

Content

1. General Information.....	3
1.1. Version.....	3
1.1.1. User Manual.....	3
1.1.2. Technikmedia IPEmotion Plug In Universal Modbus.....	3
1.1.3. IPEmotion.....	3
2. Introduction.....	3
3. Setting Up And Removing	4
3.1. System Requirements.....	4
3.1.1. Hardware	4
3.1.2. Platforms.....	4
3.2. Installation	4
3.3. Uninstall	7
4. Licenses	7
4.1. Demo License	7
4.2. Full Version	7
5. Working with technikmedia IPEmotion Universal Modbus Plug In.....	8
5.1. Adding a Universal Modbus system to IPEmotion.....	8
5.1.1. Adding Modbus devices	9
5.1.2. Define function code groups.....	11
5.1.3. Adding register values.....	12
5.1.4. Modbus addressing schema	13
5.1.5. Output values	14
6. Pitfalls.....	16
6.1. Hardware does not support data block reading.....	16
6.2. No values or wrong values while acquisition because documentation starts with address 0.....	17
6.3. Implausible values.....	17
6.4. No values because high sample rate	18

1. General Information

This manual describes the usage of the **technikmedia IPEmotion Plug In Universal Modbus**. Please read this manual carefully to get to know the operating and to learn more about the functions and special features. This manual also contains information for installing and removing the software.

1.1. Version

1.1.1. User Manual

This manual has the version 01.03.

1.1.2. Technikmedia IPEmotion Plug In Universal Modbus

The description in this documentation refers to the current release with the version number 01.01.12

1.1.3. IPEmotion

The description in this documentation refers to the current release with the version number 2015 R1.

2. Introduction

The **technikmedia IPEmotion Plug In Universal Modbus** offers you the ability to read data from or write data into devices with Modbus interface.

Actually the following Modbus variants are supported:

- Modbus TCP
- Modbus RTU over TCP
- Modbus serial RTU

The supported function codes are:

Reading

- Function Code 1, Read Coils
- Function Code 2, Read Inputs Discrete
- Function Code 3, Read Holding Registers
- Function Code 4, Read Input Registers

Writing

- Function Code 5, Write Single Coil
- Function Code 6, Write Single Register
- Function Code 15, Write Multiple Coils
- Function Code 16, Write Multiple Registers

Depending on the used function code, the following data types are supported:

- Bit
- 16 bit integer
- 16 bit unsigned integer
- 32 bit integer (combined 2 x 16 bit addresses)
- 32 bit unsigned integer (combined 2 x 16 bit addresses)
- 32 bit float (combined 2 x 16 bit addresses)

3. Setting Up And Removing

3.1. System Requirements

The minimum hardware and platform requirements for the application of the **technikmedia IPEmotion Plug In Universal Modbus** are outlined below.

3.1.1. Hardware

The minimum hardware requirements correspond to those of IPEmotion.

3.1.2. Platforms

The Plug In can be run under the following operating systems:

- Windows 7 (32 Bit) or higher

3.2. Installation

The following chapters guide you through the installation process of the Universal Modbus Plug In.

technikmedia IPEmotion Plug In Universal Modbus needs administrator rights during installation.

The installation of the plug in is based on an installation wizard that guides you through the setup process step by step.

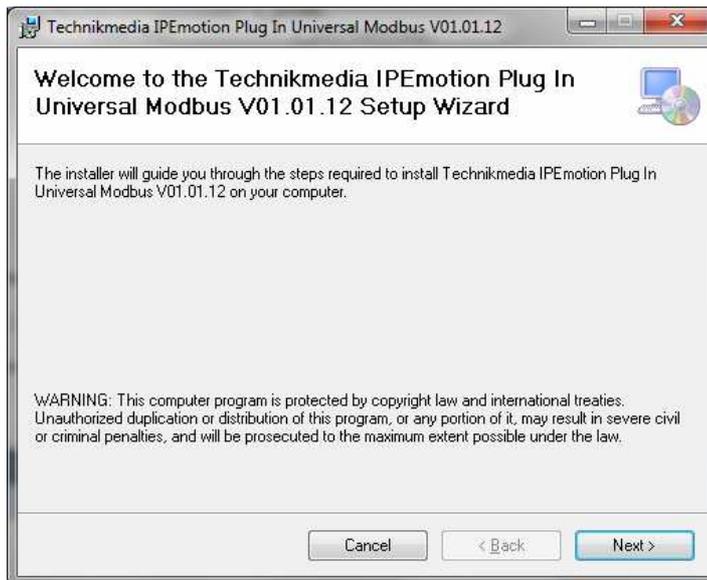
To install the plug in:

