compare the one rung in this solution to the solution in ladder shown above.

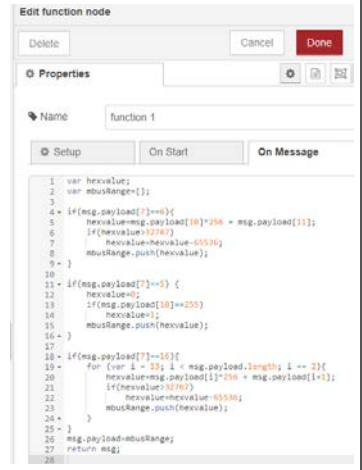
1: TCP IN—Set a port other than the default modbus port. This must be greater than 1024.



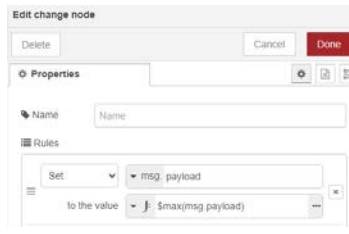
2: TCP OUT—This node will only Echo the TCP message back to the CLICK. This is default the Modbus expectation.



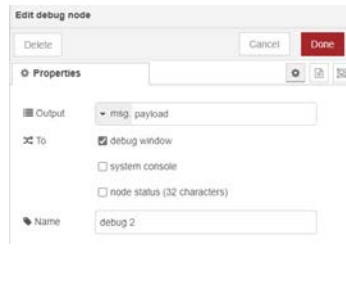
3: Function—Convert the Hex data sent in the Modbus packet to an array of bits or integers.



4: Use JSONata to find the max value in the array and return an integer.



5: Display results.



### Importable Project:

See our repository at <https://github.com/AutomationDirect/CLICK-PLC/tree/main/Node-RED> for an importable copy of this project.

### PROJECT: Find your IP Address

node-red-contrib-ip

Some Services Like AzureSQL require you to setup your C2-NRED's IP Address to be able to connect to their servers as part of a robust security authentication process. This node will return your public IP address.

Download the IP node here:

<https://registry.npmjs.org/node-red-contrib-ip/-/node-red-contrib-ip-1.0.1.tgz>

## PROJECT: Connect to a Microsoft SQL Azure Database

This should only be attempted by (or with) an experienced Database Administrator. Automation Direct will not support database configuration. This guide is a minimal explanation of the basic steps involved. Every database configuration is different and tailored to the security requirements of each individual organization.

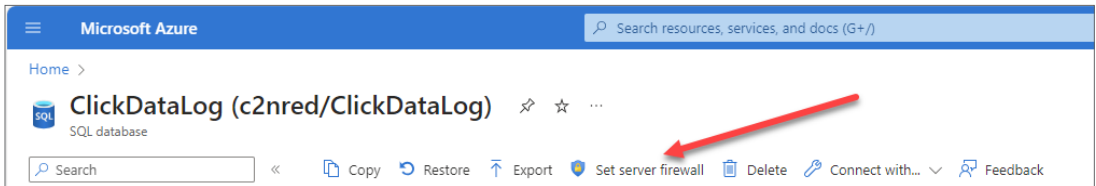
The MSSQL Node found here has been tested to work with Azure:

<https://registry.npmjs.org/node-red-contrib-mssql-plus-box/-/node-red-contrib-mssql-plus-box-0.1.4.tgz>

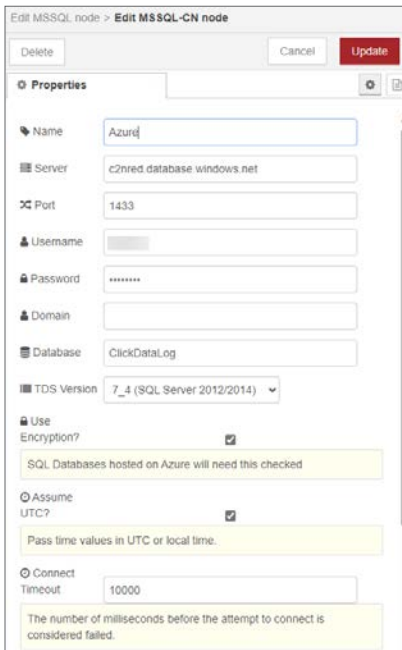
Create a Microsoft Azure Account.

Create an Azure Database – Enable SQL Authentication (ADO.NET).

From the Azure SQL Dashboard, near the top middle of the screen, select “Set Firewall Rule”.



Create a Firewall rule in Azure to allow your Node-RED unit to connect to Azure by entering the IP Address of the C2-NRED module. See Instructions to find your IP in the previous example.



