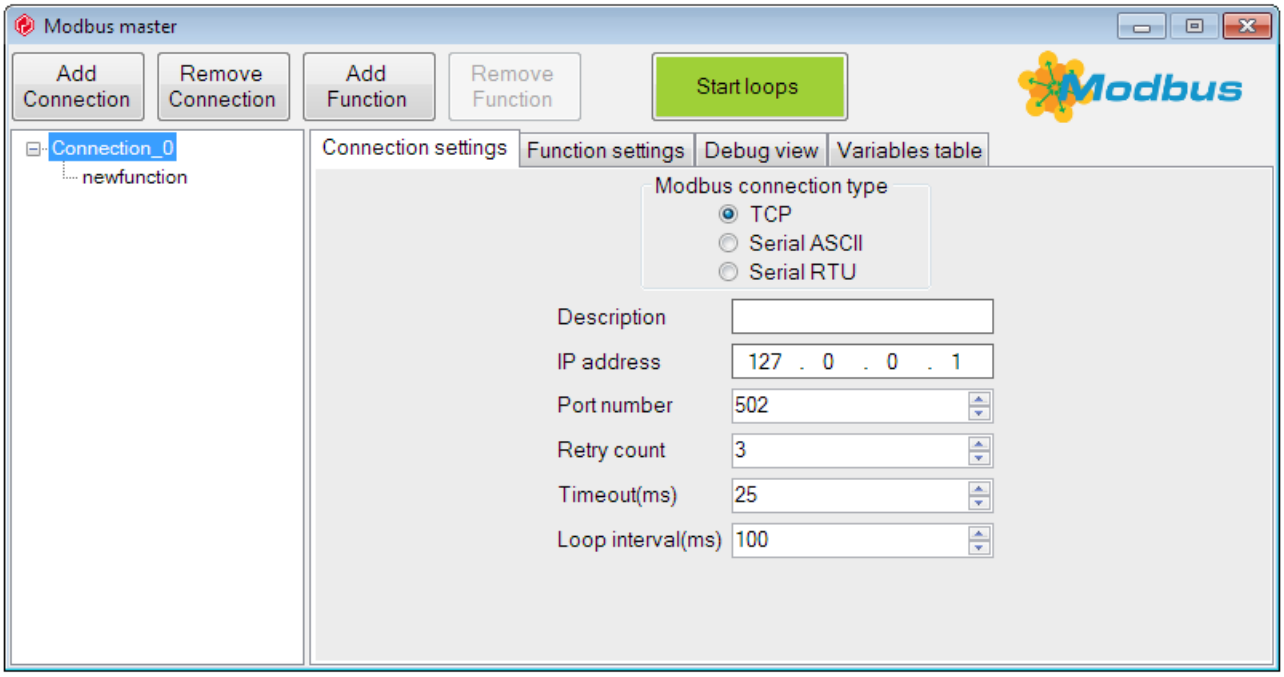
Modbus master plugin user's guide
a module of the UCCNC software



Version of this software manual: 1.0004
Software version: 1.2113

Contents

1.Description of the Modbus maser software plugin module, introduction.
2.The connections.
    2.1.TCP connection.
    2.2.Serial ASCII and RTU connection.
3.The functions.
    3.1.Function types.
    3.2.Function parameters.
4.Running the communication loop.
5.The debug view.
6.The variables table.
7.Accessing the Modbus variables table from within the UCCNC.
8.Example macro codes.

1 .Description of the Modbus maser software plugin module, introduction.

For first of all we thank you for your interest in our software product and reading this user's guide.

The Modbus master plugin is part of the UCCNC software and installs with the software.
To enable the plugin run the UCCNC software and go to the Configuration/General settings page and press the Configure plugins button. On the popup window enable the Modbus master plugin. To make the plugin window to show up on every software startup check the Call startup checkbox.
To run the plugin close and restart the UCCNC software after enabling it in the plugins configuration.

The Modbus master plugin is a plugin to connect the UCCNC software with external devices.
The plugin is a modbus master device and can communicate with modbus slave devices.