1. Double-click the file

Setup Technikmedia IPEmotion Plug In Universal Modbus Vxxx.exe

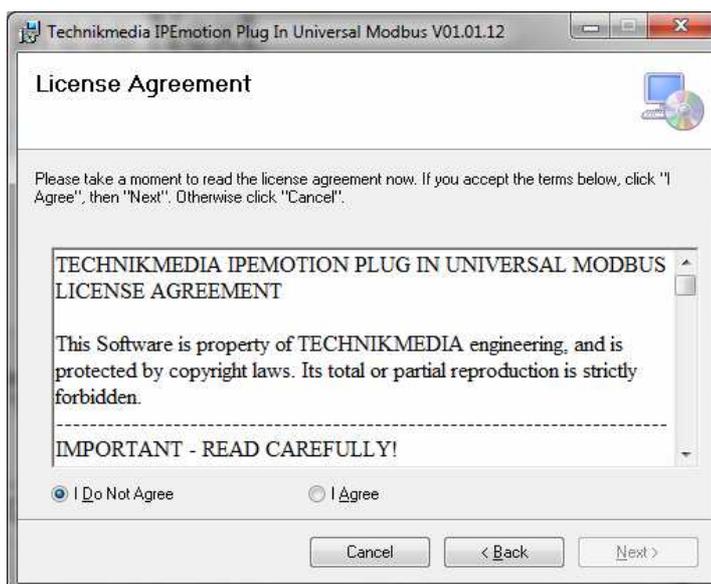
to start the installation wizard.

2. **Welcome screen:** This is the first screen in the plug in installation wizard.

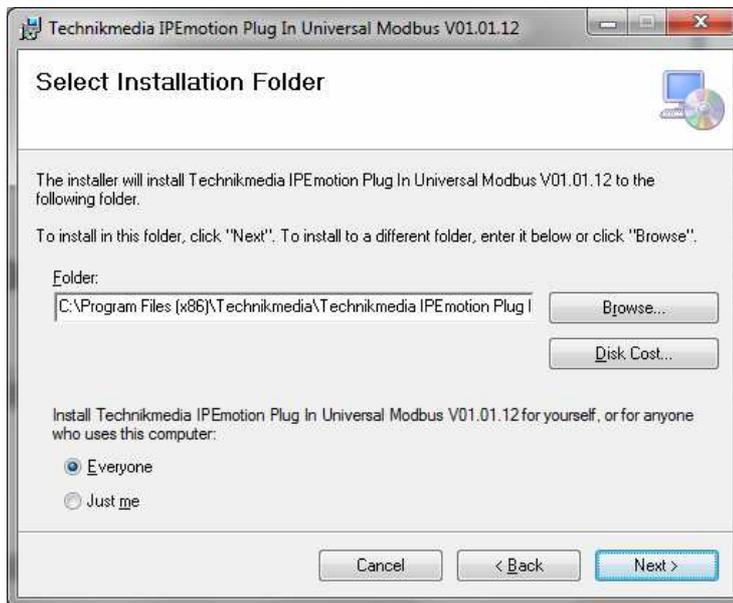


Click **Next** to continue.

3. **License Agreement:** If you agree with the license agreement, check the radio button **I Agree** and press **Next**.



4. **Installation folder:** Accept the default installation location for the plug in. To select another location click **Browse...** and select another folder.

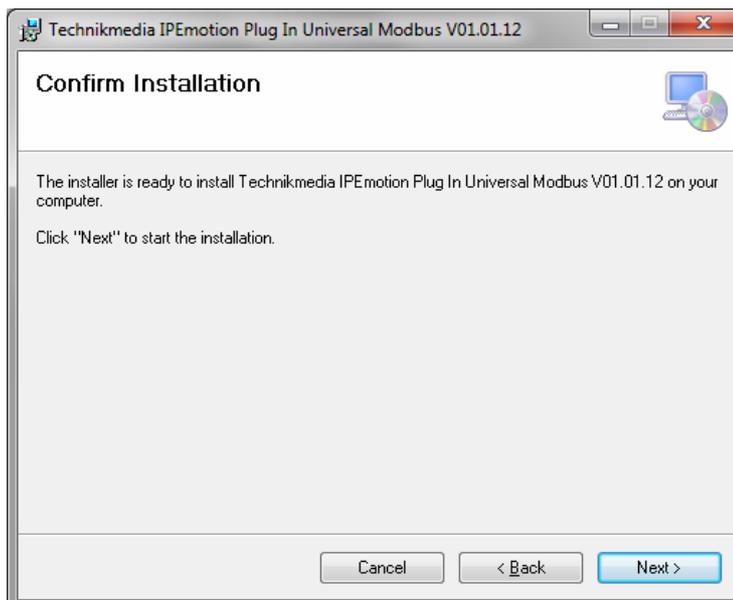


To get information about available disk space click **Disc Cost...**

By default the plug in is available for anyone. If you want to use it for yourself select **Just me**.

