WAGO I/O SYSTEM 750

Ethernet Controller 750-842
Modbus Master

Application Notes

A300002, English
Version 1.0.1

WAGO®
INNOVATIVE CONNECTIONS

WAGO Kontakttechnik GmbH
Hansastraße 27
D-32423 Minden

Phone: +49 (0) 571/8 87 – 0
Fax: +49 (0) 571/8 87 – 1 69

E-Mail: info@wago.com

Web: http://www.wago.com

Technical Support
Phone: +49 (0) 571/8 87 – 5 55
Fax: +49 (0) 571/8 87 – 85 55

E-Mail: support@wago.com

Every conceivable measure has been taken to ensure the correctness and completeness of this documentation. However, as errors can never be fully excluded we would appreciate any information or ideas at any time.

We wish to point out that the software and hardware terms as well as the trademarks of companies used and/or mentioned in the present manual are generally trademark or patent protected.

TABLE OF CONTENTS

1 Important comments

To ensure fast installation and start-up of the units described in this manual, we strongly recommend that the following information and explanation is carefully read and adhered to.

1.1 Legal principles

1.1.1 Copyright

This manual is copyrighted, together with all figures and illustrations contained therein. Any use of this manual which infringes the copyright provisions stipulated herein, is not permitted. Reproduction, translation and electronic and photo-technical archiving and amendments require the written consent of WAGO Kontakttechnik GmbH. Non-observance will entail the right of claims for damages.

1.1.2 Personnel qualification

The use of the product detailed in this manual is exclusively geared to specialists having qualifications in PLC programming, electrical specialists or persons instructed by electrical specialists who are also familiar with the valid standards. WAGO Kontakttechnik GmbH declines all liability resulting from improper action and damage to WAGO products and third party products due to non-observance of the information contained in this manual.

1.1.3 Intended use

For each individual application, the components supplied are to work with a dedicated hardware and software configuration. Modifications are only admitted within the framework of the possibilities documented in the manuals. All other changes to the hardware and/or software and the non-conforming use of the components entail the exclusion of liability on part of WAGO Kontakttechnik GmbH.

Please direct any requirements pertaining to a modified and/or new hardware or software configuration directly to WAGO Kontakttechnik GmbH.

1.2 Range of validity

This application note is based on the stated hardware and software of the specific manufacturer as well as the correspondent documentation. This application note is therefore only valid for the described installation.

New hardware and software versions may need to be handled differently. Please note the detailed description in the specific manuals.

1.3 Symbols

Attention
Marginal conditions must always be observed to ensure smooth operation.

More information
References to additional literature, manuals, data sheets and INTERNET pages

2 Description

2.1 General

The ETHERNET_MODBUSMASTER_UDP and ETHERNET_MODBUS-MASTER_TCP function blocks have been designed to allow for efficient programming of the data exchange between several programmable fieldbus controllers (PFC) 750-842 or between one fieldbus controller and one or more Modbus slave devices like the Ethernet fieldbus couplers 750-342.

If possible, the ETHERNET_MODBUSMASTER_UDP function block should be used. It uses the UDP protocol which allows for faster data transmission.

The MODBUSMASTER_TCP function block has been designed for data transmission with products that are based on MODBUS TCP exclusively (e.g. operation terminal E300 Beijer).

The communication is based on the Master/Slave principle with MODBUS functionalities (see 2.4).

The controller along with the ETHERNET_MODBUSMASTER_UDP/TCP function block is the master, the controllers/couplers or devices from different manufacturers (all addressed via IP Address) are slaves.

2.2 Prerequisite

This application note assumes that the handling of WAGO-I/O-PRO 32 is understood. For information on assembly, installation, start-up and functioning of the Ethernet Controllers, please refer to the manual.