For the modbus communication to work the modbus slave device has to be connected to the communication channel where the Modbus master plugin talks to and the communication channel properties have to match in the master and the slave and the slave device has to talk the Modbus language and has to listen to the Modbus master's commands.
For more details about the modbus protocol and devices which support the modbus protocol please check the http://www.modbus.org/ website.

The Modbus master plugin supports 3 type of communication, Modbus TCP (Ethernet), Modbus RTU (serial), Modbus ASCII (serial).

The high level objects which can be created with the plugin are the Connections.
A connection represents a communication channel which can be TCP or RTU or ASCII.
The number of connections which can be created is unlimited. Any number and any of the 3 type connections can be created in any combinations.

The lower level objects are the Functions. The Functions can be attached to the Connections.
Every connection can have an unlimited number of functions.
The Functions represents a data writing or reading operation to and from a Modbus slave device.
There are 8 types of Functions, 4 data read and 4 data write operations. These will be described in details later in this manual.

2 .The Connections

The first thing to establish comunication with a modbus slave is to create a connection in the modbus master plugin. To achive this press the Add connection button on the plugin page.
This will create a new connection and that will be visualized in the Tree view on the left side of the page.
To remove any connections select the connection with clicking it in the Tree view and press the Remove connection button.
To edit and setup the properties of any connections just select the connection with clicking on it's name in the Tree view. Selecting the connection will highlight the name of the connection.
Now with the connection selected click the Connection settings tab page to see and edit the connection parameters.

The connection parameters are the following:

- Connection mode (TCP or RTU or ASCII).

Checking one of the checkboxes will change the communication channel to the selected mode.

- Description.

The description string does not influence the working of the connection, it is just a text note about the connection.

The rest settings are all connection specific. Different settings appear for the TCP and for the RTU and ASCII connection modes and these will be described in the next chapter.

2.1 .TCP connection.

If the TCP connection mode was selected then the Connection setup will show the TCP connection specific parameters which are the following:

- IP address

This is the IP address where the plugin will talk to through this connection.

- Port number

This is the IP port where the plugin will talk to through this connection.

- Retry count

This parameter defines how many times the plugin should try to resend data through the connection channel to the slave in case no response or wrong response gets received.

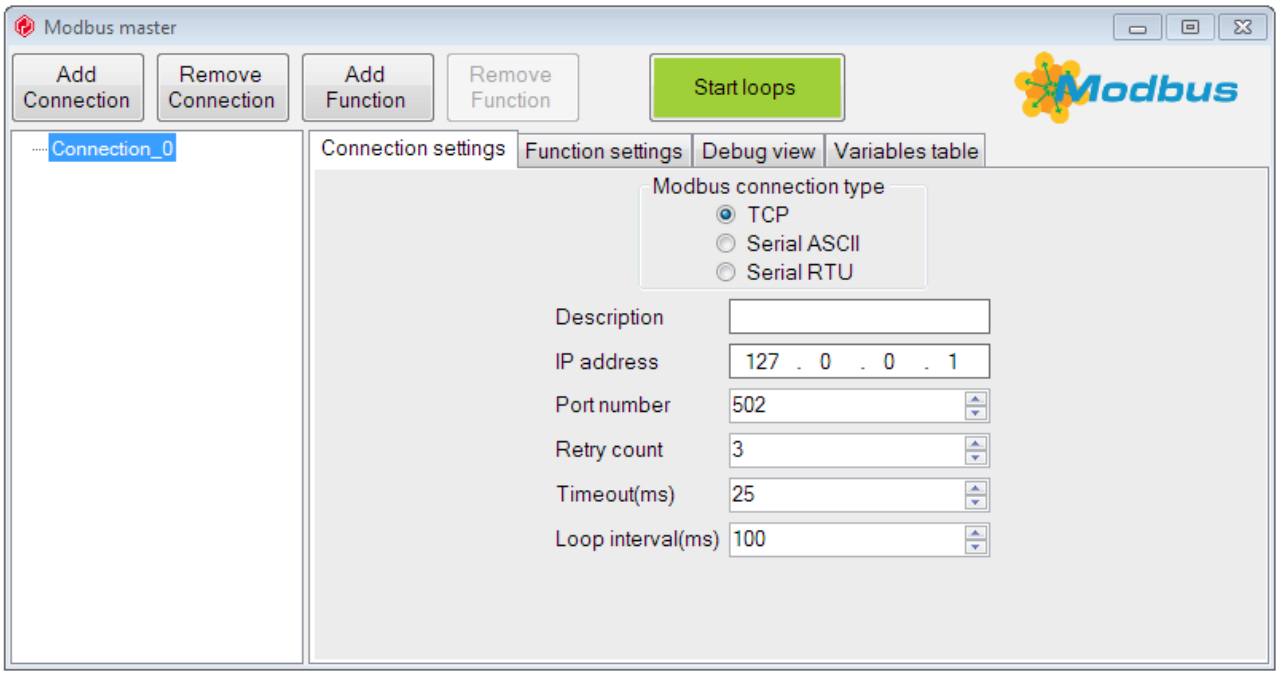
- Timeout(ms)

