ADDI-DATA®
PARTNER FÜR PRÄZISION

Instruction Manual

Intelligent Ethernet I/O systems (Part 1)

MSX-Exxxx and PLC: Principles

DIN EN ISO 9001:2015 certified

Edition: 02.04-09/2018

Product information

This manual contains the technical installation and important instructions for correct commissioning and usage, as well as production information according to the current state before printing.
The content of this manual and the technical product data may be changed without prior notice.
ADDI-DATA GmbH reserves the right to make changes to the technical data and the materials included herein.

Warranty and liability

The user is not authorised to make changes to the product beyond the intended use, or to interfere with the product in any other way.
ADDI-DATA shall not be liable for obvious printing and phrasing errors. In addition, ADDI DATA, if legally permissible, shall not be liable for personal injury or damage to materials caused by improper installation and/or commissioning of the product by the user or improper use, for example, if the product is operated despite faulty safety and protection devices, or if notes in the operating instructions regarding transport, storage, installation, commissioning, operation, thresholds, etc. are not taken into consideration. Liability is further excluded if the operator changes the product or the source code files without authorisation and/or if the operator is guilty of not monitoring the permanent operational capability of working parts and this has led to damage.

Copyright

ADDI-DATA software product licence

Please read this licence carefully before using the standard software. The customer is only granted the right to use this software if he/she agrees with the conditions of this licence.
The software may only be used to set up the ADDI-DATA products.
Reproduction of the software is forbidden (except for back-up and for exchange of faulty data carriers). Disassembly, decompilation, decryption and reverse engineering of the software are forbidden. This licence and the software may be transferred to a third party if this party has acquired a product by purchase, has agreed to all the conditions in this licence contract and the original owner does not keep any copies of the software.

Trademarks

- ADDI-DATA, APCI-1500, MSX-Box and MSX-E are registered trademarks of ADDI-DATA GmbH.
- Turbo Pascal, Delphi, Borland C, Borland C++ are registered trademarks of Borland Software Corporation.
- Microsoft .NET, Microsoft C, Visual C++, MS-DOS, Windows XP, Windows 7, Windows 8, Windows 10, Windows Server 2000, Windows Server 2003, Windows Embedded and Internet Explorer are registered trademarks of Microsoft Corporation.
- LabVIEW, LabWindows/CVI, DASYLab, DIAdem are registered trademarks of National Instruments Corporation.
- CompactPCI is a registered trademark of PCI Industrial Computer Manufacturers Group.
- VxWorks is a registered trademark of Wind River Systems, Inc.
- RTX is a registered trademark of IntervalZero.
- Mozilla Firefox is a registered trademark of Mozilla Foundation.
- SIMATIC S7, S7-300, STEP7, TIA Portal, CPU313C-2DP and CP343-1 Lean are registered trademarks of Siemens AG.

Warning!

The following risks result from the improper implementation of the Ethernet system and from use contrary to the regulations:

Personal injury

Damage to the Ethernet system, the PC and peripherals

Pollution of the environment.

- Protect yourself, others and the environment!
- Read the safety precautions (yellow leaflet) carefully!

  If this leaflet is not enclosed with the documentation, please contact us and ask for it.
- Observe the instructions of this manual!

  Make sure that you do not forget or skip any step!
  We are not liable for damages resulting from the wrong use of the Ethernet system.
- Pay attention to the following symbols:

NOTICE!
Designates hints and other useful information.

NOTICE!
Designates a possibly dangerous situation.
If the instructions are ignored, the Ethernet system, the PC and/or peripherals may be **destroyed**.

WARNING!
Designates a possibly dangerous situation.
If the instructions are ignored, the Ethernet system, the PC and/or peripherals may be **destroyed** and persons may be **endangered**.

Contents

Figures

1 Documentation

1.1 Content

This documentation describes how to use intelligent Ethernet I/O systems, Ethernet and a standard protocol such as TCP/IP to carry out complex and fast measuring tasks or signal processing (such as average value computation) in a simple, reliable and cost-effective way via a PLC.

More detailed information on the respective MSX-E systems[1] can be found in the ADDI-DATA product catalog and in the corresponding manuals.

1.2 Structure

The documentation is divided into two parts:

- **Part 1: MSX-Exxxx and PLC: Principles**
  The first part of the documentation describes the principles of combining a PLC with an MSX-E system.
- **Part 2: Connection to a SIMATIC S7 PLC**
  Part 2 answers the following questions:
  Which requirements are necessary to connect an MSX-E system to a PLC (hardware, etc.)?
  How are PLC and MSX-E system configured?
  Which settings generally have to be made – and for what reason?
  What do I have to be aware of when connecting one (or more) MSX-E systems to a PLC?
- **Part 3: ADDI-DATA Modbus TCP client library for a SIMATIC S7 PLC**
  Part 4 describes how the MSX-E systems can be directly managed and controlled by a PLC using a Modbus TCP client library.

[1] **MSX-Exxxx** = Common name for the Ethernet I/O systems

2 The combination principle

2.1 More powerful PLC using intelligent Ethernet I/O systems

High flexibility and good value components have helped the PLC concept to reach its triumphant success. Alongside control and regulation, PLCs are increasingly taking over other tasks such as alarms, visualisation and data logging. Sensors and actuators ceased to be discrete some time ago, but can be combined directly with the PLC at reduced wiring costs. Ethernet networks have proven themselves in office IT for a long time now. The link between office IT and manufacturing IT meets the information, monitoring and security requirements of modern management simultaneously.

Working in series makes programmable logic controllers slower than hardwired programmed logic controllers. But demanding tasks such as the simultaneous reading of several channels or the simultaneous acquisition of several signal types which originate from inductive/digital transducers, shaft encoders or analog inputs are carried out using high-performance electronic measuring equipment. Examples of this are an acquisition of many measurement values in a buffer in order to make data available later or a very rapid acquisition and computation of values independent of the PLC cycle.

2.2 Requirements regarding electronic measuring equipment of the MSX-E system

- Simultaneous acquisition of different measurement values (e.g. surface quality and position)
  - ➢ Allows for precise corrections at detected fault locations
- Calculation of minimum and maximum values or average values
  - ➢ In order to categorise any irregularities within or outside a predefined tolerance or to relieve the load on the PLC
- Direct use in the production hall near the test item
- Reliable operation at high temperatures and with water jets
- Easy connection to the PLC
- Communication via standard Ethernet
  - ➢ In order to also forward values to the IT level for prompt assessment of production quality
- Configuration without programming, e.g. over an integrated web interface
  - ➢ Integration as easy as possible
- Diagnostics and monitoring without special programs or PLC
- Onboard cache so that no values are lost

2.3 Significant benefits of the MSX-E systems for the PLC

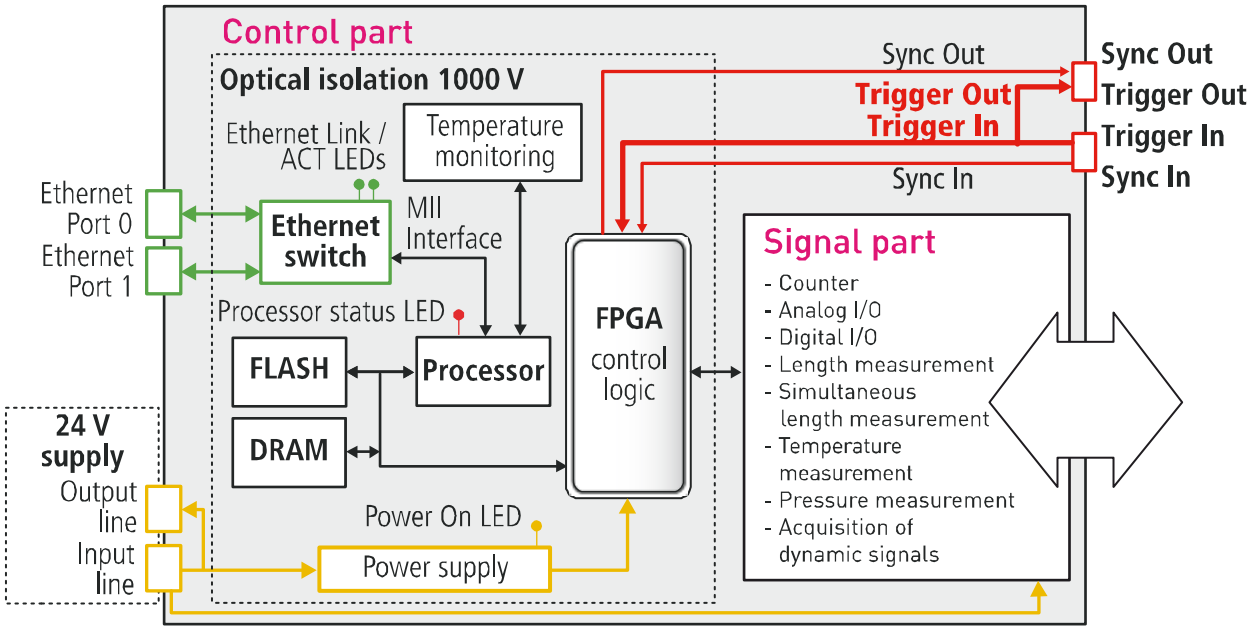
- They can be cascaded and synchronised in the µs area.
  - ➢ Simultaneous acquisition of different measurement values, e.g. inductive transducers and position of the test item
- Extended temperature range from -40 °C to +85 °C
- Degree of protection IP 65: protection against water jets and dust
  - ➢ No “protection” through switch cabinets necessary; money- and space-saving
- Numerous protective circuits: Optical isolation of up to 1,000 V; short-circuit and reverse polarity protection
- Web interface to configure, control and monitor the acquisition
  - ➢ Measurement values directly on the screen of the technician's PC
  - ➢ Saving of time, in particular when setting up measuring facilities, and flexibility
- Easy connection to the PLC via TCP/IP
  - ➢ Problem-free forwarding of measuring data to the IT level
  - ➢ Remote maintenance with password protection and encryption
- Onboard RAM for data storage
- Simple software updates can be installed at any time on the MSX-E systems.
- Complex computing tasks (such as calculating several sensors in one result) can be realised if required.

The MSX-E systems are available for different signal types:

- Digital I/O, 24 V
- Analog inputs, voltage or current, 16 bit
- Analog outputs, voltage or current, 16 bit
- Inductive transducers (HB, LVDT, Mahr-compatible)
- Incremental shaft encoders and transducers
- Sin/cos signals (1 V_{pp} / 11 μA_{pp}) for shaft encoders or digital transducers
- Etc.

The mode of operation is identical for all MSX-E systems, i.e. each system has a control side and a signal side. The connection to the control side is the same for all systems.