More information
You find the manuals on the INTERNET page
http://www.wago.com/

2.3 Libraries

The Ethernet Controller 750-842 requires at least firmware version 02.02.00(04You can retrieve the specifications of the version via the IP address of the controller/coupler using the Internet Explorer.
The library manager of the project must include the following libraries:

Bibliothek	Beschreibung
Standard.lib	Basisfunktionen
System.lib	Hilfsfunktionen
Ethernet.lib	Netzwerktreiber
ModbusEthernet_03.lib	Modbus-Master (see LibraryManual ML00100)

2.4 MODBUS Functions (Function Codes FC)

In accordance with manual (Ethernet Coupler / Controller), the following MODBUS Function Codes are implemented into the function blocks of the controller:

Function code hex		Function
FC1:	0x01	Read coils
FC2:	0x02	Read input discretes
FC3:	0x03	Read multiple registers
FC4:	0x04	Read input registers
FC5:	0x05	Write coil
FC6:	0x06	Write single register
FC7:	0x07	Read exception status
FC11:	0x0B	Get comm event counter
FC15:	0x0F	Force multiple coils
FC16:	0x0010	Write multiple registers
FC23:	0x0017	Read/Write multiple registers

2.5 Material

Manufacturer	Article No.	Description
WAGO	750-842/-342	Ethernet Controllers (two at least)
WAGO	750-xxx	Digital/analog input or output modules, depending on application.
		Commercially available hub/switch
		Ethernet connecting cable with RJ45
		Standard PC with WAGO-I/O-PRO 32

3 Solutions

3.1 Data exchange with one MODBUS slave

The following describes the data exchange between an Ethernet controller 750-842 and an Ethernet-Coupler 750-342.

On the controller running a program that add the Modbus-Master functionality.
The Coupler allready implements the Modbus Slave functionality inside the firmware.

The exchange is made using the function block ETHERNET_MODBUSMASTER_UDP and the MODBUS function code 23 (read/write registers).

The programming language is function plan (FUP).

IP address assignment

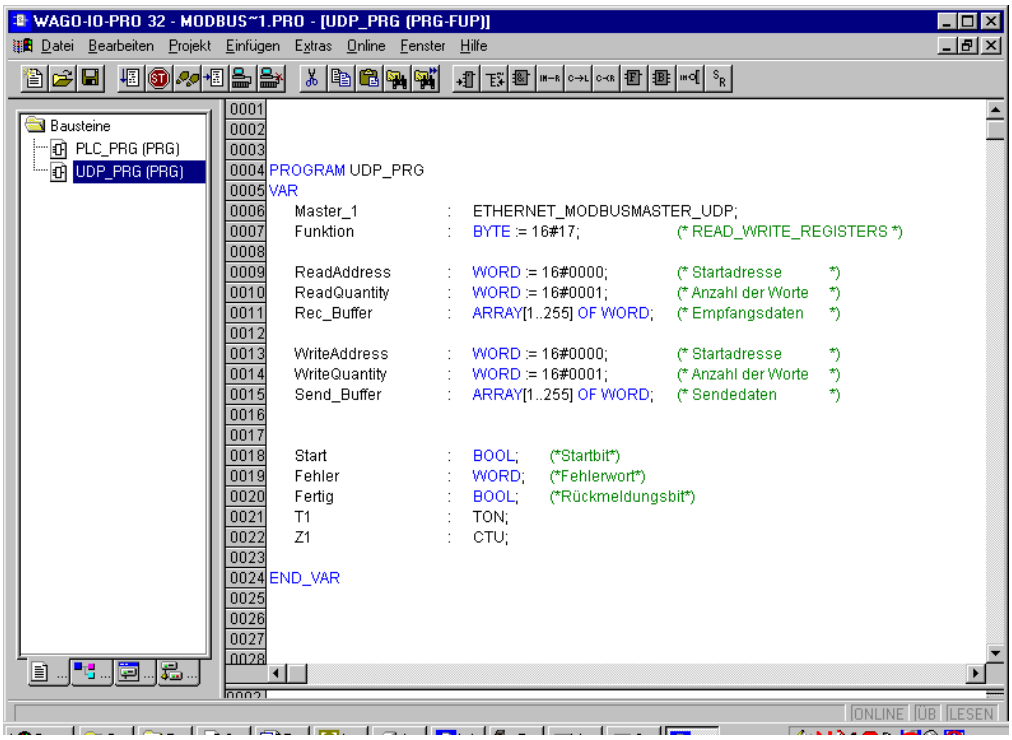
The IP address can be assigned using the WAGO BootP Server.