This parameter defines for how long the plugin should wait for the slave to answer before timing out the communication. The value is in milliseconds.

- Loop interval(ms)

This parameter sets the time to wait between running the Functions. When all functions in a connection ran the plugin waits this amount of time. The value is in milliseconds.

The following image shows the TCP connection settings:



2.2 .Serial ASCII and RTU connection.

If the Serial ASCII or RTU connection mode was selected then the Connection setup will show the corresponding connection specific parameters which are the same for these 2 connection types, because both connections work with serial communication protocol. The parameters are the following:

- Serial port.

This is the name of the serial port in the computer. The current possible selections are COM0 to COM99.

- Baud Rate

The baud rate defines the data rate per seconds on the serial communication channel.

- Data bits

The number of data bits per communication bytes, can be 7 or 8.

- Parity

The Parity bit in the communication, can be Non to have no parity bit or Odd or Even parity bit.

- Stop bits

The number of stop bits inserted at the end of the bytes. Can be 1 or 2.

- Retry count

This parameter defines how many times the plugin should try to resend data through the connection channel to the slave in case no response or wrong response gets received.

– Timeout(ms)

This parameter defines for how long the plugin should wait for the slave to answer before timing out the communication. The value is in milliseconds.

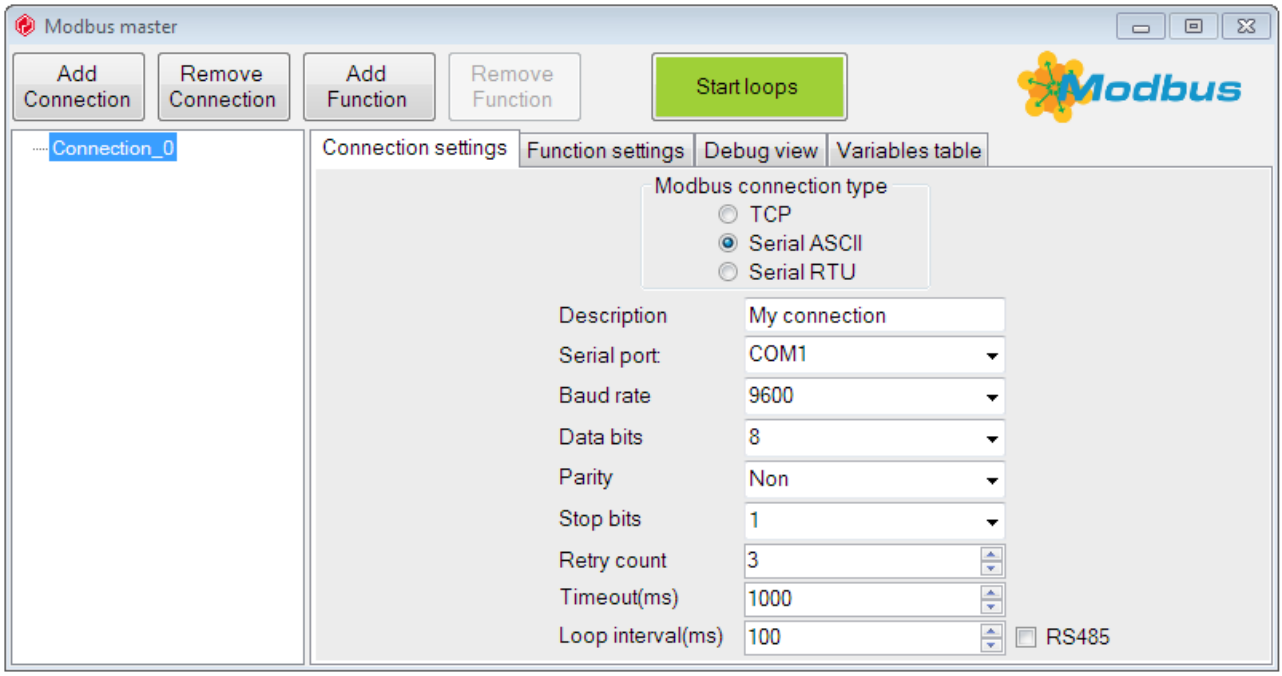
– Loop interval(ms)