Create test data. The following SQL will create a table named “Persons” in a database named NodeRed. It

will insert 4 records into that table that you can use for testing.

```
USE [NodeRed]
```

```
GO;
```

```
DROP TABLE If EXISTS dbo.Persons
```

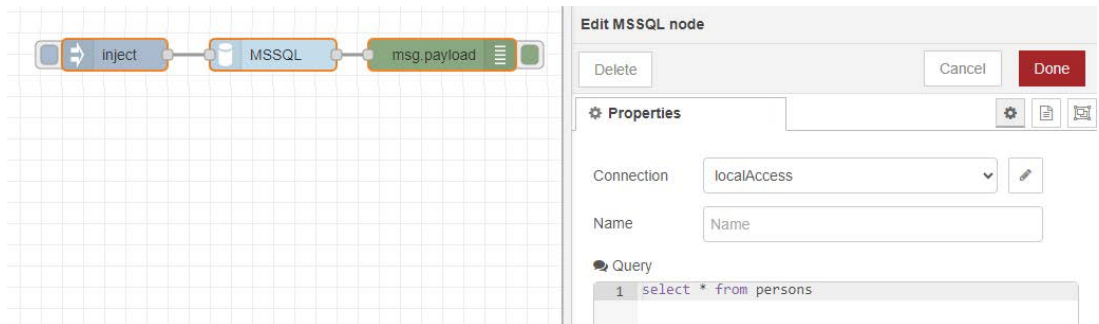
```
CREATE TABLE Persons (  
    Personid int IDENTITY(1,1) PRIMARY KEY,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int  
);
```

```
Insert into dbo.Persons  
(LastName, FirstName, Age)  
Values ('♥Smith', 'Andy', 25),  
( 'Carson', 'Kelly', 18),  
( 'Nells', 'Jeff', 47),  
( 'Peters', 'Bill', 54);
```

### SQL Select

1. The following Select statement will retrieve all the rows in the Persons table

```
Select * from persons
```



The screenshot shows a Node-RED workflow on a grid. It consists of three nodes: an 'inject' node, an 'MSSQL' node, and a 'msg.payload' node, all connected in a sequence. The 'MSSQL' node is selected, and the 'Edit MSSQL node' dialog is open. The dialog has a 'Delete' button, 'Cancel', and 'Done' buttons. Under the 'Properties' tab, the 'Connection' is set to 'localAccess', and the 'Name' field is empty. The 'Query' field contains the SQL statement: '1 select \* from persons'.

2. That will return a JSON object:

```
[{
  "Personid":1,
  "LastName":"Smith",
  "FirstName":"Andy",
  "Age":25
},
{
  "Personid":2,
  "LastName":"Carson",
  "FirstName":"Kelly",
  "Age":18
},
{
  "Personid":3,
  "LastName":"Nells",
  "FirstName":"Jeff",
  "Age":47
},
{
  "Personid":4,
  "LastName":"Thompson",
  "FirstName":"Bill",
  "Age":54
}]"
```

3. Notice, it is an array of objects, so there is one object per array element returned in the query. Each object has a Key:value pair. This enables Node-RED to address these using JSONata or JavaScript using dot notation. For example, `msg.payload[1].LastName` will return "Carson".

## SQL Update

1. The following update will change the First name of personid=4 to Larry:

```
Update NodeRed.dbo.Persons
set FirstName='Larry'
Where Personid=4;
```

2. The return value (`msg.payload`) will be null.

### SQL Insert

1. The following insert will add a new person to the Persons table:

```
Insert into dbo.Persons
(LastName, FirstName, Age)
Values ('Brooks', 'Paul', 25)
```

2. The return value (msg.payload) will be null.

### SQL Delete

1. The following SQL will delete the person with Last Name='Nells'.

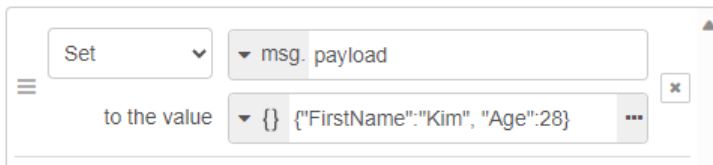
```
Delete from persons
where LastName='Nells'
```

2. The return value (msg.payload) will be null.

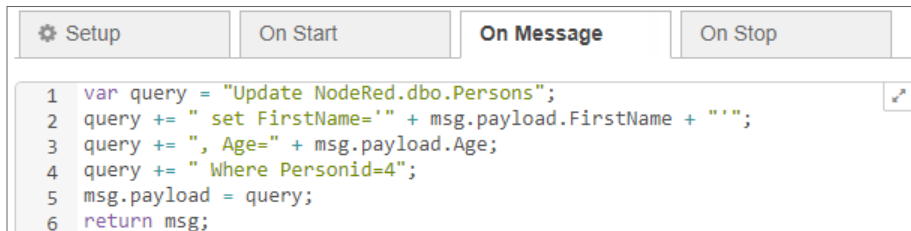
You may notice these are all static queries. There is no variable information in the query, so they are great for testing, but don't provide much practical value. In order to add variables, we need to add a couple of nodes to our flow:



In this flow, the set msg.payload creates a JSON Object with 2 key value pairs representing our variables



The function node creates a query string using those variables.



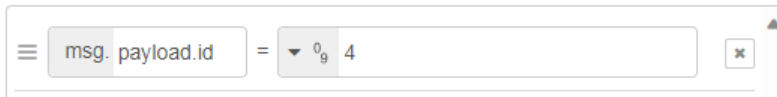
This is JavaScript and is adding together String values including the variables passed in from the prior Set node.