Further information
The WAGO BootP Server can be downloaded from the WAGO INTERNET page free of charge:
http://www.wago.com/

The use of the BootP Server is described in the manual 750-842 / 342.

Programming

The variable declaration must contain the variables listed below. They can be parameterized according to the application.



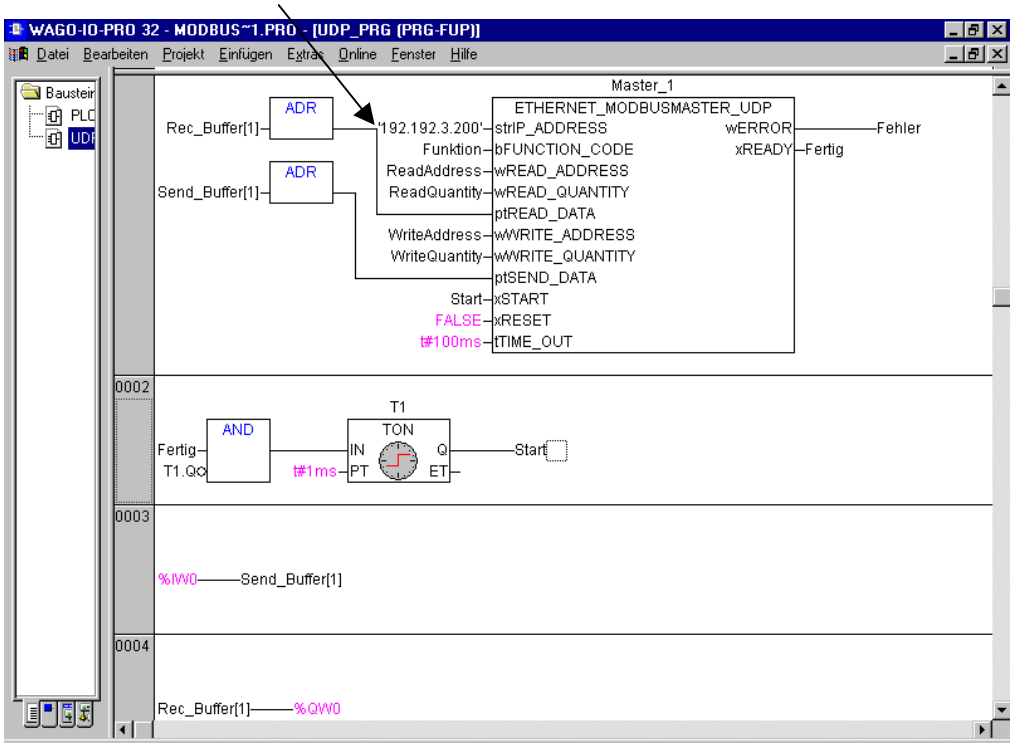
- The start bit (**Start**) begins the data transmission. As soon as the transmission has been completed successfully the return bit **“Fertig”** will be set to 1.
- The variable **“Funktion”** defines the Modbus Code. In the example given above it is 23 (corresponds to 17 Hex = read/write registers).
- **“ReadAddress”** defines the start address for the words to be read in the slave (example 16#0000 equals %IW0)
- **“ReadQuantity”** indicates the number of words to be read.
- **“Rec_Buffer”** stands for the input buffer of the master for the data to be read.
- **“WriteAddress”** and **“WriteQuantity”** indicate the start address and the number of words in the slave (e.g. WriteAddress 16#0001 equivalent to %QW1 in the slave).
- **“Send_Buffer”** stands for the send buffer in the master for the words to be written.
- **“Fehler”** contains the error code in the event of a faulty data transmission.

Programming example:

The ETHERNET_MODBUSMASTER_UDP function block is processed in the master. Hence, the other controller, whose IP address is registered in the module, becomes a slave.

The data exchange is as follows:
the input word %IW 0 (= e.g. first 16 input bits) is assigned to the first word in the Send_buffer (network 3). Master_1 writes the word on address 0 of the slave (equals output word %QW0 = 16 output bits).
On the other hand, the input word of the slave will be read and written into the output word of the master.