This parameter sets the time to wait between running the Functions. When all functions in a connection ran the plugin waits this amount of time. The value is in milliseconds.

– RS485 mode

This is a boolean value represented by a checkbox. Setting the checkbox will enable RS485 mode.

The following image shows the Serial ASCII connection settings:



3 .The Functions.

The functions are the lower level objects, because they can be attached to connections.
To attach a new function to a connection first select a connection and press the Add function button. This will add a new function to the connection and the new Function will be visualized in the Tree view on the left side of the screen.
To remove a function select the function in the Tree view and press the Remove function button.
To edit the function properties click the name of the function in the Tree view and then select the Function settings tab page. The function properties can be setup on this page.

3.1 .Function types.

There are 8 types of functions supported by the plugin. There are 4 different read and 4 different write operations. The supported operations are the following:

- Read Coils

This operation reads a set number of coils from the modbus slave. The coils are boolean variables, their values can be 0 or 1, but the values are represented as 0 or 1 ushort variables in the UCCNC modbus register table.

- Read HoldingRegisters

This operations reads a set number of the holding registers from the modbus slave. The holding registers are 16bits wide unsigned variables. The possible range is therefor 0-65535.

- Read InputRegisters

This operations reads a set number of the input registers from the modbus slave. The input registers are 16bits wide unsigned variables. The possible range is therefor 0-65535.

- Read Inputs

This operations reads a set number of the inputs from the modbus slave. The inputs are 16bits wide unsigned variables. The possible range is therefor 0-65535.

- Write SingleCoil

This operation writes one coil of the modbus slave. The coils are boolean variables, their values can be 0 or 1, but the values are represented as 0 or 1 ushort variables in the UCCNC modbus register table.

- Write SingleRegister

This operation writes one holding register of the modbus slave. The holding registers are 16bits wide unsigned variables. The possible range is therefor 0-65535.