Finally, the MSSQL node is blank. It has a rule that if you don't give it anything to run, it will run whatever you pass in through the msg.payload value:

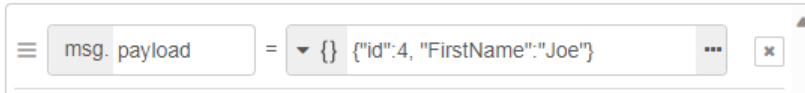


In this example, the Function node enables you to create whatever query you want with all the variables you can pass it. For instance, if you have a barcode reader connected to your CLICK PLUS PLC, you could have a CLICK Read node, read the barcode value and then create a query to look up the barcode and return a Bin number and quantity so your automated pallet system can retrieve the proper bin and number of parts.

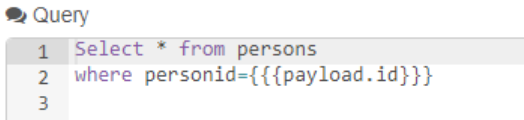
An alternate approach which doesn't use JavaScript is to pass a value into the query. This can be an object if you have multiple values:



or



Then in the query you can use the “Mustache” format to extract the object's values and inject them into the query:



This example in conjunction with the “Initiate a flow from a CLICK” and the “Find your IP” provides everything needed for a system to “Send” a recipe ID from a CLICK, start a flow which runs a query to retrieve a recipe from SQL. The next step would be to use a CLICK Write node to write the recipe back to the CLICK. These steps enable a system with machines distributed around the country to use a common data set that can be administered from a central point.

### PROJECT: Connect to the local SQLite Database

The C2-NRED comes with a SQLite database preinstalled as part of the firmware. SQLite, as the name suggests, is a lightweight single file database that follows standard SQL query rules. It is important to note there are several pros and cons of using an on-device database instead of connecting to a remote SQL database.

#### Pros and Cons of using a local database:

Pros	Cons
Development speed – there is nothing additional you need to install.	SQLite is a single connection DB – It is not thread safe so only the Node-RED application should read/write to it. No concurrent users or external connections are allowed.
No additional database server so you can save hardware costs, software costs, and electricity costs associated with servers.	There are no tools available to backup or restore the data stored in the C2-NRED memory. It will be the user's responsibility to write any code necessary to generate backup files.
Read/write data regardless of network availability.	The data in the DB is only available to the host Node-RED (without custom code to expose it).
Read/write speeds are not network dependent.	The CLICK Project Loader does not backup user files.
If writing to the SD Card, up to 32gb of storage is available.	Solid state memory has limited read/write cycles and DB can wear them out prematurely.
Possible to remove the SD Card and read/write the DB from a PC. (You must power cycle the PLC to reconnect to a DB on a SD card.)	

#### Frequently Asked Questions:

*Can the DB be written to the SD card and read /written by an instance on the PC?*

YES, but if a DB is on an SD Card, after that card is removed, the CLICK must be restarted to reconnect to the DB on the card.

*Is there a network path to the user directory on the C2-NRed?*

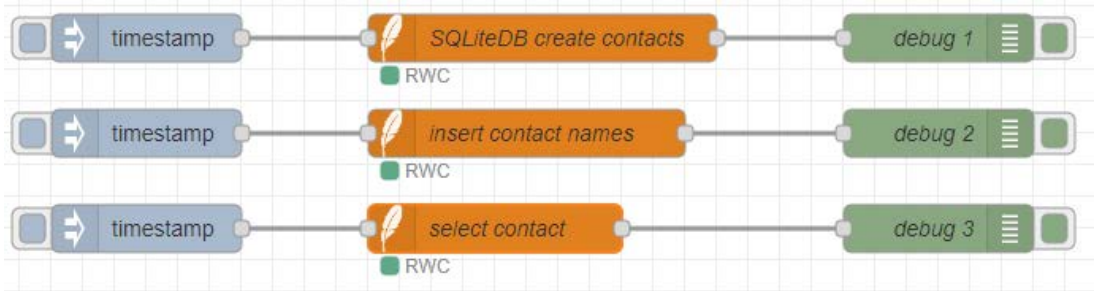
No, this would require software running in Linux like Samba (software that provides access to SMB/CIFS protocols used by Windows). See [How to Share Files Between Windows and Linux \(howtogeek.com\)](http://howtogeek.com) for more information.

*Will files be backed up and restored by the Project Loader tool for CLICK?*

No.

## Minimum configuration to use SQLite

Using the SQLite module includes a minimum of three nodes:



### First, create a database.

There are no extra steps required to install or configure the DB. You simply drop a sqlite node from the storage group onto your canvas. Click the pencil icon to set the DB file path and name. There are only two choices of file path (parent directory):

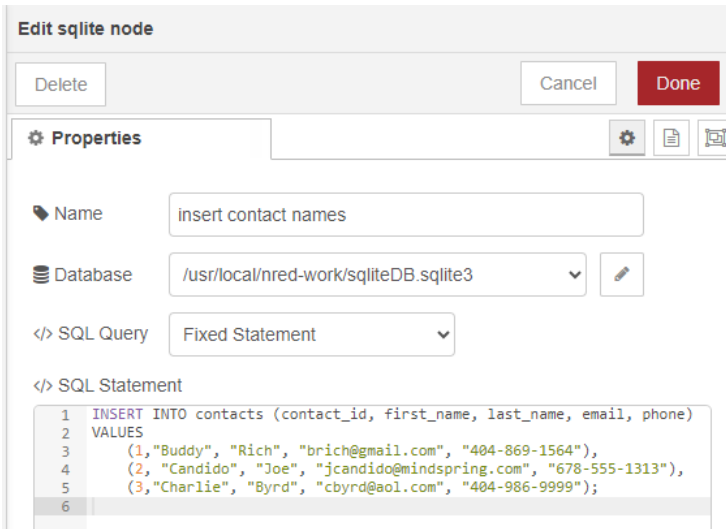
1. Local storage: `/usr/local/nred-work/`
2. SD Card storage: `/run/media/mmcbk0p1/`

The database name and extension can be anything you want, but it is recommended to use `.sqlite3` as the extension to make it easy to identify.