IP address of the slave



The Timeout value is directly written into the function block. This is the maximum time the master waits for an answer of the slave. If the slave does not respond during that time, the error code will be written into the error word.

Dimensioning for Rec_Buffer and Send_Buffer:
The Rec_Buffer should be configured for the max. possible data capacity. A too small dimensioned Rec_Buffer can overflow when receiving a lengthy telegram. Data can thereby be destroyed, which can lead to non-predictable errors.
A too small dimensioned Send_Buffer transmits coincidental data to the slave.

3.2 Data exchange with three MODBUS slaves

The second example describes the data exchange between one master and three slaves.

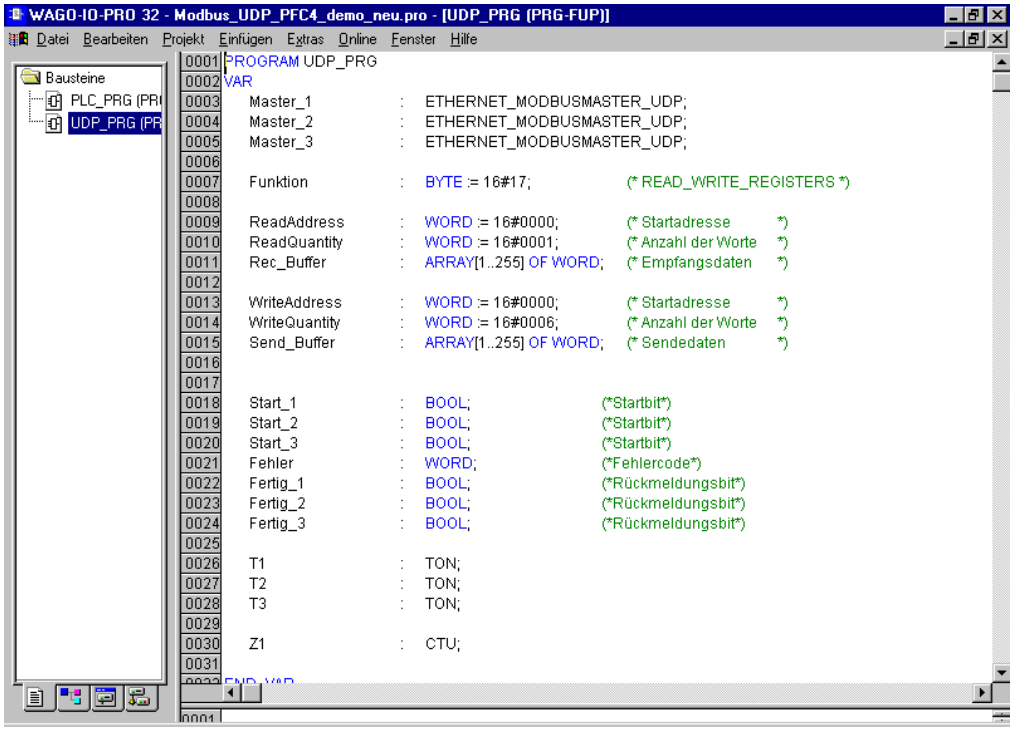
Therefore it is necessary to program the same number of ETHERNET_MODBUSMASTER_UDPs as slaves. The function blocks are initialized by giving each of them a different name (Master_1, Master_2, ...).

Then, the master function blocks have to be parameterized in accordance with the data of the slaves (IP address, read area, write area,...).