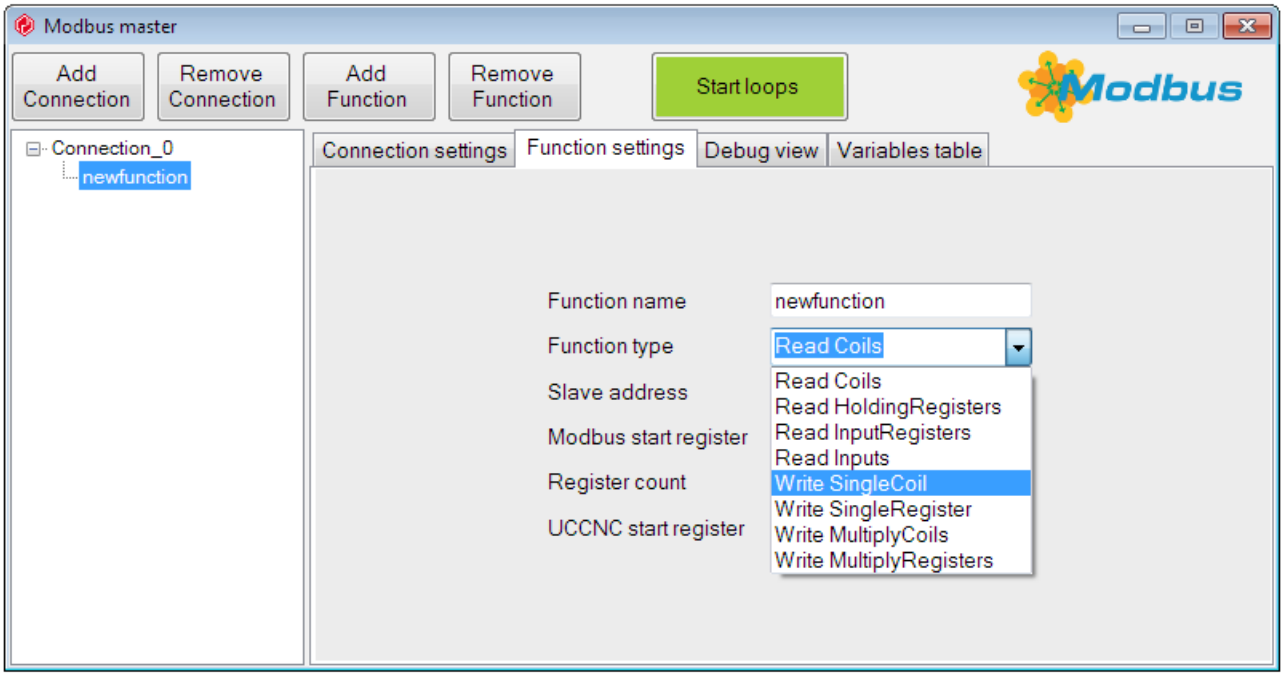
- Write MultiplyCoils

This operation writes a set number of coils of the modbus slave. The coils are boolean variables, their values can be 0 or 1, but the values are represented as 0 or 1 ushort variables in the UCCNC modbus register table.

- Write MultiplyRegisters

This operation writes a set number of holding registers of the modbus slave. The holding registers are 16bits wide unsigned short variables. The possible range is therefor 0-65535.

The following image shows the function types selection:



3.2 .Function parameters.

The function parameters define how the data registers are read and write from the UCCNC registers table to the slave device's modbus registers and from the slave device's modbus registers to the UCCNC registers table.
The parameters are the following:

- Function name

Is simply a note about the function. The function will appear under the set name in the Tree view.

- Function type

The Function type parameter selects the operation the function will do. The available function types were described in the previous chapter of this manual.

- Slave address

This parameter sets the address of the slave on the modbus this function should communicate with.

- Modbus start register.

This parameter defines the first register address in the slave's modbus register table where the operation will read from or will write to.