Click **Next** to continue.

5. **Confirm installation:** This screen indicates that the plug in is ready to install.



Click **Next** to start installation.

6. **Installation:** A progress bar is shown during the installation process.
7. **Installation complete:** After successful installation this screen is shown.

Click **Close** to finish installation.

3.3. Uninstall

Windows XP:

1. Click on **Start** button, click on **Settings** and then on **Control Panel**, click on **Add or Remove Programs**
2. Select the **technikmedia IPEmotion PlugIn Universal Modbus** and then click **Remove** to start the installation wizard.
3. Click **Yes** to confirm removal.

Windows 7:

1. Open Programs and Features by clicking the **Start** button , clicking **Control Panel**, clicking **Programs** and the clicking **Programs and Features**
2. Select the **technikmedia IPEmotion PlugIn Universal Modbus** and then click **Uninstall**.
3. Click **Yes** to confirm removal.

After the successful removal of the plug in the program has been removed from your computer and is no longer indicated in the program list.

4. Licenses

4.1. Demo License

If you have an IPEmotion demo version you automatically can test the plug in until the demo license is expired.

4.2. Full Version

With your full version you have no limitations.

5. Working with technikmedia IPEmotion Universal Modbus Plug In

The following chapters offer an overview about the usage of the plug in to handle connected Modbus systems. It shows how to configure modules and acquire data. The documentation of analysing and managing the acquired data will not be part of this manual. To get further information for these topics see IPEmotion documentation.

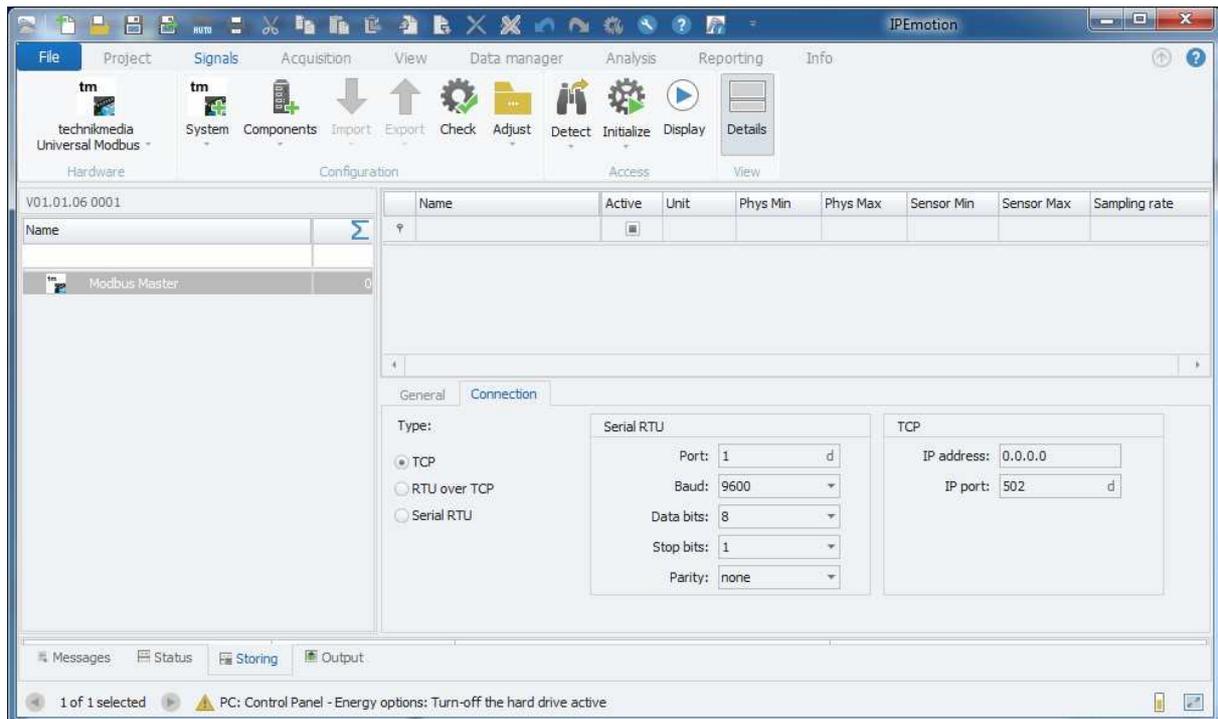
This documentation describes the special features and functions of the plug in. Common IPEmotion functionalities are not part of this documentation. To get further information, see the IPEmotion documentation.