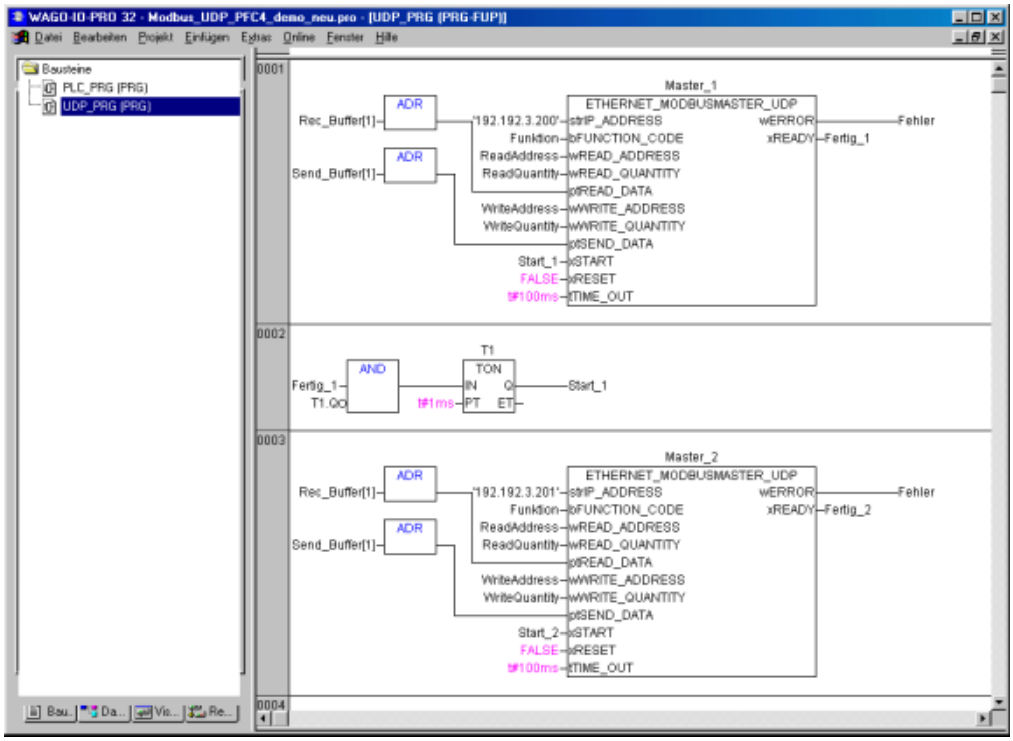
Attention
No overlaps must occur.

This condition might cause an increase of the network load (Ethernet) when data is exchanged permanently, i.e. without efficiency, resulting in a slower data communication.

Variable declaration



Initialization of the ETHERNET_MODBUSMASTER_UDP function blocks



The ETHERNET_MODBUSMASTER_TCP function blocks have to be programmed identically.

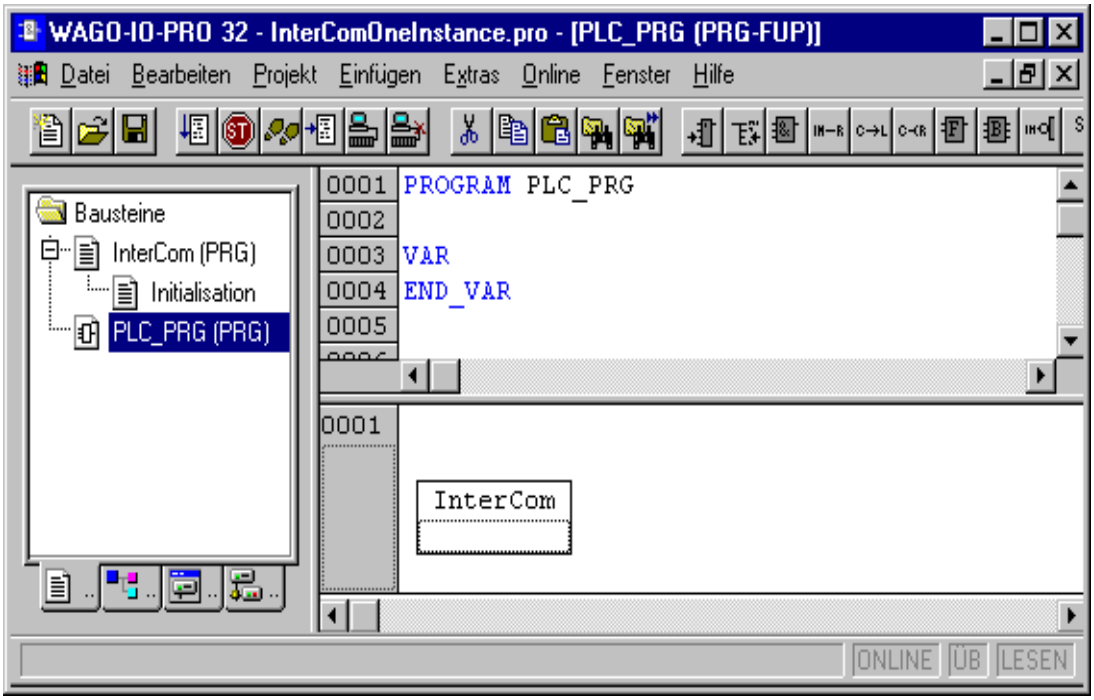
If needed, the TIME_OUT may have to be adjusted.