Fig. 2-1: MSX-E system: Structure

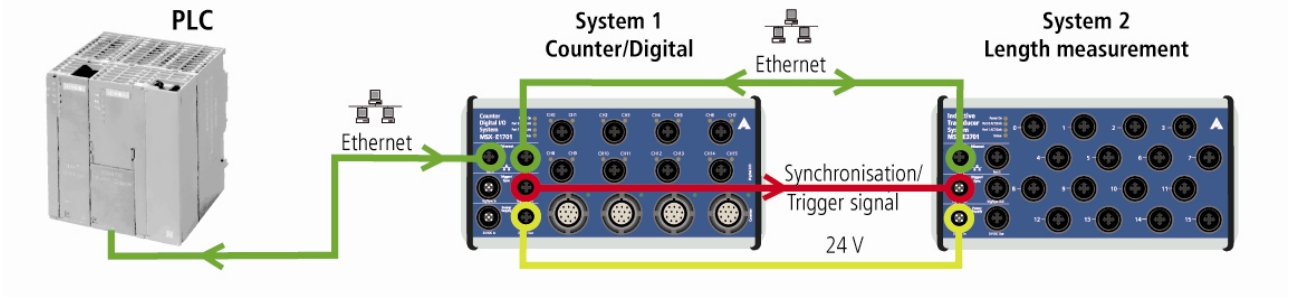


2.4 Connection of an MSX-E system to a PLC

The MSX-E systems are connected to the PLC via Ethernet. Thus no special lines or communication buses are needed. Additional systems can be added to the network very easily.

By means of an Ethernet controller, which manages the TCP/IP protocol, the PLC is able to access the additional systems. There are both hardware solutions (communication processor, CP) and software solutions for this, whereas a hardware solution may be more suitable, depending on the application. A separate hardware controller enables faster communication and reduces the load on the PLC CPU more than software Ethernet management over the PLC CPU.

Fig. 2-2: MSX-E systems: Connection to a PLC



2.5 The connection of an MSX-E system and a PLC

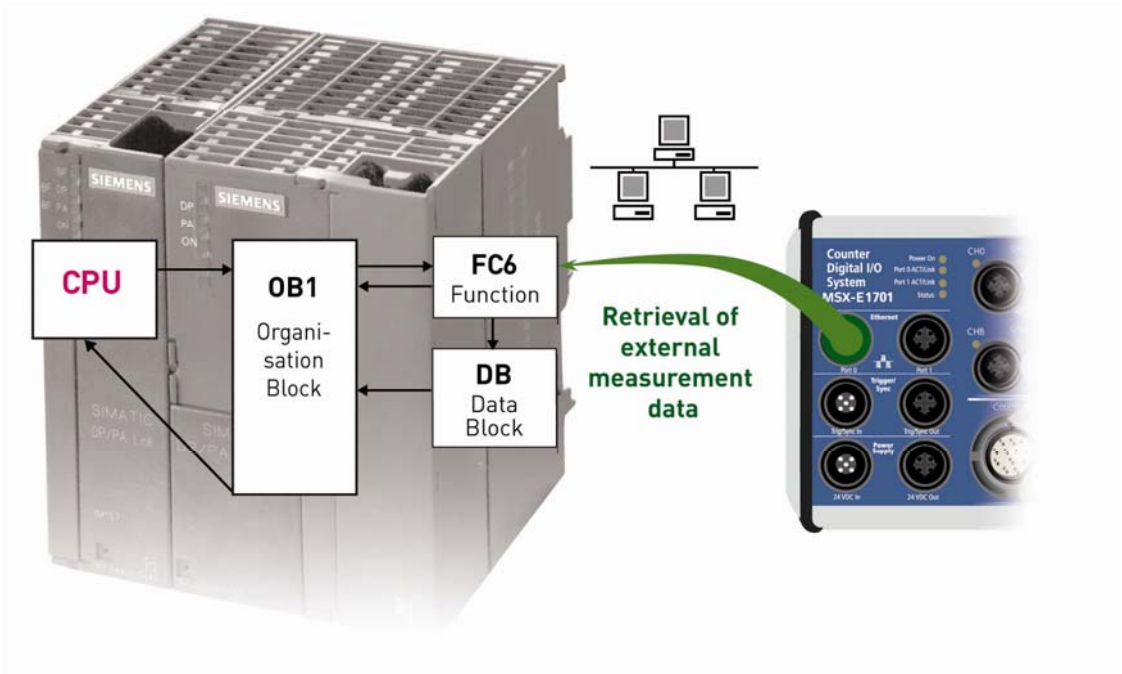
2.5.1 Acquisition start without programming (Autostart)

The actual acquisition is configured and saved on the MSX-E systems using the web interface. You can use the Autostart function to load the acquisition settings when the system is booted and to execute them automatically. As such, no additional programming is required for the PLC.

Example: 16 analog inputs (8 x ± 10 V, 8 x ± 5 V), each with a speed of 25 kHz, should be acquired for a total of 0.5 seconds. The acquisition should only be started once the test item is in the correct position, that is, the PLC signals the start of the acquisition to the MSX-E system using a digital 24 V output (hardware trigger).
If the acquisition has been previously configured and saved in the MSX-E system, the system boots up after it has been switched on and waits for the hardware trigger. As soon as the hardware trigger has been detected, the acquisition starts and the system acquires the measurement values for 0.5 seconds. It subsequently waits for the next trigger.

Via the receive function (FC6), the PLC can access the values and save these in a data block (DB) (1 DWORD per channel). Afterwards, the values can be further processed as required. Due to their own intelligence (ARM9), the systems can carry out calculations, e.g. of the minimum, maximum and average values, independently.

Fig. 2-3: PLC: Retrieval of external measurement data



The configuration of PLC and MSX-E systems is described in detail in **Part 2** of the Ethernet I/O systems' **Instruction Manual**.

2.5.2 Acquisition start with TCP/IP Modbus

The acquisition of the measurement values can also be started using TCP/IP Modbus. Here, the data acquisition is not configured in the MSX-E systems and loaded when they are booted; instead, they are called in the PLC through Modbus functions.

More detailed information on this can be found in **Part 3** of the Ethernet I/O systems' **Instruction Manual** as well as in the **MODBUS Interface Description** on the MSX-E CD ("msxexxxx_modbus.html" file in the directory "\[MSX-E system]\Modbus\Documentation\html").

3 Appendix

3.1 Glossary

Counter
A counter is a circuit which counts pulses or measures pulse duration.

Ethernet
The Ethernet is a baseband bus system originally developed in order to connect mini-computers. It is based on the CSMA/CD access method. Coaxial cables or twisted-pair cables are used as the transmission medium.
The transmission speeds are 10 Mbit/s (Ethernet), 100 Mbit/s (Fast Ethernet) and 1 Gbit/s or 10 Gbit/s (Gigabit-Ethernet).
This widely used technology for computer networking in a LAN has been standardised since 1985 (IEEE 802.3 and ISO 8802-3).
Ethernet technology is now common practice in the office environment. After making even very tough real-time requirements possible and adapting the device technology (bus cables, patch fields, junction boxes) to the harsh application conditions of the industrial environment, Ethernet is now also increasingly used in the field areas of automation technology.

IP address
The IP address is a unique string of numbers separated by full stops that identifies each computer attached to the Internet. Usually, it also has a version containing words separated by full stops.

LSB
= Least Significant Bit

LSB is the lowest order bit in a digital quantity.

Modbus TCP
The Modbus TCP protocol is an open query/response protocol for the communication between master and slave or client and server. A PLC, for example, can act as a master that initialises the communication processes.
The data is transferred in the form of TCP/IP packets.

MSB
= Most Significant Bit

MSB is the highest order bit in a digital quantity.

PLC
= Programmable Logic Controller

The PLC is a computer-based control unit whose functionality is defined by an application program. With standardised technical languages, this application program is relatively easy to produce. Because of its serial mode of operation, reaction times of PLCs are slower than those of VPS. As a family of devices with graduated and matched components, PLCs can now cover all levels of an automation hierarchy.

Socket
A socket is a bidirectional software interface to interprocess (IPC) or network communication. Sockets are a standardised interface (API) between the network protocol implementation of the operating system and the actual application software.

Trigger
A trigger is a pulse or signal for starting or stopping a special task. Triggers are often used for controlling data acquisition.

3.2 Index

4 Contact and support

Do you have any questions? Write or call us:

Address: ADDI-DATA GmbH
Airpark Business Center
Airport Boulevard B210
77836 Rheinmünster
Germany

Phone: +49 7229 1847-0
Fax: +49 7229 1847-222

E-mail: info@addi-data.com

Manual and software download from the Internet:

www.addi-data.com

ADDI-DATA®
PARTNER FÜR PRÄZISION

Instruction Manual

Intelligent Ethernet I/O systems (Part 2)

Connection to a SIMATIC® S7® PLC

DIN EN ISO 9001:2015 certified

Edition: 02.03-09/2018

Product information

This manual contains the technical installation and important instructions for correct commissioning and usage, as well as production information according to the current state before printing.
The content of this manual and the technical product data may be changed without prior notice.
ADDI-DATA GmbH reserves the right to make changes to the technical data and the materials included herein.

Warranty and liability

The user is not authorised to make changes to the product beyond the intended use, or to interfere with the product in any other way.
ADDI-DATA shall not be liable for obvious printing and phrasing errors. In addition, ADDI DATA, if legally permissible, shall not be liable for personal injury or damage to materials caused by improper installation and/or commissioning of the product by the user or improper use, for example, if the product is operated despite faulty safety and protection devices, or if notes in the operating instructions regarding transport, storage, installation, commissioning, operation, thresholds, etc. are not taken into consideration. Liability is further excluded if the operator changes the product or the source code files without authorisation and/or if the operator is guilty of not monitoring the permanent operational capability of working parts and this has led to damage.

Copyright

ADDI-DATA software product licence

Please read this licence carefully before using the standard software. The customer is only granted the right to use this software if he/she agrees with the conditions of this licence.
The software may only be used to set up the ADDI-DATA products.
Reproduction of the software is forbidden (except for back-up and for exchange of faulty data carriers). Disassembly, decompilation, decryption and reverse engineering of the software are forbidden. This licence and the software may be transferred to a third party if this party has acquired a product by purchase, has agreed to all the conditions in this licence contract and the original owner does not keep any copies of the software.

Trademarks

- ADDI-DATA, APCI-1500, MSX-Box and MSX-E are registered trademarks of ADDI-DATA GmbH.
- Turbo Pascal, Delphi, Borland C, Borland C++ are registered trademarks of Borland Software Corporation.
- Microsoft .NET, Microsoft C, Visual C++, MS-DOS, Windows XP, Windows 7, Windows 8, Windows 10, Windows Server 2000, Windows Server 2003, Windows Embedded and Internet Explorer are registered trademarks of Microsoft Corporation.
- LabVIEW, LabWindows/CVI, DASYLab, DIAdem are registered trademarks of National Instruments Corporation.
- CompactPCI is a registered trademark of PCI Industrial Computer Manufacturers Group.
- VxWorks is a registered trademark of Wind River Systems, Inc.
- RTX is a registered trademark of IntervalZero.
- Mozilla Firefox is a registered trademark of Mozilla Foundation.
- SIMATIC S7, S7-300, STEP7, TIA Portal, CPU313C-2DP and CP343-1 Lean are registered trademarks of Siemens AG.