5.1. Adding a Universal Modbus system to IPEmotion

First select the **technikmedia Universal Modbus** plug in in the IPEmotion **Hardware** selection and then the **Modbus Master** in the **System** selection



The Modbus Master is an IPEmotion system that represents a connection to the Modbus system. The Modbus system can be connected via TCP or serial connection to the PC. After selecting the **Modbus Master** you have to define the connection to the Modbus system in the tab **Connection**.



To connect the Modbus system via TCP:

1. Select **TCP** Radio Button
2. Enter **IP Address** and **IP port** in the TCP group box

To connect the Modbus system via RTU over TCP:

1. Select **RTU over TCP** Radio Button
2. Enter **IP Address** and **IP port** in the TCP group box

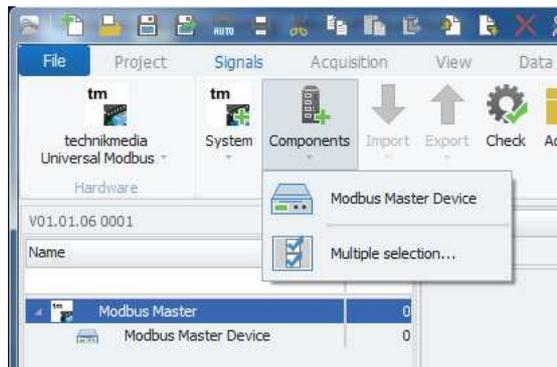
To connect the Modbus system via serial RTU:

1. Select **Serial RTU** Radio Button
2. Enter serial **Port**, **Baud rate**, **Data Bits**, **Stop Bits** and **Parity** in the Serial RTU group box

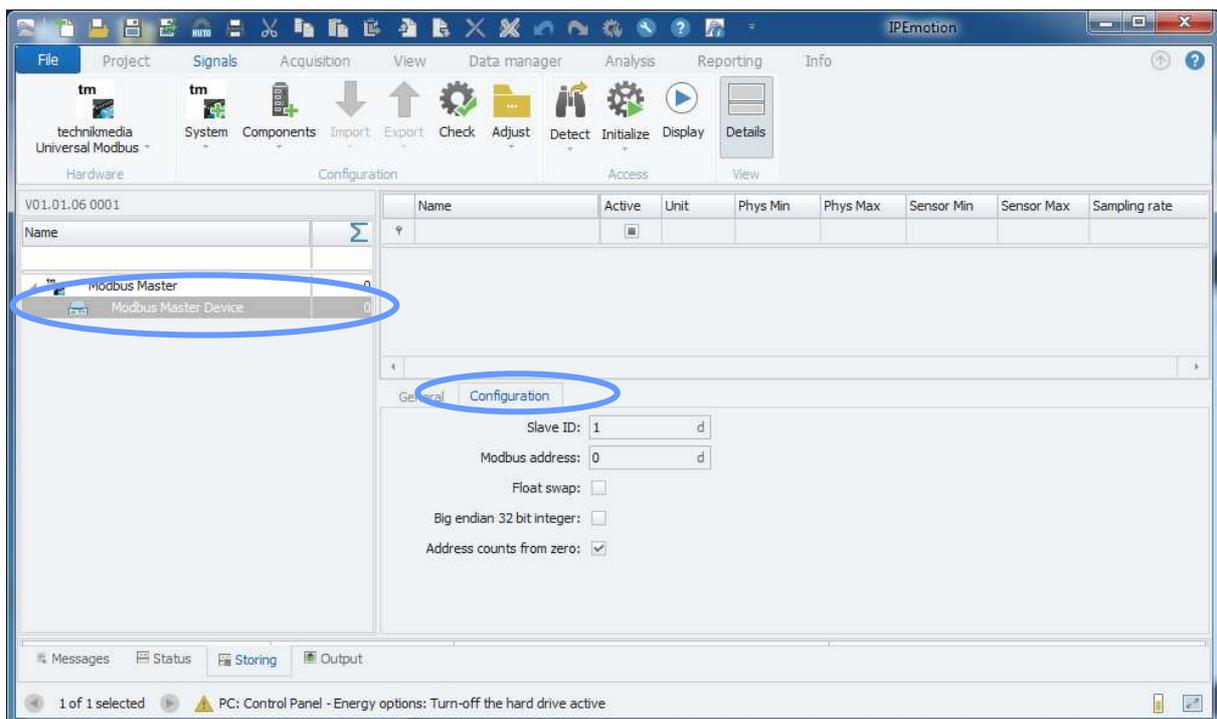
5.1.1. Adding Modbus devices

In general every Modbus system is build by one or more Modbus devices (real hardware). To add a device in IPEmotion you have to add a **Modbus Master Device** by clicking on the **Modbus Master Device** item in the IPEmotion **Components**

selection.