You will also need to set the mode to “Read-Write-Create” to enable the creation of a new DB. Click update to close the configuration window. In the SQL statement panel, you will want to create your table(s).

## Node-RED Project Examples

Use a second node to populate your table:



**Edit sqlite node**

Delete Cancel Done

**Properties**

Name: insert contact names

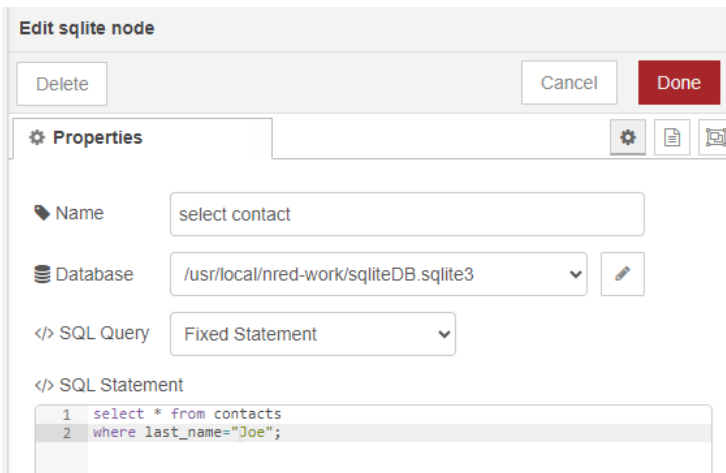
Database: /usr/local/nred-work/sqliteDB.sqlite3

SQL Query: Fixed Statement

SQL Statement

```
1 INSERT INTO contacts (contact_id, first_name, last_name, email, phone)
2 VALUES
3 (1, "Buddy", "Rich", "brich@gmail.com", "404-869-1564"),
4 (2, "Candido", "Joe", "jcandido@mindspring.com", "678-555-1313"),
5 (3, "Charlie", "Byrd", "cbyrd@aol.com", "404-986-9999");
6
```

Use a third node to retrieve data from the table you just created and populated:



**Edit sqlite node**

Delete Cancel Done

**Properties**

Name: select contact

Database: /usr/local/nred-work/sqliteDB.sqlite3

SQL Query: Fixed Statement

SQL Statement

```
1 select * from contacts
2 where last_name="Joe";
```

It is possible to create the Database file using a SQLite editor on your PC (e.g <https://sqlitebrowser.org/>). Write the DBfile to the correct path on the SD card, and insert it into the SD card slot on the Node-RED. However, you must cycle power to your PLC to get it to connect to the SD card slot and read the data.

Additional information can be found at:

<https://www.sqlitetutorial.net/> or [SQLite tutorial - W3schools](#)

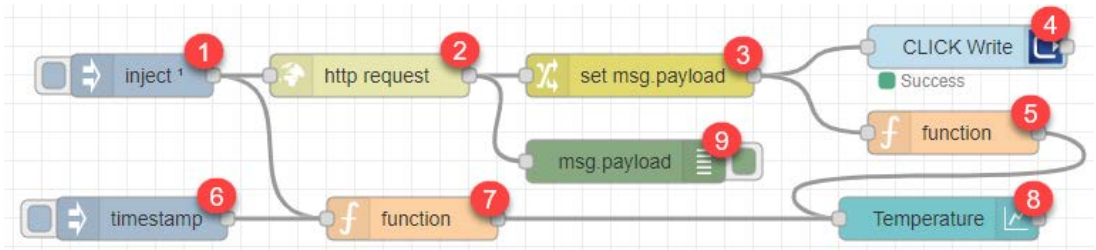
## PROJECT: Call a Web Service to retrieve data and chart the results

### Project Summary:

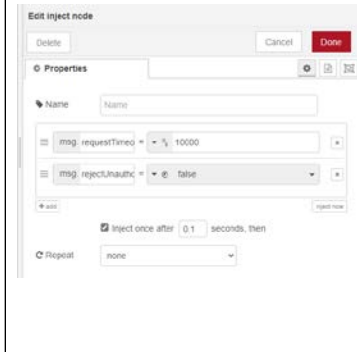
This script initiates an HTTP request to a weather API. When it receives the response as JSON, it uses JSONata to strip out most of the data and is left with Time and Temperature for the next 12 hours. It writes those values to the CLICK and sends them to a function where the data is formatted as a line graph. In parallel, there is a second thread that calls a function to zero out the line graph.