Warning!

The following risks result from the improper implementation of the Ethernet system and from use contrary to the regulations:

Personal injury

Damage to the Ethernet system, the PC and peripherals

Pollution of the environment.

- Protect yourself, others and the environment!
- Read the safety precautions (yellow leaflet) carefully!

  If this leaflet is not enclosed with the documentation, please contact us and ask for it.
- Observe the instructions of this manual!

  Make sure that you do not forget or skip any step!
  We are not liable for damages resulting from the wrong use of the Ethernet system.
- Pay attention to the following symbols:

NOTICE!
Designates hints and other useful information.

NOTICE!
Designates a possibly dangerous situation.
If the instructions are ignored, the Ethernet system, the PC and/or peripherals may be **destroyed**.

WARNING!
Designates a possibly dangerous situation.
If the instructions are ignored, the Ethernet system, the PC and/or peripherals may be **destroyed** and persons may be **endangered**.

Contents

Figures

Tables

Chapter overview

In this manual, you will find the following information:

Chapter	Content
1	Principles of the MSX-E system and PLC configuration
2	Description of the PLC parameterisation or Ethernet configuration
3	Explanation of the S7 function blocks
4	Description of the S7 sample networks
5	Recommendations in terms of acquisition mode, trigger, etc.
6	Appendix with glossary and index
7	Contact and support address

1 MSX-E system and PLC configuration

1.1 Requirement

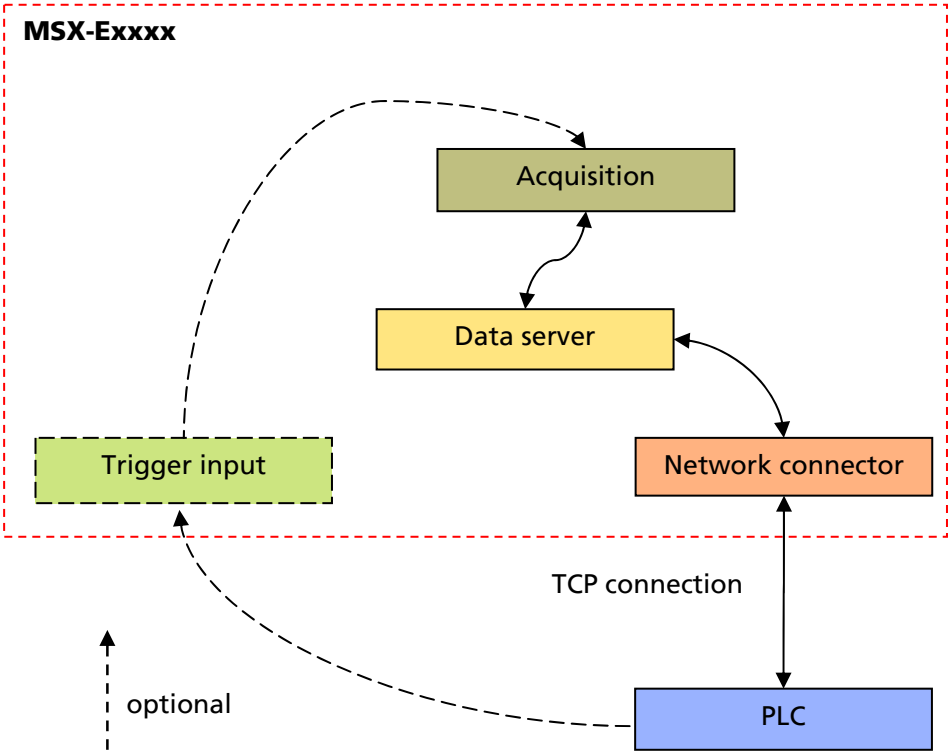
In order to use MSX-E systems with a PLC, it is required that the PLC has an Ethernet interface and can open an active TCP connection (TCP sockets).

1.2 Principle structure

Each MSX-E system has a data server that provides the acquired measurement values. The PLC can access these measurement values using a TCP connection, thereby reading the data.

The acquisition itself can be started either by an external trigger (digital 24 V input) or automatically.

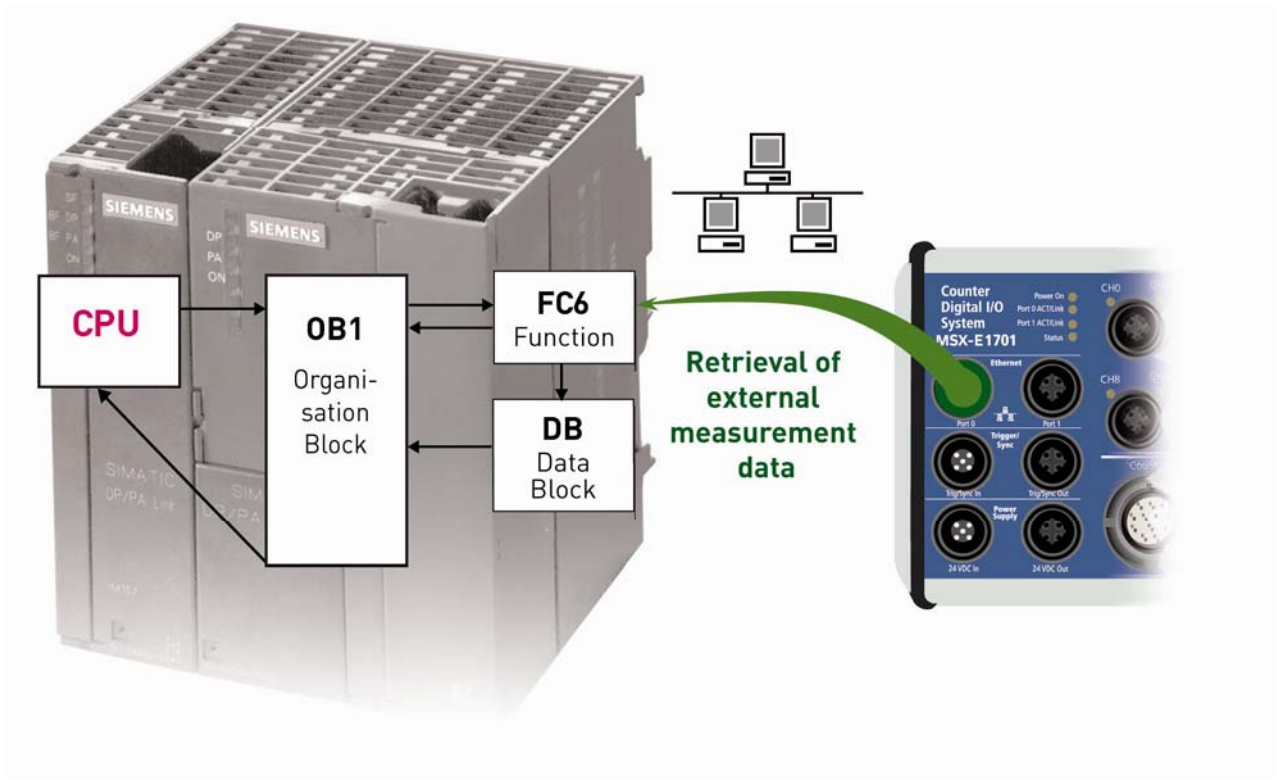
Fig. 1-1: MSX-E system and PLC: Principle structure



In the PLC, the data is read using the FC6 function (receive function) and then stored in a data block (DB). From here, it can be further processed as required.

1.3 Program overview

Fig. 1-2: Retrieval of external measurement data



OB1: This organisation block calls up the receive function (FC6).

FC6: With this function, data is read from a TCP connection (socket) and subsequently saved in a data block.

1.4 Data conversion

Please note that the MSX-E systems send data from the data server in the Little Endian format (Intel format), whereas an **S7 PLC** works with the Big Endian format (Motorola format).
As a result, the values of the **MSX-E systems** have to be converted on the PLC into the Big Endian format (Motorola format). This data coding plays an important role, as otherwise, the measurement results will be interpreted incorrectly. ADDI-DATA provides you with a corresponding conversion function:

- **Function:** FC1
- **Name:** INTEL_TO_MOTOROLA

1.4.1 Little Endian format (Intel format

The least significant bit (LSB) is transferred first:

16-bit	0x1234	>>	0x34	0x12		
32-bit	0x12345678	>>	0x78	0x56	0x34	0x12

1.4.2 Big Endian Format (Motorola Format

The most significant bit (MSB) is transferred first:

16-bit	0x1234	>>	0x12	0x34		
32-bit	0x12345678	>>	0x12	0x34	0x56	0x78

2 Parameterising the PLC

2.1 Software

ADDI-DATA provides you with a software package that displays the connection between an MSX-E system and an S7 PLC. The programming sample is based on a SIMATIC **S7** PLC from Siemens.

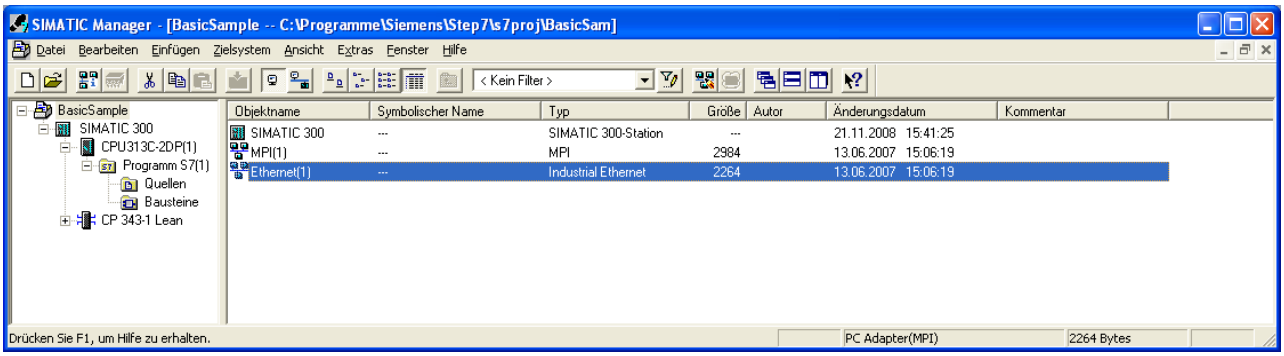
If you want to apply an **S7**-compatible PLC from a different manufacturer (such as IBHsoftec, VIPA, etc.), you have to use the relevant manufacturer's original FCs. You may also have to adapt the data blocks, etc. (Siemens: 9 DWORDs; VIPA: 36 bytes).

The mentioned sample for the **S7** from Siemens is called **BasicSample** and has been developed for the Ethernet connection of the **S7-300** with **CP343-1**. It serves for reading values from the TCP connection (TCP socket).