In the plug in every Modbus device has characteristic parameters you have to configure. You find the configuration by clicking on the **Modbus Master Device** in the tab **Configuration**.



The following parameters can be configured:

- Slave ID** Modbus slave id
- Modbus address** The Modbus address is the base address of the device.
- Float swap** A float value in this plug in uses two 16 bit Modbus registers. Most devices store float values in little endian format, but some platforms perform a word swap. To

handle these float values you have to check this parameter.

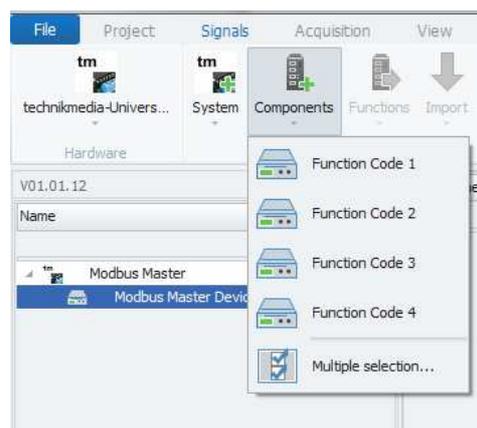
Big endian 32 bit integer 32 bit integers need two 16 bit Modbus registers. By default the values are assumed as in little endian format. This means low word first. If this parameter is checked, the values will be handled in big endian format.

Address counts from zero Some device documentations start from Modbus address zero, but Modbus specifies the first address with 1. To let you use the Modbus addresses used in the documentation, you can check this parameter. Then the plug in adds 1 to the given Modbus address.

5.1.2. Define function code groups

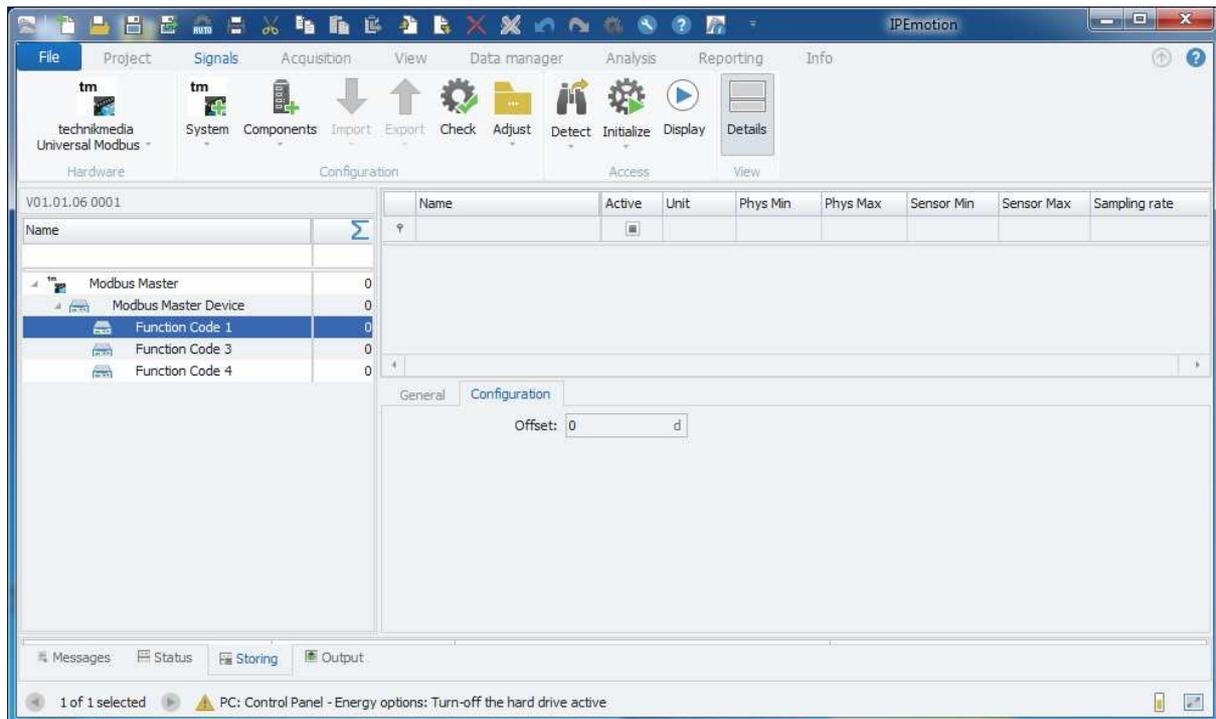
The plug in only supports coils and registers that are readable (and optional are also writable). To configure your Modbus system, you can select one or more of the following function codes

- Function Code 1, Read Coils
- Function Code 2, Read Input Discretes
- Function Code 3, Read Holding Registers
- Function Code 4, Read Input Registers



To add a function code to your device, first select your **Modbus Master Device** and then click on the IPEmotion **Components** selection. Select the function code.