3rd Party Modules required: None

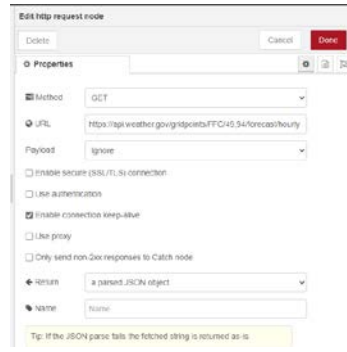
Skill Level: Beginner



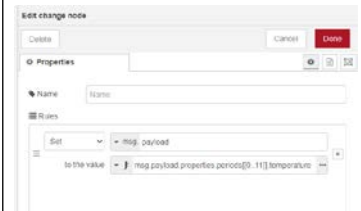
#### 1: Inject.



#### 2: HTTP Request.



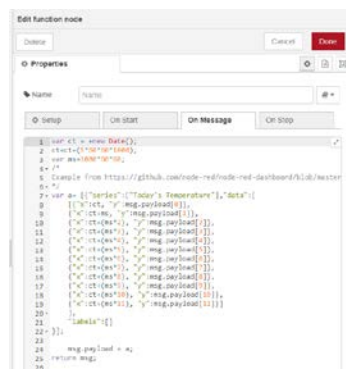
#### 3: Set Payload – Use JSONata to strip everything but time and temp from the web service response.



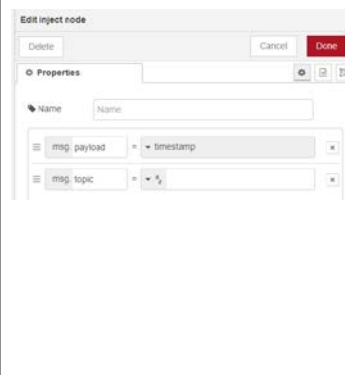
## 4: Write Temperature to the CLICK.



## 5: Script.



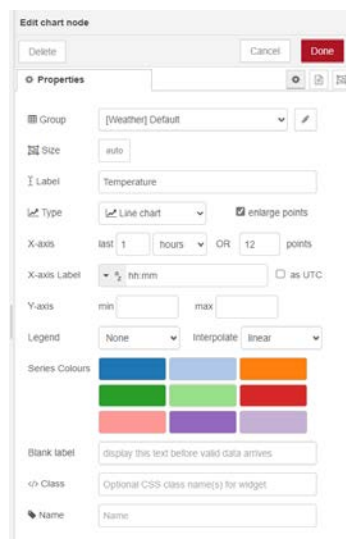
## 6: Inject.



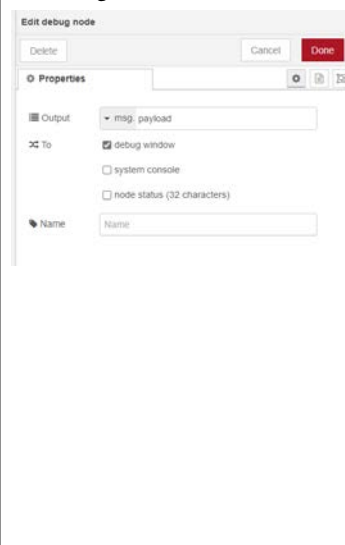
## 7: Script.



## 8: Chart.



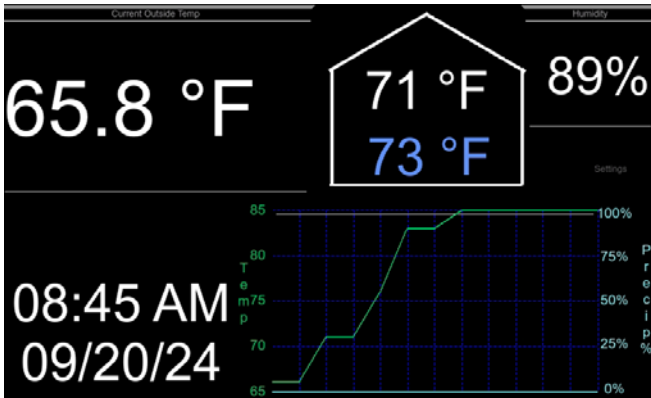
## 9: Debug.



## Importable Project

See our repository at <https://github.com/AutomationDirect/CLICK-PLC/tree/main/Node-RED> for an importable copy of this project.

## PROJECT: Local Weather Station



### Technologies Implemented:



Shelly BLU H&amp;T



Shelly 1 Mini Gen3



Google Nest

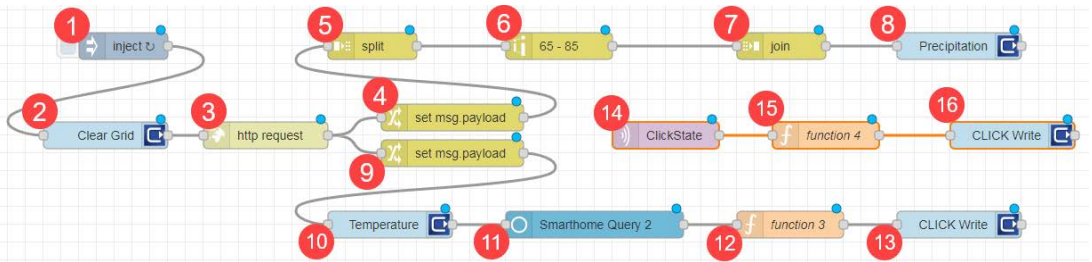


Amazon Echo Dot

- Shelly Bluetooth-Temp and Humidity sensor
- Shelly Bluetooth-Wi-Fi Gateway
- MQTT over Wi-Fi
- Node-RED Hosting MQTT Broker
- Node-RED MQTT Client
- Node-RED HTTP Request to [api.weather.gov](https://api.weather.gov) to retrieve forecast
- Node-RED Alexa Nodes to integrate with Alexa, and Nest Thermostats
- JSONata to parse the JSON response from [API.WEATHER.GOV](https://api.weather.gov)
- CLICK PLC to process the data and communicate with CM5 HMI
- CM5 HMI for data display
- CM5 NTP Server to get time from the cloud