2.2 S7 Ethernet configuration

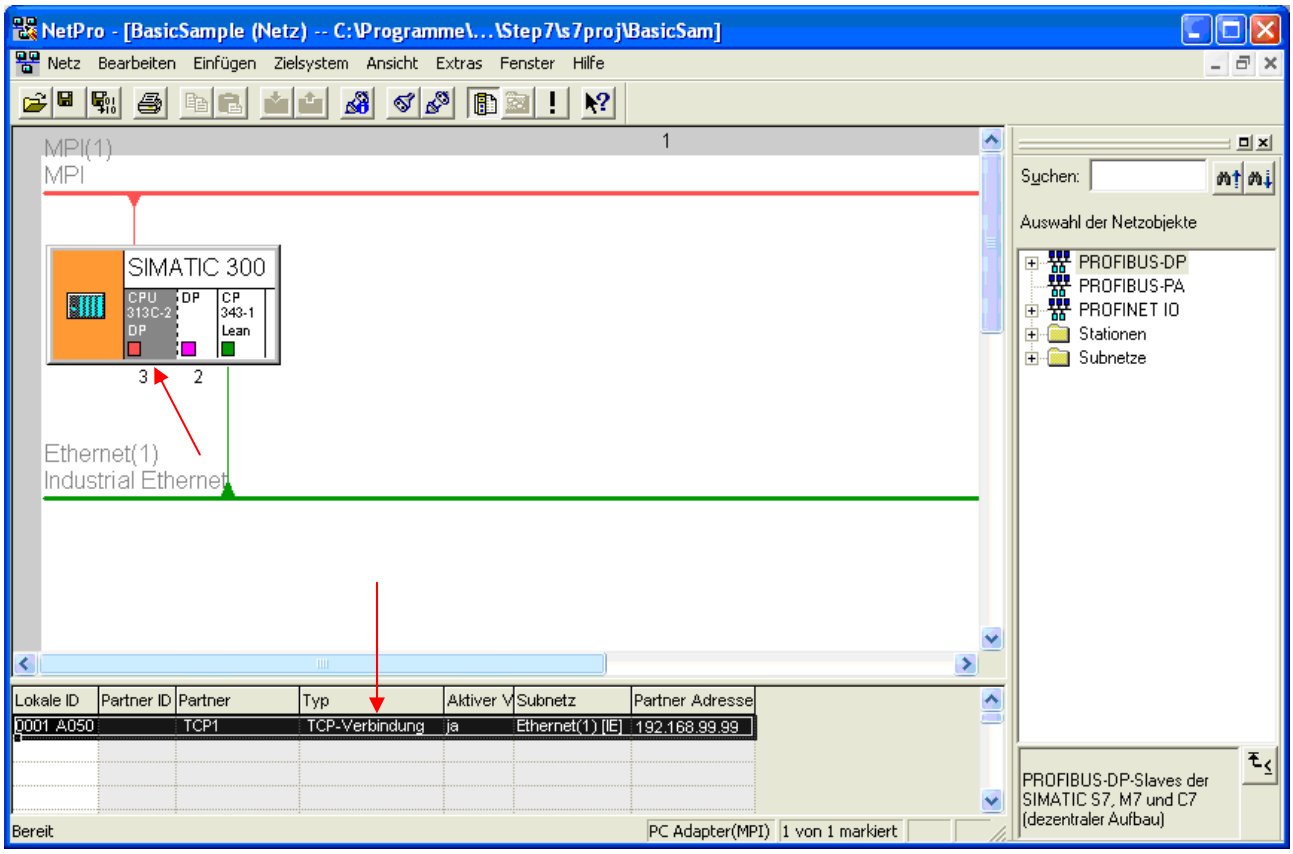
The following configuration is based on an **S7-300** with **CP343-1**. Your PLC may display other menus, but the settings are the same.

Fig. 2-1: SIMATIC Manager: S7 Ethernet configuration



- Open the SIMATIC Manager and then the **BasicSample** project.
- To configure the TCP connection, double-click on the object name (Objektname) **Ethernet**.

Fig. 2-2: Ethernet configuration: TCP connection

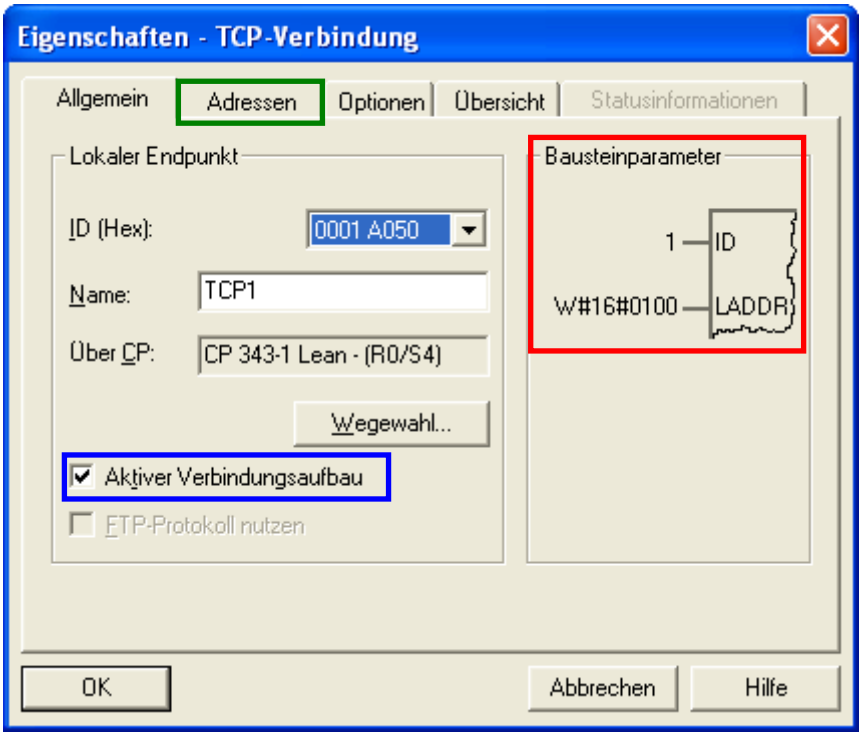


- Click on **CPU313C-2DP** (see top arrow).
- In the table at the bottom, double-click on the selected line (see bottom arrow).

The table column **Typ** [Type] says that the Ethernet connection is a **TCP connection** (TCP-Verbindung).

Afterwards, the following window is displayed:

Fig. 2-3: Properties – TCP connection: General information

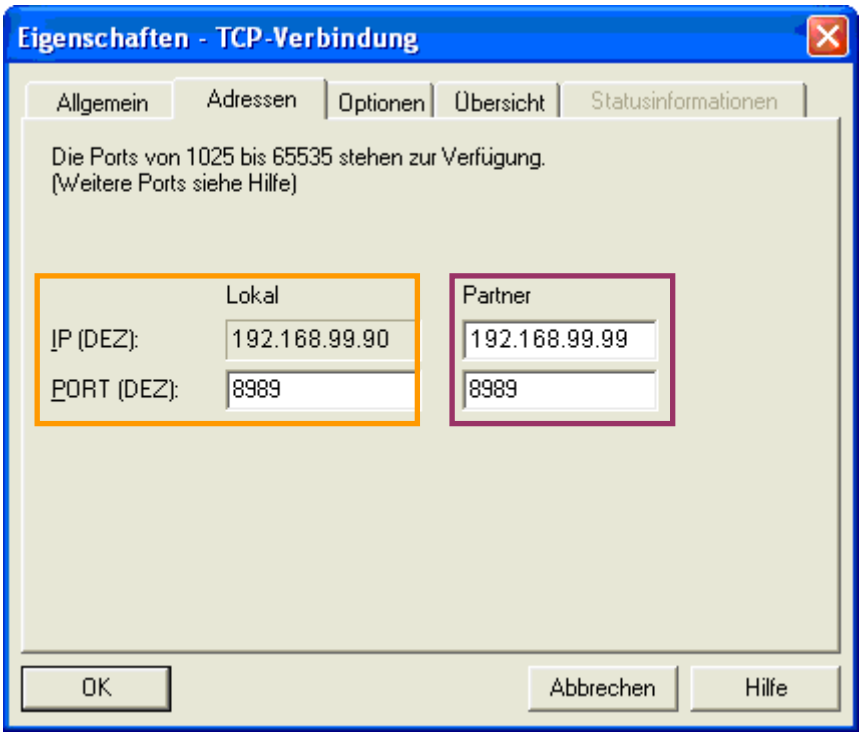


In order for the **S7** to create the connection to the MSX-E system, **Aktiver Verbindungsaufbau** [Active connection establishment] must be selected.

On the right-hand side of the window, you see the block parameters **(Bausteinparameter)** that have to be used together with the "FC6 (AG_RECV)" function. The connection ID (socket ID) and the address are specified.

- Click on the **Adressen** [Addresses] tab.

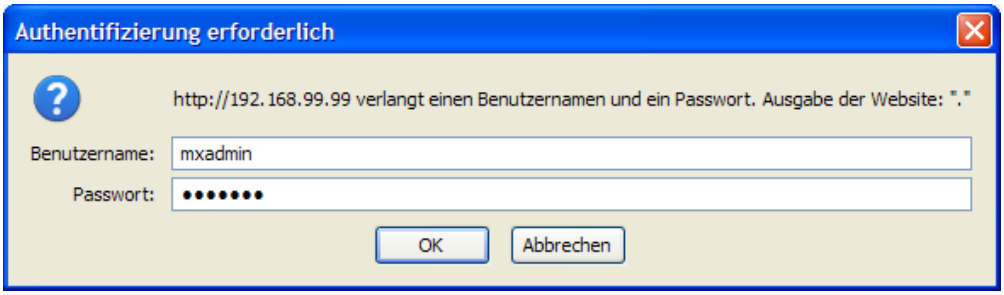
Fig. 2-4: Properties – TCP connection: Addresses



- In the **Lokal** [Local] area, in the **PORT** field, enter the port number of the **S7**. This must correspond to the port number of the MSX-E system.
- In the **Partner** [Remote] area, in the **IP** field, enter the IP address of the **MSX-E system**, and in the **Port** field, the port number of the MSX-E system. This port number must correspond to the one that has been defined on the web interface of the MSX-E system.
- To check the port number of the MSX-E system, open a web browser (such as Mozilla Firefox, Internet Explorer, etc.) and enter the following address: "http://[IP address of the MSX-E system]".

A login window is displayed:

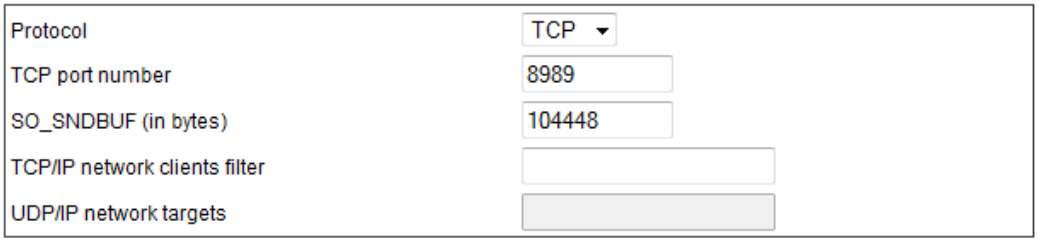
Fig. 2-5: Login window



- Enter **mxadmin** as the user name and password.