You have to configure the function code by setting the offset address in the tab **Configuration**.



Every device uses coils or registers for storing input and/or output data. In a device configuration you can configure one or more function code groups of the same or different type. Every function code group is a container of coil or register values. The values belonging to a function code group have a start address that belongs to the first value in the list.

The start address is a combination of the Modbus address of the Modbus device and the offset of the function code group. For a detailed description see next chapter.

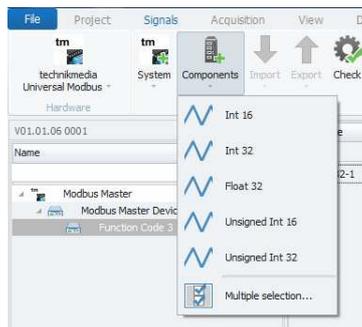
5.1.3. Adding register values

Depending on the function code you added to a device, a selection of values is available. The following table shows the available values.

Function Code	Value type	Description
1	Bit	Bit
2	Bit	Bit
3	Int 16	16 bit integer
	Int 32	32 bit integer (2 addresses)
	Float 32	32 bit float (2 addresses)
	Unsigned Int 16	16 bit unsigned integer
	Unsigned Int 32	32 bit unsigned integer
4	Int 16	16 bit integer

Int 32	32 bit integer (2 addresses)
Float 32	32 bit float (2 addresses)
Unsigned Int 16	16 bit unsigned integer
Unsigned Int 32	32 bit unsigned integer

To add a value to the function code, click on the target function code and click on **Components** selection. Select the value type you want to add or use the **Multiple selection...** item.



You can add values of different types to a function code group.



5.1.4. Modbus addressing schema

As already mentioned a Modbus system in this plug in consist of one or more devices. Every device has one or more function code groups.

To handle the special situation of your Modbus hardware you can set a Modbus base address for every device and then you can define an offset for every function code group you want to configure. The start address of your function code group is always:

Start address = Modbus Device Address + Offset function code group

It is not important to set the Modbus device address but helps to use addresses which correspond with the Modbus hardware documentation.



Every function code group has an address range without gaps. That means that the first value has the start address of the function code group. If the value is a bit value, a 16 bit integer or a 16 bit unsigned integer, the address of the next value in this function code group has the next address. If the value needs 32 bit, the address of the next value is the next but one. The following table shows the addresses of the example value list above.

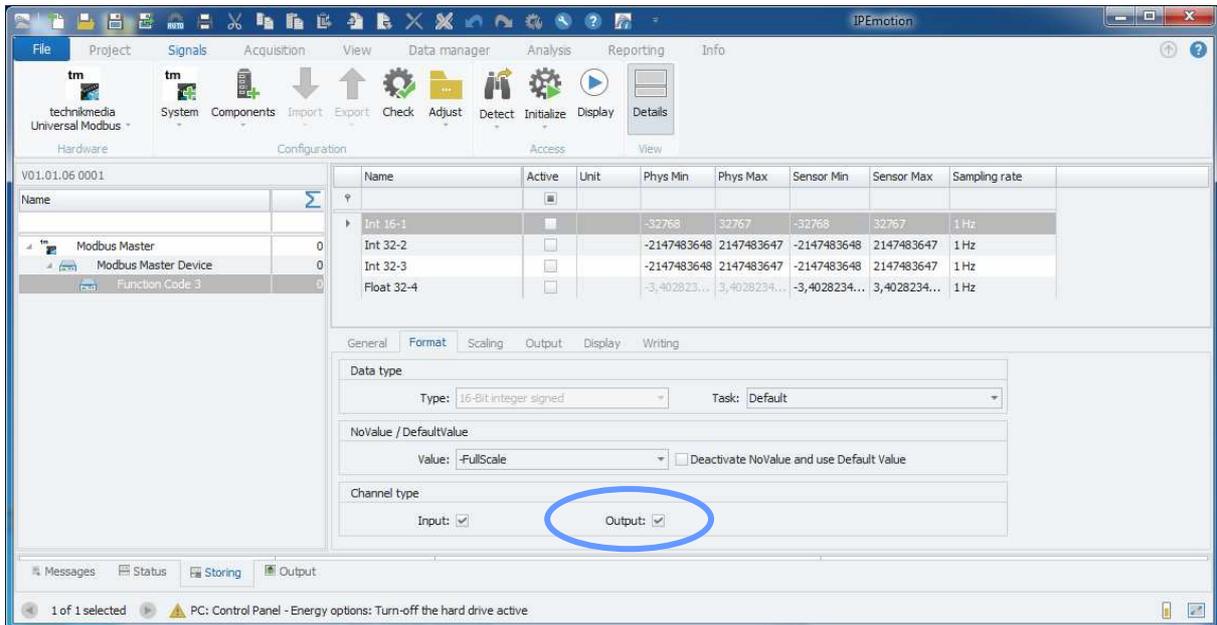
Assumed Start Address = 0 (Modbus Device Address) + 100 (Function code offset)