3.3 Data exchange with several MODBUS slaves

The following example describe a cycled data exchange between a ethernet-controller 750-842 as Modbus master and configurable number of Modbus slaves with only one instance of the function block “ETHERNETMODBUS-MASTER_UDP”.

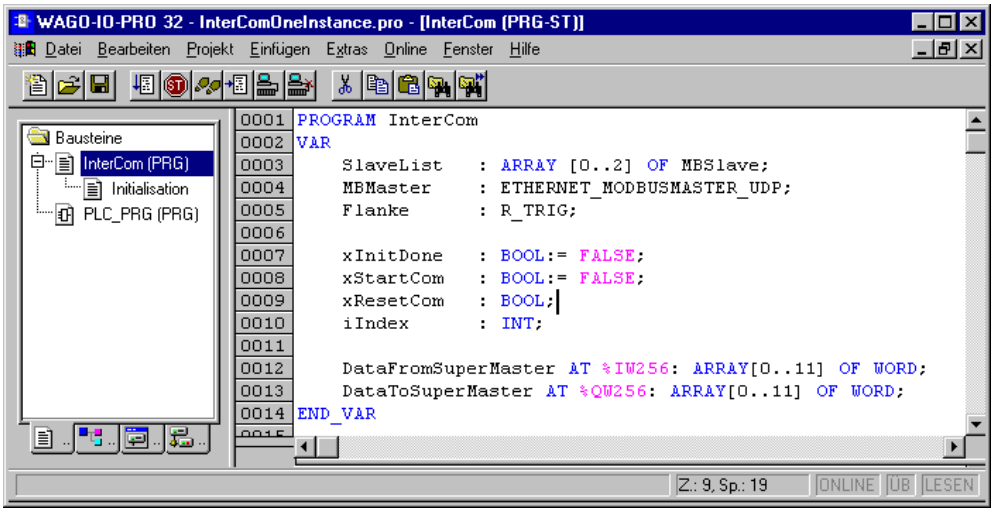
Every device that implements the Modbus-TCP(UDP) slave protocol can act as communication partner. In our case it should be a number of ethernet controllers or couplers.

The hole programcode necessary for the data exchange is capsulated in an seperate program called “InterCom”.



Existing programs can easily expanding they are functionality for data exchange by adding this program, and call the program In the PLC_PRG.

The processing of data exchange need only one instance of the function block „ETHERNET_MODBUSMASTER_UDP“. To do so, the parameters of the function block have to change after each data echange and restart the next data exchange.