- On the web interface of the MSX-E system, click on the menu item **Data server**.

Fig. 2-6: Data server: Configuration



- In the **Configuration** section, in the **TCP port number** field, change the port number if you cannot use the default port number (8989).

Fig. 2-7: Data server: Set and save

- After changing the port number, click on **Set and Save** and then on **Reload**.
- Click on the **Blocking transfer** tab.

Fig. 2-8: Data server: Blocking transfer

- Ensure that in the **Configuration** section, with **Activate blocking TCP/IP transfer**, the **No** option is selected.
- When you have made this change, click on **Set and Save** and then on **Reload**.

3 Reading the measurement data

As already mentioned in Chapter 2, ADDI-DATA provides you with a programming sample of an S7 PLC from Siemens. The aim of this is sample is to demonstrate how the PLC reads the measurement data of an MSX-E system from a TCP connection.

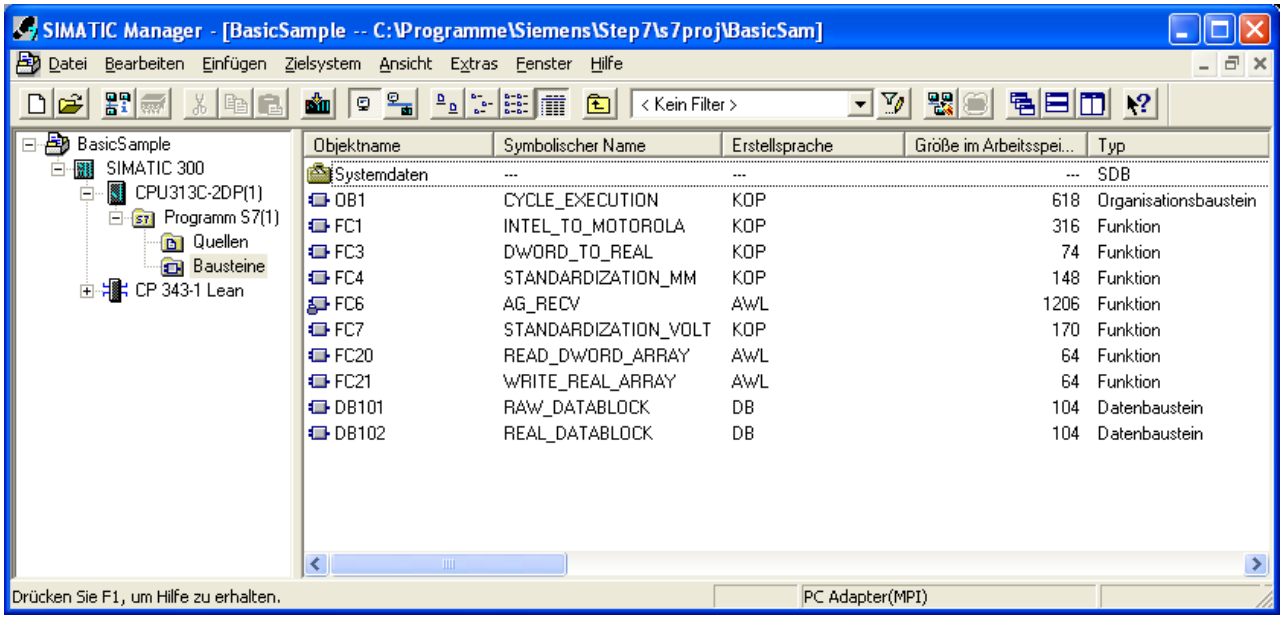
3.1 Requirements

Please ensure that the following requirements are fulfilled:

- Siemens CPU313C-2DP (PLC)
- Siemens CP343-1 Lean (Ethernet system for the PLC)
- FC6 (AG_RECV) for S7-300 (function for asynchronous communication)
- S7 sample for STEP7.

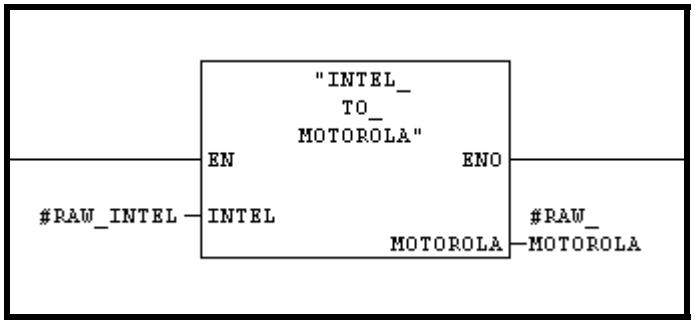
3.2 S7: Description of the function blocks

Fig. 3-1: SIMATIC Manager: List of blocks



3.2.1 FC1: Converting data to the Motorola format

Description:
With the MSX-E systems, the data is displayed in the Intel format, whereas the S7 PLC displays it in the Motorola format.
If your PLC already uses the Intel format, you do not need this function block. However, if your PLC works in the Motorola format, you can use this block to convert the values.



Input parameters:

INTEL: Raw value in the **Intel format** as a DWORD data type that is read from the MSX-E system

Output parameters:

MOTOROLA: Value in the Motorola format as a DWORD data type

Example of the sequence of function calls:

Table 3-1: FC1: Function call sequence

FC number	Name	Description
FC6	**AG_RECV**	Read value from the network
FC1	**INTEL_TO_MOTOROLA.**	Convert value to the Motorola format

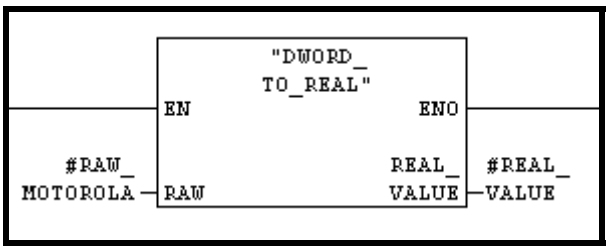
3.2.2 FC3: Converting a DWORD data type to a REAL data type

Description:
On the MSX-E systems the measurement values are coded to 4 bytes and sent via the TCP connection as byte packets. By default the measurement values are sent as raw values.

Example: System MSX-E3011 with Gain 1 Unipolar
If 10 volts are measured, the system does not send the value 10, but 65535. This value is designated as a raw value.

65535 is a raw value that is converted to the suitable unit:
((Voltage range/resolution) x raw value) + minimum voltage
((20 / 65535) x 65535) + (-10) = 10 volts

If the systems convert the raw values to the appropriate unit immediately and send them, no additional conversion is required on the PLC. The values that are read from the TCP connection should first be displayed correctly and then be converted with a type cast to the REAL data type. The type cast is required, because the values that are received cannot be directly sent and received as a REAL type.



Input parameters:

RAW: Raw value that has been converted by the system to the appropriate unit, as a DWORD data type in the **Motorola format**

Output parameters:

REAL_VALUE: Type cast value as a REAL data type

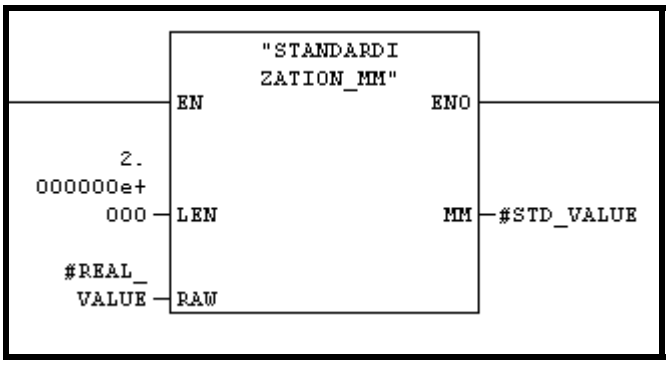
Example of the sequence of function calls:

Table 3-2: FC3: Function call sequence

FC number	Name	Description
FC6	**AG_RECV**	Read value from the network
FC1	**INTEL_TO_MOTOROLA.**	Convert value to the Motorola format
FC3	**DWORD_TO_REAL**	Display value as a REAL type (type cast)

3.2.3 FC4: Converting to millimetres (for transducers

Description:
This function can be used with the MSX-E370x and MSX-E3711 systems to convert raw values into millimetres.



Input parameters:

LEN: Value as a REAL data type that defines the transducer path in millimetres

RAW: Value as a REAL data type that is already displayed in the **Motorola format** (see **Functional call sequence**)

Output parameters:

MM: Measurement value as a REAL data type in millimetres

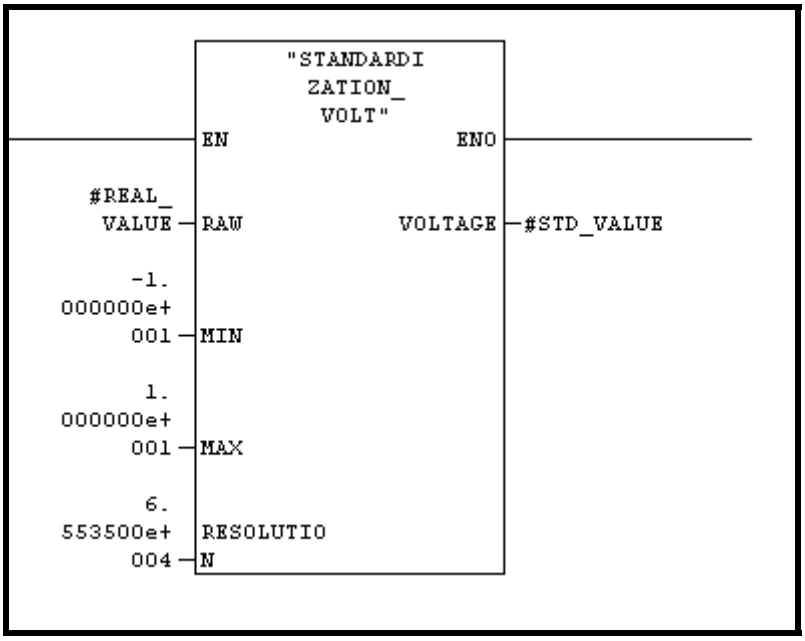
Example of the sequence of function calls:

Table 3-3: FC4: Function call sequence

FC number	Name	Description
FC6	**AG_RECV**	Read value from the network
FC1	**INTEL_TO_MOTOROLA.**	Convert value to the Motorola format
FC3	**DWORD_TO_REAL**	Display value as a REAL type (type cast)
FC4	**STANDARDIZATION_MM**	Convert value to millimetres

3.2.4 FC7: Converting to volts or mA

Description:
This function can be used with the MSX-E3011 system to convert raw values into volts.