Value name	Value type	Address
Int 16-1	Int 16	100 (= Start Address)
Int 32-2	Int 32	101
Int 32-3	Int 32	103
Float32-4	Float 32	105

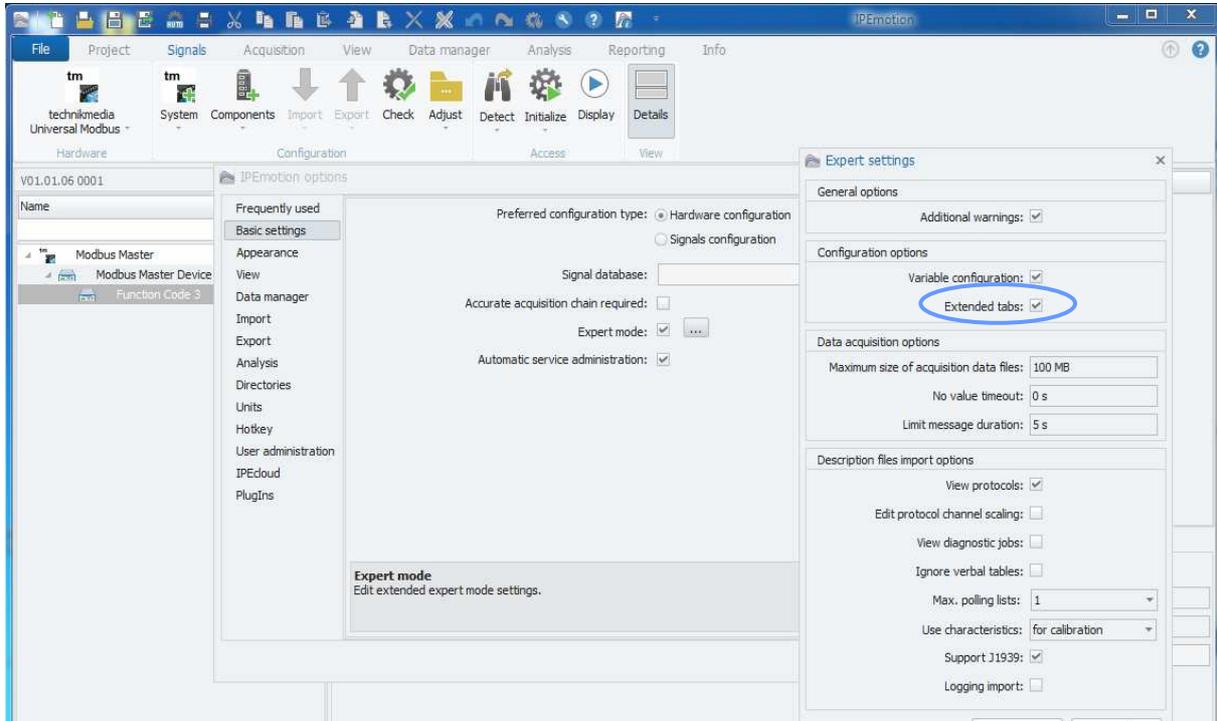
If you have a gap in your Modbus address space, you have to create a new function code group and set the new start address.