## Node-RED Project Examples

- CM5 Webserver to make the data available - securely – from anywhere.



1. Inject – start the Flow and rerun it every 10 minutes
2. Send a bit to CLICK To start charting incoming data
3. Http request – call <https://api.weather.gov> to get the 7 day forecast in JSON format
4. Use JSONata to parse the api response into an array of precip % values
5. Split the array into individual values and stream them out
6. Scale each precip value from 0-100 to 65-85 for charting
7. Rejoin the scaled data into an array
8. Sent the array of precip data to CLICK as a range of integers
9. Use JSONata to parse the api response into an array of temperature values
10. Send the temperatures directly to CLICK
11. Use ALEXA to query my NEST Thermostat. Google’s Security requires a cloud-based server to access devices. This approach uses Amazon to access the device, then queries Amazon for the data.
12. Use a JavaScript function to convert Amazon’s Object based response into an array.
13. Send the inside temp and the Nest setpoint to CLICK
14. The Shelly sensor is connected to the Shelly 1 gateway – the Gateway broadcasts temp, humidity, and sensor battery state every minute or when the data changes using MQTT. The “ClickState” node receives the MQTT message
15. This function parses the temp, converts it to Fahrenheit, and sends all three values as an array
16. Writes the temp, humidity and battery to the CLICK

Upon The CLICK receiving all the data, it starts a pulse signal for 13 pulses. Each rise of the pulse copies the next data value into a register that the Cmore will plot. Each fall of the pulse triggers C-more to plot the value.

Weather data is available via FREE API at: <https://api.weather.gov/>

Documentation is available at: <https://www.weather.gov/documentation/services-web-api/>

The forecast info in this example can be found by running the Get /point/point API in the Swagger tool.

1. Open Google maps and find the location you are testing.
2. Right click to see the latitude and longitude of that location

3. Single click on those values to copy them to your clipboard
4. Paste those values into the String field for the point API
5. Remove the space after the comma
6. Execute

Look below in the response body field for the following values:

1. gridId
2. gridX
3. gridY
4. URLs for Forecast data, weatherstation information and more

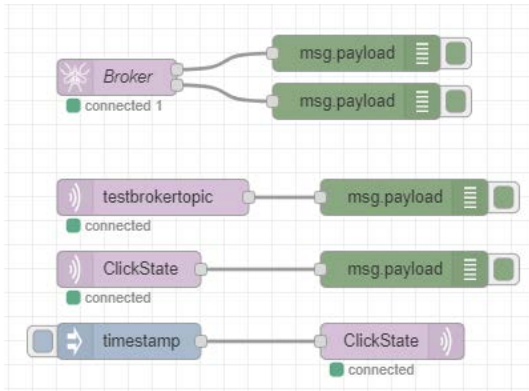
### PROJECT: Create an MQTT Broker

Use the module `node-red-contrib-aedes` (MQTT Broker) (0.11.1) available at the following link:

<https://registry.npmjs.org/node-red-contrib-aedes/-/node-red-contrib-aedes-0.11.1.tgz>

This flow allows the C2-NRED module to act as an MQTT Broker, Subscriber and Publisher simultaneously. Even if you don't need MQTT, this flow enables the CLICK PLUS to trigger events in Node-RED. For example, if you have a sensor on an overflow gate, that sensor can trigger an MQTT message which will in turn trigger a Node-RED flow which can send a text message or publish the event to a web server.

For troubleshooting MQTT connections, an external client like [MQTTX: Your All-in-one MQTT Client Toolbox](#) is recommended.