Input parameters:

RAW: Value as a REAL data type that is already displayed in the **Motorola format** (see **Functional call sequence**)

MIN: Value as a REAL data type that defines the minimum voltage value of the voltage range.

Example: System MSX-E3011
With Gain 1 Bipolar, this value is equal to -10 volts. With Gain 2 Bipolar, this value is -5 volts.

MAX: Value as a REAL data type that defines the maximum voltage value of the voltage range.

Example: System MSX-E3011
With Gain 1 Bipolar, this value is equal to +10 volts. With Gain 2 Bipolar, this value is +5 volts.

RESOLUTION: Value as a REAL data type that defines the resolution of the measurement range.

Example: An MSX-E3011 system has a resolution of 16 bit (= 65535 steps).

Output parameters:

VOLTAGE: Measurement value as a REAL data type in volts

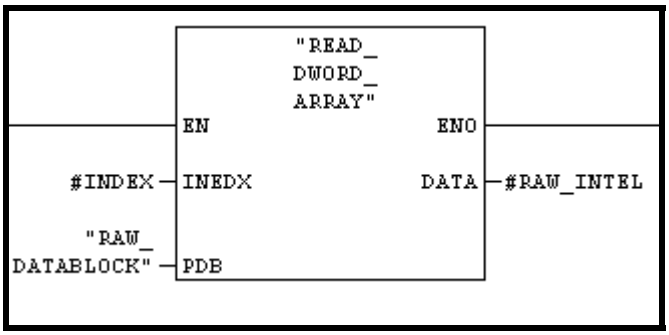
Example of the sequence of function calls:

Table 3-4: FC7: Function call sequence

FC number	Name	Description
FC6	**AG_RECV**	Read value from the network
FC1	**INTEL_TO_MOTOROLA.**	Convert value to the Motorola format
FC3	**DWORD_TO_REAL**	Display value as a REAL type (type cast)
FC7	**STANDARDIZATION_VOLT**	Convert value to volts

3.2.5 FC20: Dynamic Reading of a DWORD Value from a Table

Description:
This function block is not absolutely necessary. It enables a DWORD value to be read from a DWORD table (which is located in a data block). Instead of a fixed definition of the table index, the latter can be dynamically managed using this block. However, FC20 does not involve any error management (such as error = index > table width).



Input parameters:

INDEX: Index as an INT data type to be read from the table cell

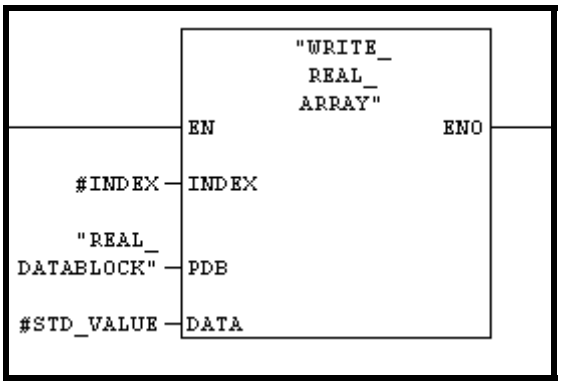
PDB: Data block in which the table is located as a DWORD data type

Output parameters:

DATA: The value that is read as a DWORD data type

3.2.6 FC21: Dynamic writing of a REAL value to a table

Description:
This function block is not absolutely necessary. It enables a REAL value to be written from a REAL table (which is located in a data block). Instead of a fixed definition of the table index, the latter can be dynamically managed using this block. However, FC21 does not involve any error management (such as error = index > table width).



Input parameters:

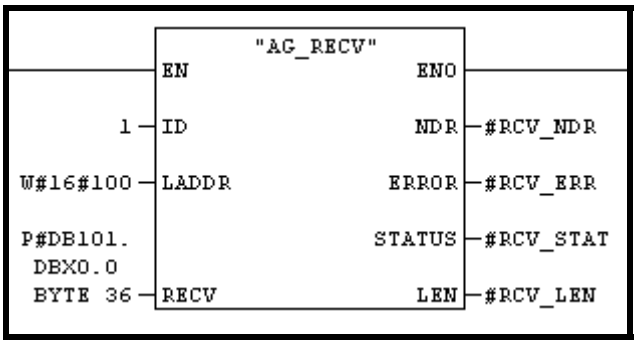
INDEX: Index as an INT data type to be written to the table cell

PDB: Data block in which the table is located as a REAL data type

DATA: Value to be written as a REAL data type

3.2.7 FC6: Reading data from a network connection

Description:
This function block has been created by Siemens and enables data to be read from a network connection.

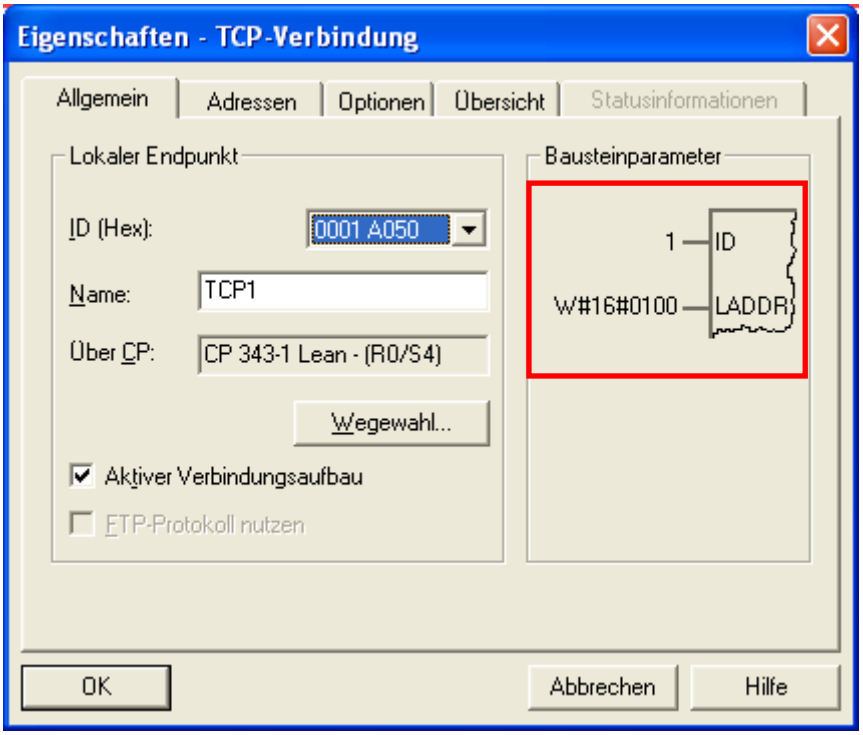


Input parameters:

ID: 1 is the number of the data server's communication identifier.

LADDR: The value "W#16#0100" is the address of the function block.

Fig. 3-2: Properties – TCP connection



RECV: Pointer in which the data has to be saved; in this case, in **DB101**.

P#DB101.DBX0.0 BYTE 36 means that a pointer to the offset 0 is passed to the DB101 data block (RAW_DATABLOCK) and that 36 bytes (9 DWORDs) are expected. The number of bytes expected may change as required. Ensure that the table in which the values are saved is sufficiently wide. The number of bytes always corresponds to a data packet.

The size of a data packet is specified on the web interface of the MSX-E system:

- On the web interface, click on the menu item **Acquisition**.
- Select the **Auto-Refresh** or **Sequence mode**.

In the **Data server frame format** section, the size of the data packet is displayed.

Fig. 3-3: Acquisition: Data server frame format

Size	Field	Description
4 Bytes	Inductive transducer 1	Digital value, encoded on 24 bits.
4 Bytes	Inductive transducer 6	Digital value, encoded on 24 bits.
4 Bytes	Inductive transducer 3	Digital value, encoded on 24 bits.

When the function block has received the values from the MSX-E system, you will find them in the following order in the table of the target data block:

RAW[0]: contains the value “counter”
RAW[1]: contains the value “channel 1”
RAW[2]: contains the value “channel 2”
…
RAW[9]: contains the value “channel 8”

The following example is based on an S7 screenshot **(Data server frame format)** and a data block containing the **RAW** table as a DWORD data type with a width of 36 bytes.

Fig. 3-4: Data block with the RAW table

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	RAW	ARRAY[0..8]		
*4.0		DWORD		
=36.0		END_STRUCT		

Fig. 3-5: RAW table in the data block

Address	Name	Type	Initial value	Actual value	Comment
0.0	RAW[0]	DWORD	DW#16#0	DW#16#6F8A0700	
4.0	RAW[1]	DWORD	DW#16#0	DW#16#BB06BF00	
8.0	RAW[2]	DWORD	DW#16#0	DW#16#8B447F00	
12.0	RAW[3]	DWORD	DW#16#0	DW#16#A1FF7F00	
16.0	RAW[4]	DWORD	DW#16#0	DW#16#EEFB7F00	
20.0	RAW[5]	DWORD	DW#16#0	DW#16#7EFE7F00	
24.0	RAW[6]	DWORD	DW#16#0	DW#16#FCFE7F00	
28.0	RAW[7]	DWORD	DW#16#0	DW#16#7DFF7F00	
32.0	RAW[8]	DWORD	DW#16#0	DW#16#B3028000	

Output parameters:

NDR: If the number of bytes has been read and NDR equals 1, no error has occurred. If NDR equals 0, then check ERROR and STATUS to find the error that is displayed.

LEN: Number of read bytes. The number must be 36 (= 9 x 4 bytes).

4 Simple S7 sample

Please configure Networks 1 to 3 according to the description in the following chapters.

4.1 Configuring the MSX-E system using the web interface

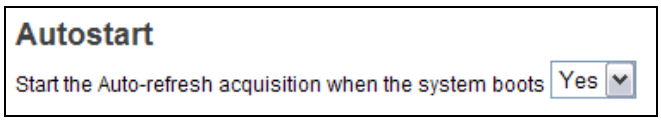
This chapter lists only those steps that are necessary to apply the programming sample **BasicSample** with an MSX-E system. A detailed description of the MSX-E web interface can be found in the general manual of the MSX-E systems (see PDF link) and in the respective system-specific MSX-E manual.

With the default setting of the **BasicSample**, you can configure an acquisition in Auto-refresh or Sequence mode.
Please note that in Sequence mode, the value of the sequence counter is not transmitted by default, whereas in Auto-refresh mode, this value is always available. For this reason, you have to activate the relevant setting on the web interface (see Chapter 4.1.2).

4.1.1 Auto-refresh mode

- On the web interface, from the menu on the left, select the item **Acquisition**, and on the right, select the **Auto-refresh** tab.

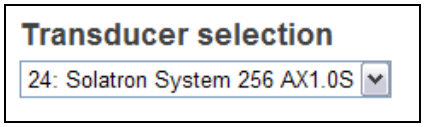
- Optional: In the **Autostart** section, select **Yes** to activate the autostart function.