- Register count

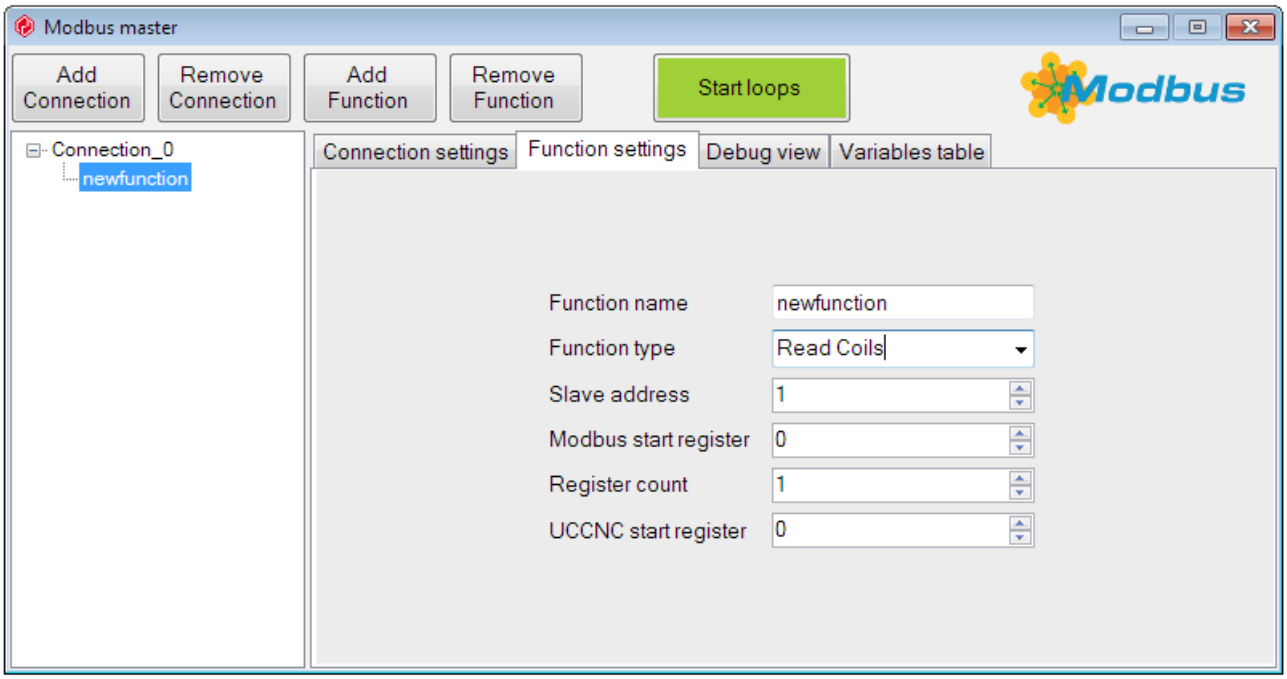
This parameter defines how many registers to read from the slave's modbus register table or write to the slave's modbus register table. The read or write will start from the Modbus start register address. If the operation is reading or writing a single Coil or Register then this parameter value can't be set to other than value 1.

– UCCNC start register

This parameter defines the first register in the UCCNC modbus register table where the read or write operations will move the data from or to.

In a read operation the data is moved from the slave's modbus register table to the UCCNC register table.
In a write operation the data is moved from the UCCNC register table to the slave's modbus register table.

The following image shows the function parameters:



4 .Running the communication loop.

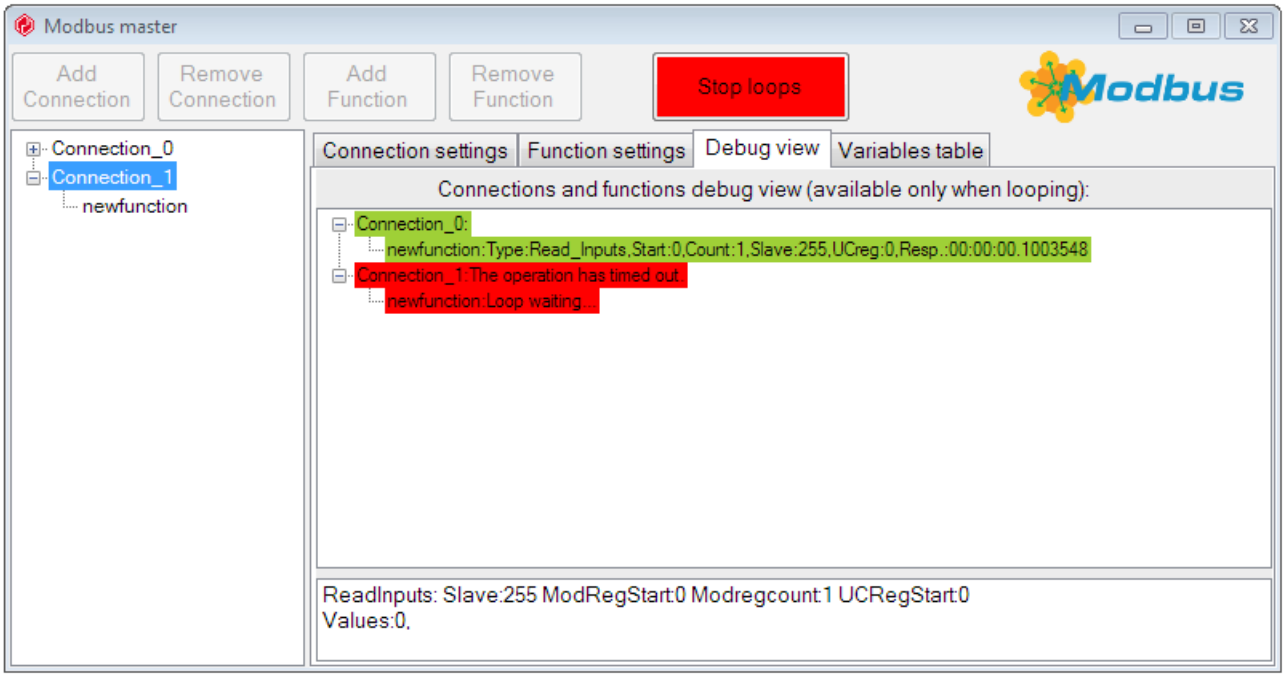
After setting up the connections and functions the functions can be started to run in a loop.
To start the communication loop press the Start loops button.
When the loop running the button turns red color and pressing the button again will stop the loop.
The loop is also automatically started when the plugin is loaded, so the modbus communication loops start running as soon as the software starts up and loads the Modbus master plugin.