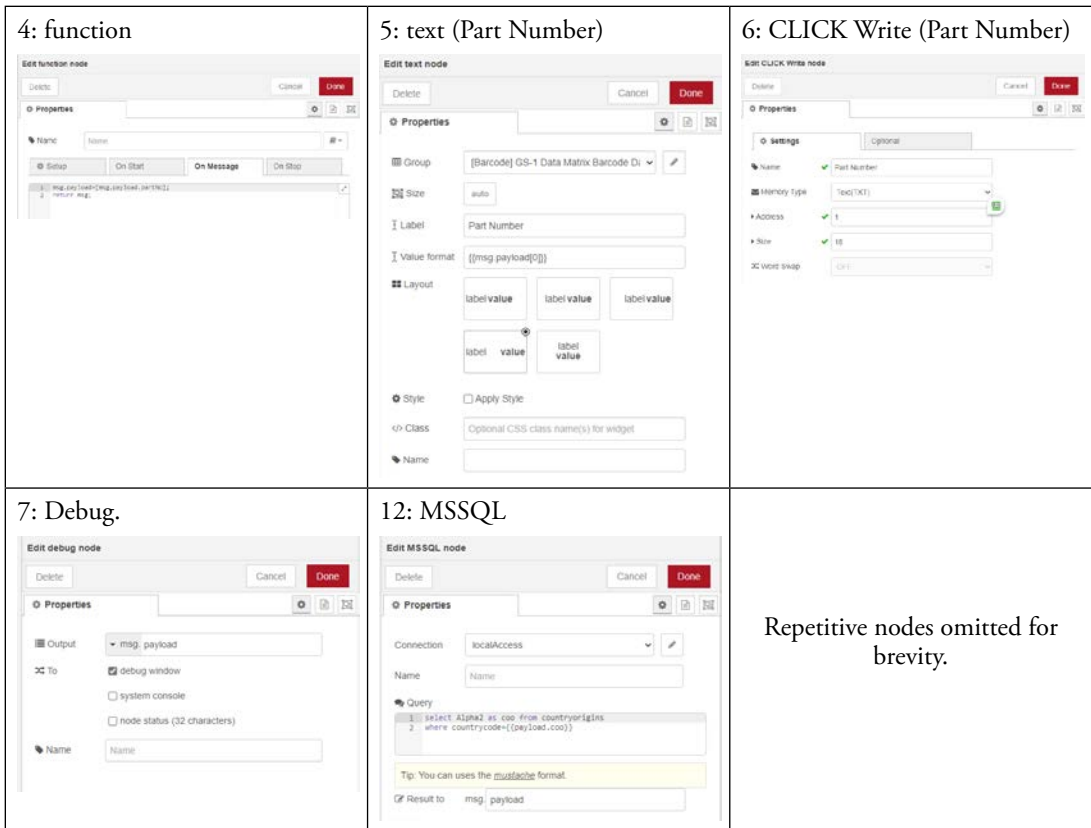


The AEDES Broker node defines the broker configuration. The gray MQTT Broker defines the connection for the publisher and subscriber nodes (`mqtt in` and `mqtt out`). The green debug nodes would be replaced with process nodes to handle the output from the MQTT subscribe nodes. The 2 debug nodes connected to the AEDES Broker are entirely optional to view events occurring within the broker. In this example, the C2-NRED's IP address is 192.168.0.115 and the Broker is monitoring port 1883.

### Importable Project

See our repository at <https://github.com/AutomationDirect/CLICK-PLC/tree/main/Node-RED> for an importable copy of this project.





1. This creates a web page at `Http://[node red IP Address]/GS1code?data`  
Calling the URL (for example `http://192.168.0.5:1880/gs1code?barcodedata`) will initiate the flow and put the barcode data in the `msg.payload`.
2. This JavaScript function parses the barcode into an object containing its 10 parameters.
3. Respond to the HTTP request with a 200.
4. This JavaScript function pulls out the part number object so that it can be written to the CLICK.
5. This dashboard node formats the Part number, adds a label and displays the part number on the dashboard. Nodes 9-11, 13, 15, and 17-20 do the same.
6. Write the part number to CLICK.
7. A debug statement to help troubleshoot.
12. This MSSQL statement takes a country code from the barcode and runs a query against a remote SQL database to look up the country of origin.

## Importable Project:

See our repository at <https://github.com/AutomationDirect/CLICK-PLC/tree/main/Node-RED> for an importable copy of this project.

## BONUS FUNCTION: Process Modbus - Four message types using JavaScript to create array of values:

```

var hexvalue;
var mbusRange=[];
var i;
//Single Register to signed INT
if(msg.payload[7]==6){           //Message type 06
    hexvalue=msg.payload[10]*256 + msg.payload[11];
    if(hexvalue>32767)
        hexvalue=hexvalue-65536; //convert unsigned int to signed int
    mbusRange.push(hexvalue);
}
//Single Coil to array of binary number
if(msg.payload[7]==5) {           //Message type 05
    hexvalue=0;                   //default to 0
    if(msg.payload[10]==255)      //if bit is high, set to 1
        hexvalue=1;
    mbusRange.push(hexvalue);
}
//Multiple Coils to array of binary numbers
if(msg.payload[7]==15){           //Message type 15
    for (i = 13; i < msg.payload.length; i += 1){
        hexvalue=msg.payload[i].toString(2).padStart(8, '0');
        mbusRange=hexvalue.split("").concat(mbusRange);
        mbusRange=mbusRange.map(Number);
    }
}
//Multiple registers to array of signed INT
if(msg.payload[7]==16){           //Message type 16
    for (i = 13; i < msg.payload.length; i += 2){
        hexvalue=msg.payload[i]*256 + msg.payload[i+1];
        if(hexvalue>32767)
            hexvalue=hexvalue-65536;
        mbusRange.push(hexvalue);
    }
}
msg.payload=mbusRange;
return msg;

```

# C2-OPCUA Module Connection to UAExpert

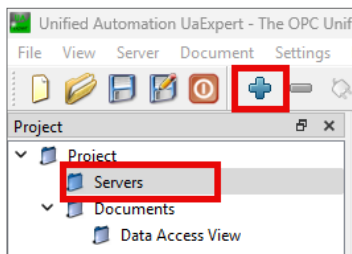
This will walk you through setting up your C2-OPCUA Intelligent Module and ensuring an OPC UA Client can communicate with the module. You will need the following for this set-up:

- C2-xxCPUx (Any CLICK Plus CPU) with Firmware v3.7x or higher
- C2-OPCUA Slot Module
- CLICK Programming Software v3.7x or higher
- UAExpert by Unified Automation installed on PC

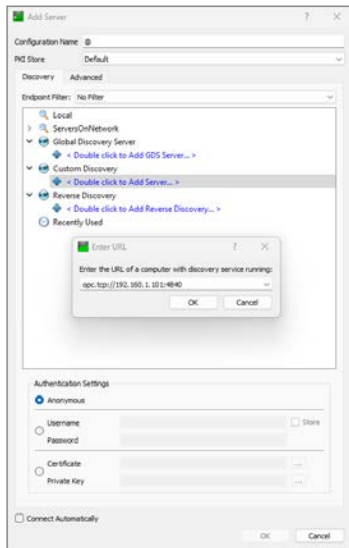
## Setup

Please follow the installation instructions in [CLICK PLUS PLC Hardware User Manual Ch. 5](#). Then follow the guide in [CLICK Help Topic CL330 C2-OPCUA Module Configuration](#). The following steps will pick up where both of those steps end:

1. It is vital to ensure the PLC clock matches the OPCUA clock to ensure data can be synchronized. *(If times stamps are not within a few minutes of each other the C2-OPCUA module will not function correctly.)* The best way to achieve this is to use the “Calendar/Clock Setup” screen under the PLC menu. Change the radio button to “Adjust to PC Clock” and press “Write to PLC”.
2. On the “C2-OPCUA Module Configuration” screen in the General tab, copy your Endpoint URL using the “Copy to Clipboard” button and make a note of Server Name.
3. On the same screen change to the Node List tab and add SD26, DS1, and C1. Also add these to a Data View in the CLICK Programming Software for testing later.
4. Transfer your project to the CLICK. Wait for the C2-OPCUA “OK” light to stop flashing for setup to complete. If the “ERR” light is on, check the CLICK Programming Software PLC Error History screen on the PLC Tab and select the Slot tab for where the C2-OPCUA is installed.
5. In UA Expert, either click the blue “+” in the tool bar or right click the Servers area in the left-hand project window.



6. Then paste the Endpoint URL that was copied in step 2 and press OK.

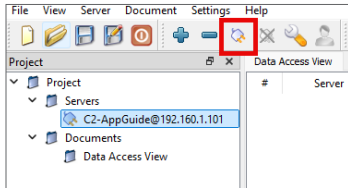


7. Now the server will show in the list under Custom Discovery. Highlight the correct entry and add the Server Name also noted in step 2 in the Configuration Name.
8. Change to the Advanced tab while the server entry is still highlighted, paste the Endpoint URL in the Endpoint Url field and press OK.

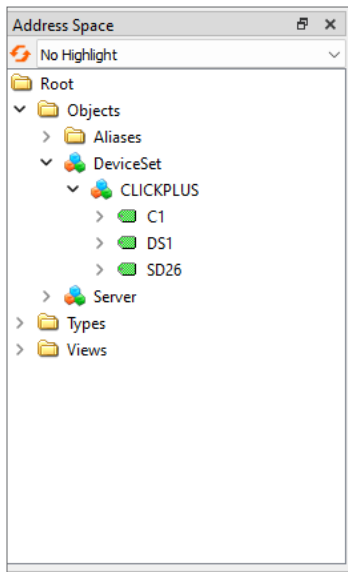


## C2-OPCUA Module Connection to UAExpert

- Now the Server will show in the left-hand Project window. Highlight the server name you just added and press the “Connect” button that looks like a plug.



- If using a self-signed certificate, you may get a pop-up warning about the certificate. Press “Trust Server Certificate” then Continue button. *(A self-signed certificate with security disabled is good for troubleshooting and set-up, but ADC recommends using security when in operation as all data is sent as plain text.)*
- In the lower left hand Address Space window, expand Objects, then DeviceSet, and finally CLICKPLUS. Now you can see the nodes you added in step 3. These can now be dragged to the middle Data Access View window.



- Statuscode should be “Good” and the Value for SD26(\_RTC\_Second) should display the current second. This tests if your C2-OPCUA module is writing data to UAExpert.

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	C2-AppGuide@192.160.1.101	NS1 Numeric 2002	C1	false	Boolean	11:16:19.043 AM	11:55:08.523 AM	Good
2	C2-AppGuide@192.160.1.101	NS1 Numeric 2001	DS1	0	Int16	11:55:36.441 AM	11:55:36.574 AM	Good
3	C2-AppGuide@192.160.1.101	NS1 Numeric 2000	SD26	6	Int16	11:56:06.130 AM	11:56:06.177 AM	Good

13. Now double click on the Value cell for DS1 and write 100 and press Enter. This should write data to DS1 in your CLICK PLC. Check your Data View you created in step 3 to ensure the C2-OPCUA module is reading from UAExpert as well.
14. Final test is to double click in the Value cell for C1 and either turn on or off the check box to test Boolean as well.