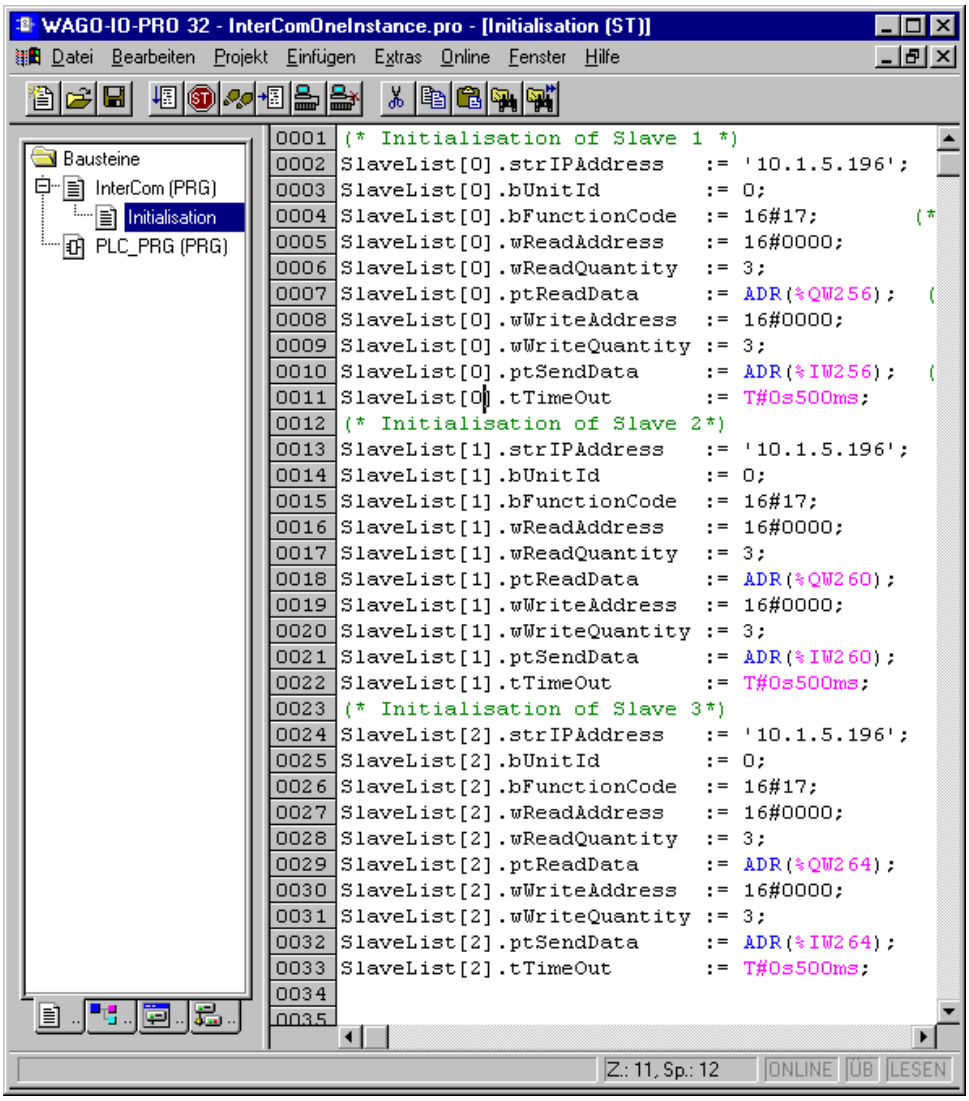


The necessary parameters for the function block are stored in the “SlaveList”.. This „SlaveList“ is an array of type „MBSlave“ and will be initialized on program start with the action „InterCom.Initialsation“.

```
TYPE MBSlave :
  STRUCT
    strIPAddress : STRING;           (* Address of Modbusslave *)
    bUnitId      : BYTE; (* Field "UnitId" in Modbusprotocolheader*)
    bFunctionCode: BYTE;             (* Modbus functioncode *)
    wReadAddress : WORD;             (* Read data from address  *)
    wReadQuantity: WORD;             (* Number of points to read  *)
    ptReadData   : POINTER TO BYTE;(* Pointer to readdata *)
    wWriteAddress: WORD;             (* Write data to address  *)
    wWriteQuantity: WORD;            (* Number of points to write  *)
    ptSendData   : POINTER TO BYTE;(* Pointer to writedata *)
    tTimeOut     : TIME;             (* max time for response from Modbusslave *)
  END_STRUCT
END_TYPE
```

The structure „MBSlave“ keep all the necessary parameters for one data exchange.

Following figure displays an example configuration for data exchange with three Modbus slaves.



To customizing the “SlaveList” to your requirements needs three steps:

1.) Change the array size of „SlaveList“ in program „InterCom“ at line 3.
2.) Edit the calculation of the next index in program „InterCom“ at line 50 to match with the size of „SlaveList“..
3.) Edit the parameters in „InterCom.Initialisation“.