- In the **Channel configuration** section, select the channels you want to acquire.

Channel	Transducer	Selection
Channel 0	Inductive transducer 0	☑
Channel 1	Inductive transducer 1	☑
Channel 2	Inductive transducer 2	☐
Channel 3	Inductive transducer 3	☐
Channel 4	Inductive transducer 4	☐
Channel 5	Inductive transducer 5	☐

- Select the connected transducer type. It is identical for all channels.

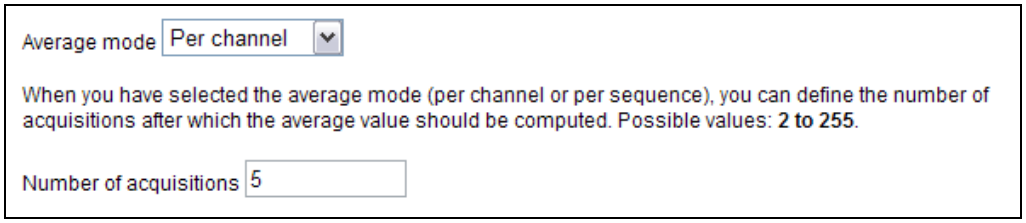


Optional: **Average** section

The MSX-E system is capable of computing an average value for each channel.
In the field **Number of acquisitions**, you have to enter the number of acquisitions (2 to 255) after which this value should be computed.

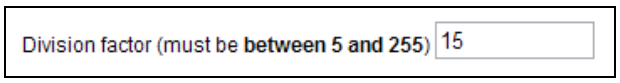
The acquisition can be carried out in two different ways **(Average mode)**:

- **Per sequence:** All selected channels are acquired simultaneously.
- **Per channel:** The selected channels are acquired individually.



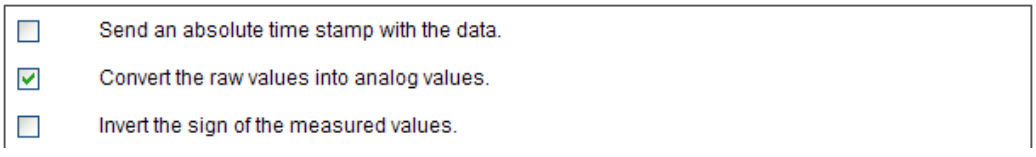
In the **Division factor** section, the settling time is set, i.e. the time required to switch from one channel to another. If only one channel is selected, this parameter is insignificant.

- Enter the **Division factor** (value between 5 and 255).



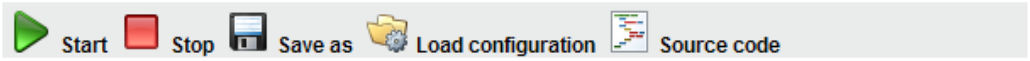
In the S7 sample, the type cast setting (see Chapter 4.2.3) is selected by default. Thus, in the section **Data server frame configuration**, you have to activate the option **Convert the raw values into analog values** so that the system immediately converts the raw values into the correct unit. The unit depends on the system type.

- Click on **Convert the raw values into analog values**.



- Optional: Click on **Set** and **Save** to save the current configuration.

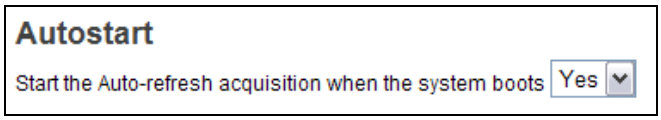
- To start the acquisition, click on **Start**.



4.1.2 Sequence mode

- On the web interface, from the menu on the left, select the item **Acquisition**, and on the right, select the **Sequence** tab.

- Optional: In the **Autostart** section, select **Yes** to activate the autostart function.

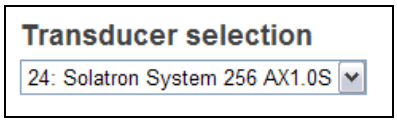


- In the **Channel configuration** section, select the channels you want to acquire.

You can define the acquisition order of the channels. This is displayed in the correspondent column as soon as you have selected a channel. Each channel can be acquired only once per sequence.

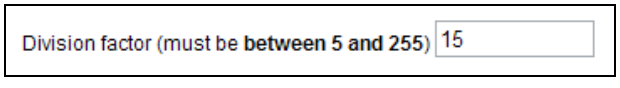
Channel	Transducer	Selection	Acquisition order
Channel 0	Inductive transducer 0	☑	2
Channel 1	Inductive transducer 1	☑	4
Channel 2	Inductive transducer 2	☑	1
Channel 3	Inductive transducer 3	☑	0
Channel 4	Inductive transducer 4	☑	5
Channel 5	Inductive transducer 5	☑	3

- Select the connected transducer type. It is identical for all channels.



In the **Division factor** section, the settling time is set, i.e. the time required to switch from one channel to another. If only one channel is selected, this parameter is insignificant.

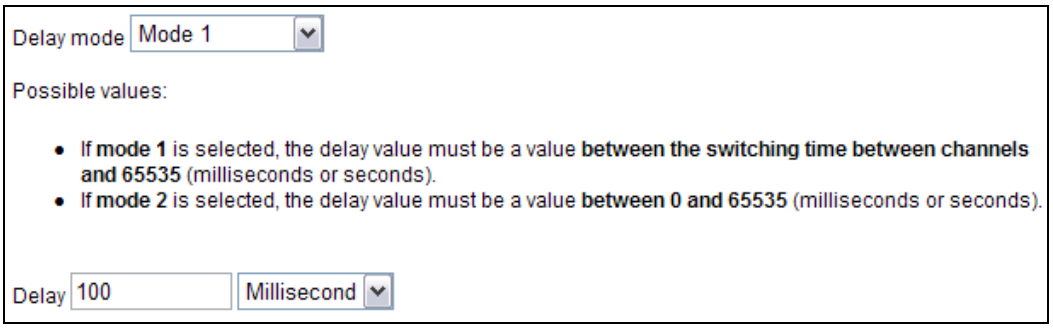
- Enter the **Division factor** (value between 5 and 255).



Optional: In the **Delay** section, you can define the delay between the individual sequences.
There are two modes:

- **Mode 1:** The time between the start of two subsequent sequences is defined as the delay.
- **Mode 2:** The time between the end of a sequence and the start of the subsequent sequence is defined as the delay.

With **Delay**, you have to define the value and the unit of the wait time (ms or s).

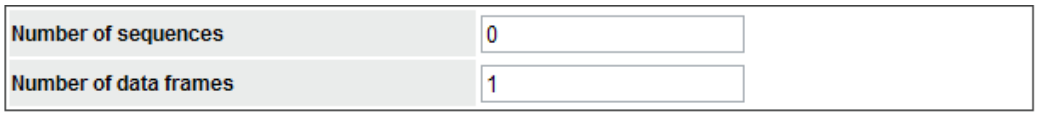


In the **Sequence measurement** section, in the field **Number of sequences**, you have to enter the number of sequences to be acquired (1 to 4294967295). If this value is 0, the acquisition is continuous.

In the field **Number of data frames**, you need to define the number of sequences (1 to 4096) that have to be acquired before the measurement values are sent to the target system.

NOTICE!
The value entered must not be higher than the value in the field "Number of sequences". The latter must be divisible by this value.

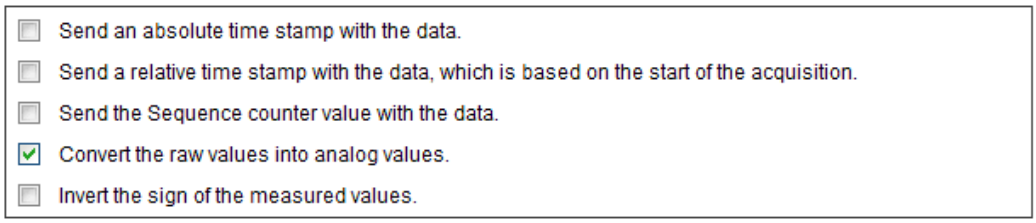
If the MSX-E system does not have sufficient memory to store the required number of sequences, the measurement values are sent earlier, that is, before the maximum number of sequences to be acquired is reached. This helps to reduce the network traffic load and the CPU resources of the MSX-E systems.



In the S7 sample, the type cast setting (see Chapter 4.2.3) is selected by default. Thus, in the section **Data server frame configuration**, you have to activate the option **Convert the raw values into analog values** so that the system immediately converts the raw values into the correct unit. The unit depends on the system type.

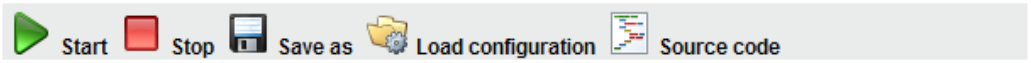
- Click on **Convert the raw values into analog values**.

In Sequence mode, the sequence counter counts the number of sequences that have already been acquired. If the option **Send the sequence counter value** is activated, the sequence counter value is sent in addition to the data.



- Optional: Click on **Set** and **Save** to save the current configuration.

- To start the acquisition, click on **Start**.



4.2 S7: Sample description

4.2.1 OB1: Main block

Description:

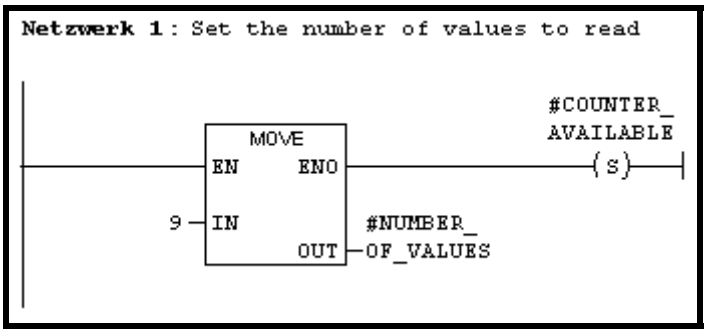
OB1 (organisation block 1) is executed periodically. As soon as it ends, it starts again. To obtain a simple overview of the entire source code, all function blocks are called from OB1 in **BasicSample**.

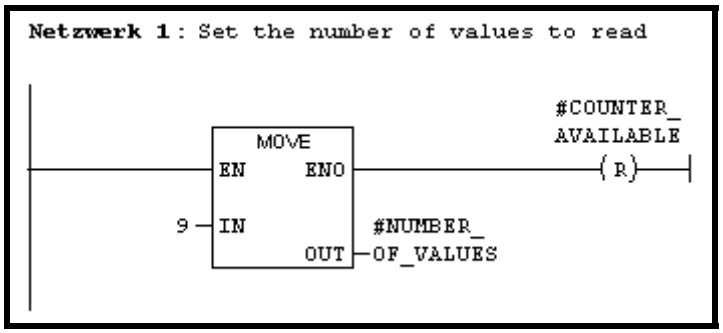
4.2.2 Network 1: Definition of the number of values to read

Description:

Enter the number of values to read.