5 .The debug view

To debug any functions of any connections select the Debug view tab page and if the communication loop is running then this page will show the same connection tree as on the left side of the screen, but this tree will show details about the connections and the functions running.
If a connection or function is running without issues then that node of the tree will be colored green or if the connection or the function has a problem then that tree node is colored red.
Also the tree node will show the properties of the connection or the properties of the issue

the connection or function has. Click the tree node to see it's properties in the Debug listbox at the bottom of the page.

The following image shows the debug view with 2 connections with 1-1 functions where the first connection is running without issues while the second connection points to a serial port with an unconnected modbus slave device:



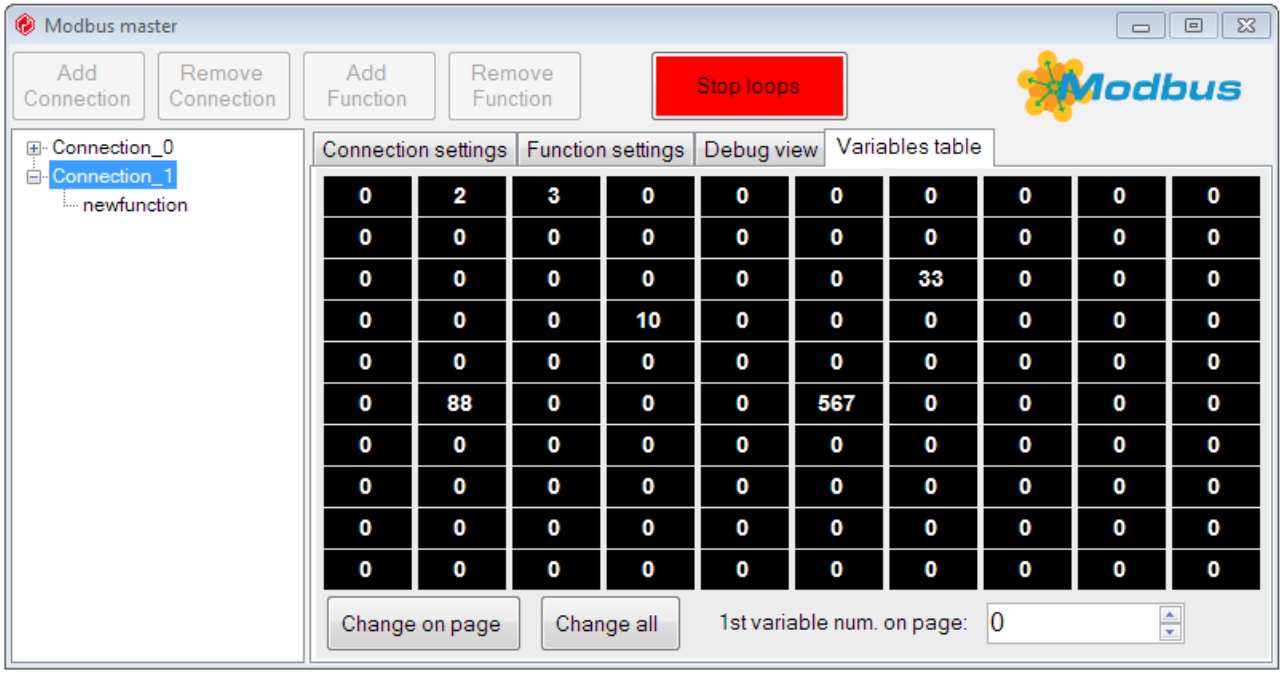
6 .The variables table

The UCCNC modbus variables table has a dimension of 100 000 ushort (16bit unsigned) numbers. This variable table is written when a modbus read operation is executed and is read when a modbus write operation is performed. The data is always moved between this data table and the modbus to the externally connected devices.

To view the UCCNC modbus variables select the Variables table tab page.
There are 100 pieces of variables shown on this page one time with the first variable on the upper left corner and the last parameter at the bottom-right corner.
To change the range of the variables shown write the first variable number to be viewed to the textbox in the bottom-right corner of the page.

To change the value of any variable click on the variable on the screen and type in the new value to the popup window. This will change the value of the variable in the UCCNC modbus register.
To change the values of a full 100 pieces of variables the same time to the same value press the Change on page button in the bottom-left corner of the page and type in the new value.
To change the values of the full variables table (100 000 variables) press the Change all button in the bottom-left corner of the page and type in the new value.

The following image shows the UCCNC modbus variables table:



7 .Accessing the Modbus variables table from within the UCCNC.

The UCCNC modbus registers are accessible from UCCNC macros and plugins through the following functions:

ushort[] GetAllModbusArray()

Reads the whole modbus register table.

bool SetModbusregister(int Registernumber, ushort Value)