4 Modbus/TCP- versus IEC1131-6 Addresses

Word-Operation:

Methode	Modbus Address	Modbus Address	IEC1131-6- Adressen	Description
FC3 – Read Multiple Register	0... 255	0x0000 – 0x00FF	%IW0... %IW255	physical INPUT's
	256... 511	0x0100 – 0x01FF	%QW256... %QW511	PFC-OUT-Variables
	512 ... 767	0x0200 – 0x02FF	%QW0... %QW255	physical OUTPUT's
	768 ... 1023	0x0300 – 0x03FF	%IW256... %IW511	PFC-IN-Variables
	4096... 4160	0x1000 – 0x1040	not supported	Configuration registers (see manual)
	8192 ... 8240	0x2000 - 0x2030	not supported	Firmware register (see manual)
	12288... 16384	0x3000 - 0x3FFF	%MW0... %MW4095	RETAIN-Variables
FC16 – Write Multiple Register	0... 255	0x0000 – 0x00FF	%QW0... %QW255	physical OUTPUT's
	256... 511	0x0100 – 0x01FF	%IW256... %IW511	PFC-IN-Variables
	4096... 4160	0x1000 – 0x1040	not supported	Configuration registers (see manual)
	12288... 16383	0x3000 - 0x3FFF	%MW0... %MW4095	RETAIN-Variables

Bit-Operation:

Methode	Modbus Address	Modbus Address	IEC1131-6-- Address	Description
FC2 - Read Input Discret FC1 – Read Coils = FC2 + $200	0... 511	0x0000 – 0x01FF	%IX(DigitalOffSet + 0).0 ... %IX(DigitalOffSet + 31).15	physical INTPUT's
	512... 1023	0x0200 – 0x03FF	%QX(DigitalOffSet + 0).0 ... %QX(DigitalOffSet + 31).15	physical OUTPUT's
	Illegal Address	Illegal Address	%I/QX(DigitalOffSet + 32).0 ... %I/QX(255 - DigitalOffSet).15	phys.IN/OUT
	4096... 8191	0x1000 – 0x1FFF	%QX256.0 ... %QX511.15	PFC-OUT-Variables
	8192... 12287	0x2000 – 0x2FFF	%IX256.0 ... %IX511.15	PFC-IN-Variables
	12288... 32767	0x3000 - 0x7FFF	%MX0.0 .. %MX1274.15	RETAIN-Variables
	not supported	not supported	%MX1275.0 .. %MX4095.15	
FC15- - Force Multiple Coils	0... 511	0x0000 – 0x01FF	%QX(DigitalOffSet + 0).0 ... %QX(DigitalOffSet + 31).15	physical OUTPUT's
	512... 1023	0x0200 – 0x03FF	%QX(DigitalOffSet + 0).0 ... %QX(DigitalOffSet + 31).15	
	Illegal Address	Illegal Address	%QX(DigitalOffSet + 32).0 ... %QX(255 - DigitalOffSet).15	
	4096... 8191	0x1000 – 0x1FFF	%IX256.0 ... %IX511.15	PFC-IN-Variables
	8192... 12287	0x2000 – 0x2FFF		
	12288... 32767	0x3000 - 0x7FFF	%MX0.0 .. %MX1274.15	RETAIN-Variables
	not supported	not supported	%MX1275.0 .. %MX4095.15	

4.1 Retain Variables

From firmware version V2.2 and later, the retain variables are stored in the MODBUS address range 16#3000 – 16#3199.
These addresses can be accessed via a SCADA system for example.

In the controller these variables can be accessed via %MW0 (equals MODBUS address 16#3000), %MW1, ...).
The non-volatile data is stored in a NOVRAM and remains unchanged after a power failure.

Note
No variables with the following syntax may be defined, when using the addressing over the MODBUS address 16#3000) or %MW1:

```
VAR RETAIN
        rem1:INT;       (*First Retain variable*)
END_VAR.
```

The system stores these variables in the same address area, which can lead to address conflicts.

WAGO Kontakttechnik GmbH
Postfach 2880 • D-32385 Minden
Hansastraße 27 • D-32423 Minden
Phone: 05 71/8 87 – 0
Telefax: 05 71/8 87 – 1 69
E-Mail: info@wago.com

Internet: http://www.wago.com