5.1.5. Output values

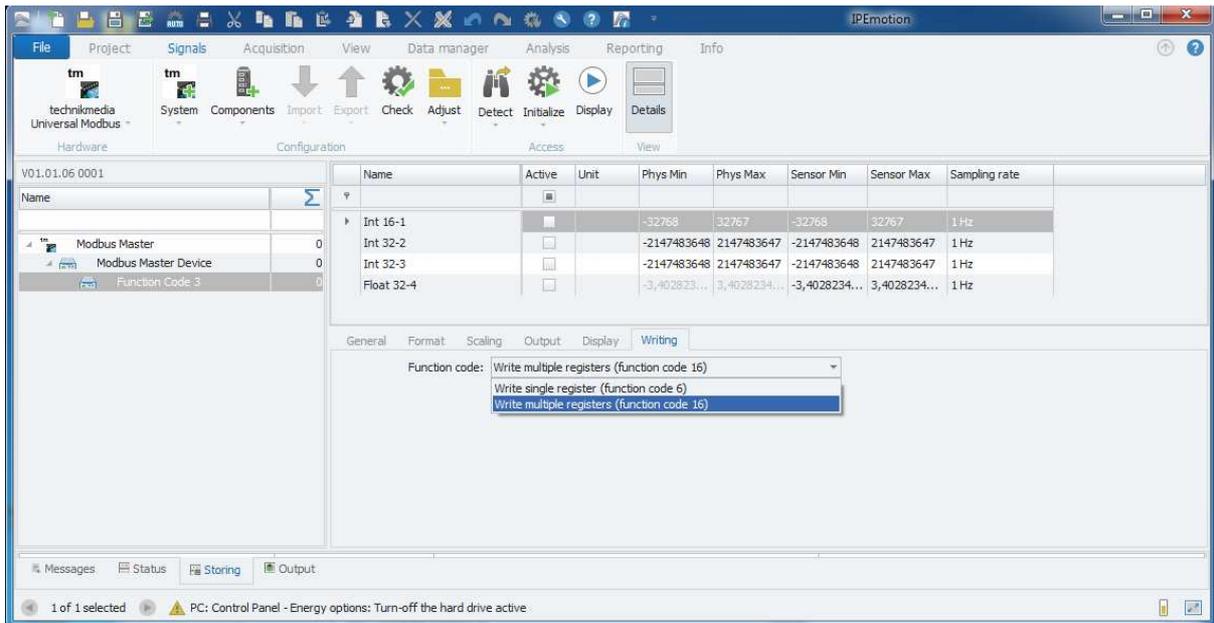
All values in the plug in are input values. That means all values belong to a function code that represents a read operation. To define a value as an output value, you have to set the **Channel type** in IPEmotion also to **Output**.



Hint! If you can not see the tab **Format**, you have to check the **Extended tabs** in **File->Options->BasicSettings->Expert Mode**.



The next step is to define the function code for writing. Select the tab **Writing** and choose the code in the combo box **Function code**:



The function code for writing depends on the data type of you variable. The following table shows the possible codes:

Value type	Function code
Bit	Function code 5: Write coil
	Function code 15: Force multiple coils
Int 16	Function code 6: Write single register
	Function code 16 Write multiple registers
Int 32	Function code 16 Write multiple registers
Float 32	Function code 16 Write multiple registers
Unsigned Int 16	Function code 6: Write single register
	Function code 16 Write multiple registers
Unsigned Int 32	Function code 16 Write multiple registers

6. Pitfalls

This chapter shows some problems that can occur while configuring the plug in for your individual Modbus hardware.

6.1. Hardware does not support data block reading

In general Modbus function code commands allow to read registers in blocks from the hardware. For example you can read 20 registers with one function code 3

command. Some hardware supports e.g. function code 3 but it does not support the block reading. In that case the plug in will show **No value** while an acquisition.

To fix this problem you have to create a function code block for every value.

6.2. No values or wrong values while acquisition because documentation starts with address 0

If you get no values or wrong values from your hardware a possible pitfall is a wrong Modbus start address.

The Modbus address range starts with 1 (+ base address for function code). Usually the base address for a special function code is not mentioned in Modbus hardware documentation. A parameter is described with a Modbus address (without base address for function code) and the function code which has to be used for reading or writing. Some documentations start not with Modbus address 1 but with 0.

For example:

If your documentation starts with address 0 and defines a parameter address of 1100, the Modbus address in reality is 1101. The usage of 1100 address in your function code may not be accepted by the hardware and IPEmotion gets no data.

You have two ways to solve the problem:

1. You add 1 to the offset of the function code group
2. You can check the parameter **Address counts from zero** in the Modbus device configuration tab (see chapter 5.1.1)

6.3. Implausible values

While running an acquisition some values always show implausible values.

Are these values of type Int 32, Unsigned Int 32 or Float 32?

If true, it is possible that the expected storage format of the values does not correspond with the format in the hardware. The values are stored in two 16 bit registers. The plug in may not use the same storage order of the high and low word as the hardware.

If the documentation of your hardware gives you no hint, please try out for

Int 32 and Unsigned Int 32

Check parameter **Big endian 32 bit integer** in the Modbus device configuration tab (see chapter 5.1.1)

Float32

Check parameter **Float swap** in the Modbus device configuration tab (see chapter 5.1.1)

6.4. No values because high sample rate

The plug in allows a high sample rate for reading data. If you have configured a high sample rate and your Modbus hardware or your connection only allow low sample rates, it may happen that you get no values or maybe sometimes no values. Please try out a lower sample rate in the plug in.