Writes a single register in the modbus register table.

bool SetModbusregisters(int Startregister, ushort[] Values)

Writes multiply registers in the modbus register table.

bool GetModbusregister(int Registernumber, out ushort Value)

Reads a single register from the modbus register table.

bool GetModbusregisters(int Startregister, int Registercount, out ushort[] Values)

Reads multiply registers from the modbus register table.

void WriteModbusString(string String, int Startregister, bool HightoLowByteorder)

Converts a string into an ushort array and writes it into the modbus register table.

In the functions where the return value is boolean indicates if the function got executed with or without problems. The true return value means that the function executed without problems while false means that a problem has occurred and the retrieved variable(s) values may be corrupt.

8 .Example macro codes

The following example code reads the modbus register 0 from the UCCNC modbus table and writes the value into the Textfield (DRO) with ID=2000.
To test run the macro create a Textfield with ID=2000 using the Screen editor and place the below text into a text macro file in the /Profiles/Macros folder.
Setup a Modbus Connection and Function in the Modbus plugin which function reads value into register 0.
Execute the macro via MDI.
To execute the macro in a loop place the macro file to the macro loops in the General settings/Configure macroloops and place your macro number into one loop slots.
Running the macro in a loop will continously read the modbus register 0 and will write the read value into the DRO.
The code is the following:

```
ushort Readvalue;
if(exec.GetModbusregister(0, out Readvalue))
{
  AS3.Setfield(Readvalue, 2000);
}
else
{
  //The read returned with error, handle it...
}
```

The following example code reads a DRO and writes the value into the modbus register 0.
To test run the macro create a Textfield with ID=2000 using the Screen editor and place the below text into a text macro file in the /Profiles/Macros folder.
Setup a Modbus Connection and Function in the Modbus plugin which writes register 0 value onto the Modbus.
Execute the macro via MDI.
To execute the macro in a loop place the macro file to the macro loops in the General settings/Configure macroloops and place your macro number into one loop slots.
Running the macro in a loop will continously read the DRO value and will write it to the modbus.
The code is the following:

```
ushort Writevalue = (ushort)AS3.Getfielddouble(2000);
if(!exec.SetModbusregister(0, Writevalue))
{
 //The write returned with error, handle it...
}
```