In this example, 9 values are read. The first value can be a value of the sequence counter, or a value of the 8 channels. If a sequence counter value is expected, set the COUNTER_AVAILABLE variable. If not, this variable should be reset.





4.2.3 Network 2: Selecting the appropriate conversion unit

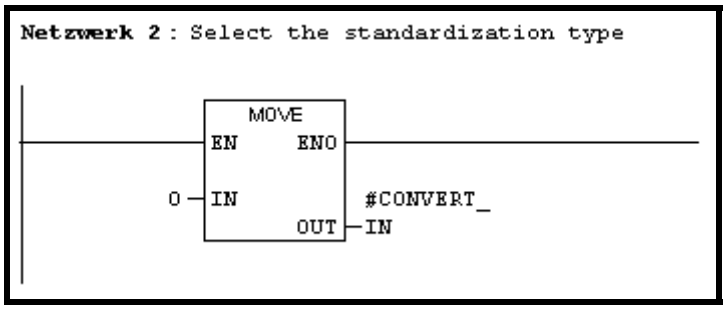
Description:

Before the **BasicSample** is transmitted to the S7, you need to define in Network 2 the conversion unit using the CONVERT_IN variable.

CONVERT_IN = 0: The system does not return any raw values, only the converted values (type cast is executed).

CONVERT_IN = 1: The system returns raw values. With the MSX-E37xx system, these are converted by the PLC into the unit millimetres.

CONVERT_IN = 2: The system returns raw values. With the MSX-E3011 system, these are converted by the PLC into the unit volts.



As already mentioned, you can define on the web interface of the MSX-E system that the measurement values should be converted into the appropriate unit (see Chapter 4.1.2, section **Data server frame configuration**).

4.2.4 Network 3: Reading the measurement values from the TCP connection

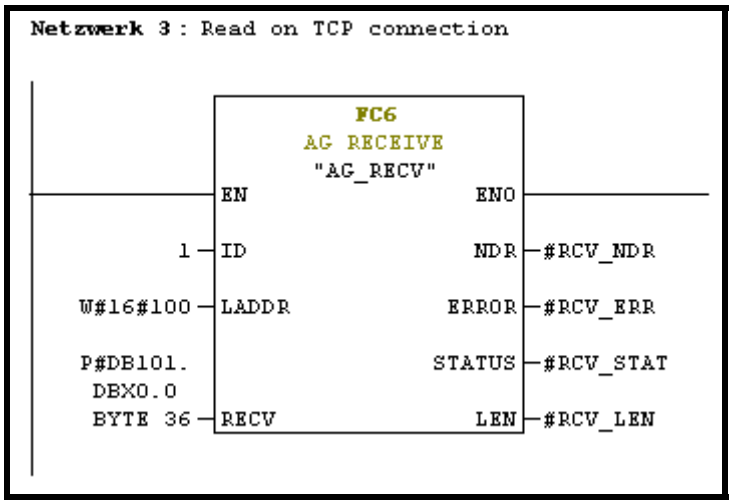
Description:

FC6 attempts to read 36 bytes from the MSX-E system from the TCP connection ID 1 and ADDR W#16#100. If no connection errors have occurred and the 36 bytes are available, the RCV_NDR variable is set to “true”.

The RECV (P#DB101.DBX0.0) input parameter is a pointer to the data.

DB101 is a data block to which the 36 bytes are copied (36 bytes = 9 x DWORD). RECV can be adapted as required. If the number of bytes to read has been changed, you should also check whether the number of elements in the DB101 and DB102 data blocks is sufficient.

For more information, see Chapter 3.2.7.

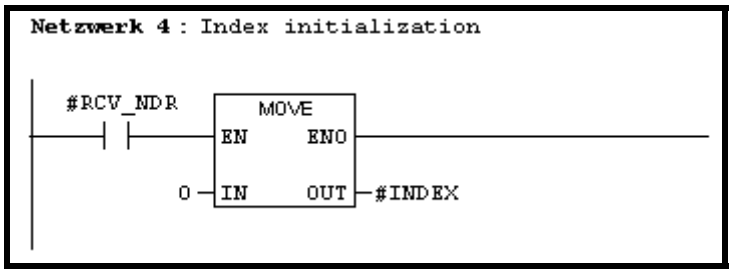


4.2.5 Network 4: Initialising the index

Description:

The program uses two data structures.

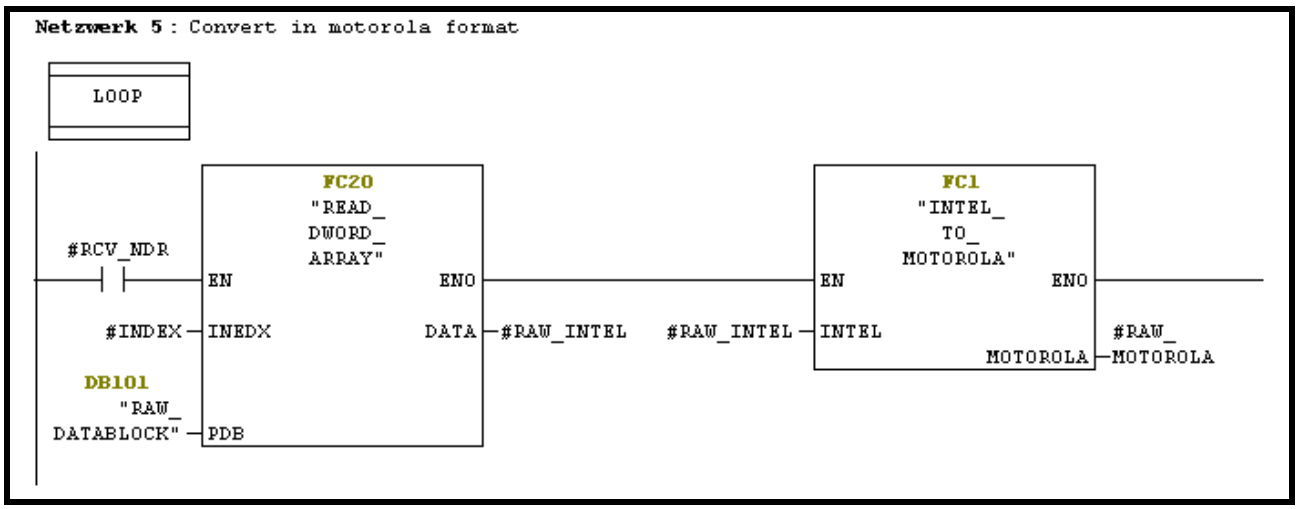
The RAW_DATABLOCK (DB101) structure contains a DWORD table in which the raw values that have been read are available.
The data structure REAL_DATABLOCK (DB102) contains a REAL table in which the converted raw values are saved as a REAL data type. The INDEX variable is set to 0. INDEX is used to manage the table index dynamically.



4.2.6 Network 5: Display in Motorola format

Description:

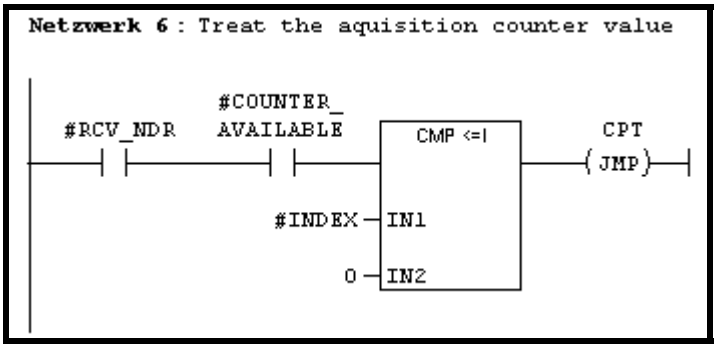
From the RAW_DATABLOCK table, a raw value is read from the relevant cell in the INDEX. This value is converted from the Intel format (MSX-E system) into the Motorola format (PLC).



4.2.7 Network 6: Managing the sequence counter value

Description:

If the sequence counter value is expected and the INDEX is 0, the value is a sequence counter value. The program then jumps into the network that contains the CPT marking. The sequence counter value is always the first value in the table.

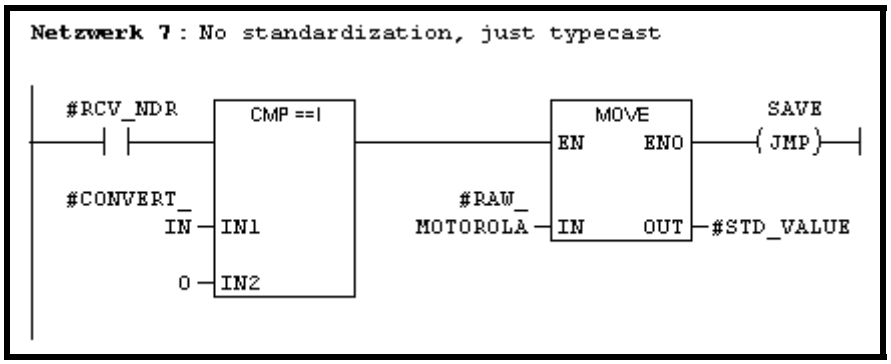


4.2.8 Network 7: Type cast instead of conversion

Description:

As the system already returns converted values, no additional conversion is required.

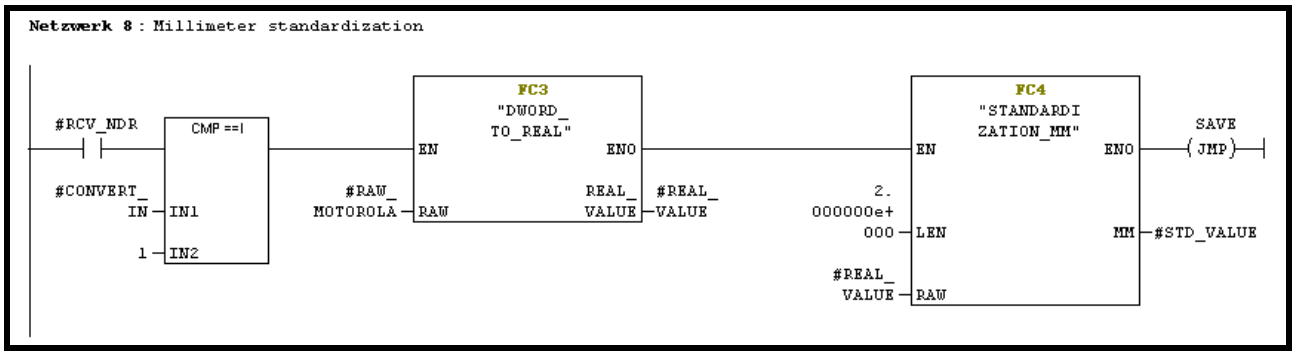
MOVE copies the DWORD raw values into a variable of the data type REAL (type cast). Afterwards, the program jumps into the network with the marking SAVE.



4.2.9 Network 8: Converting to millimetres

Description:

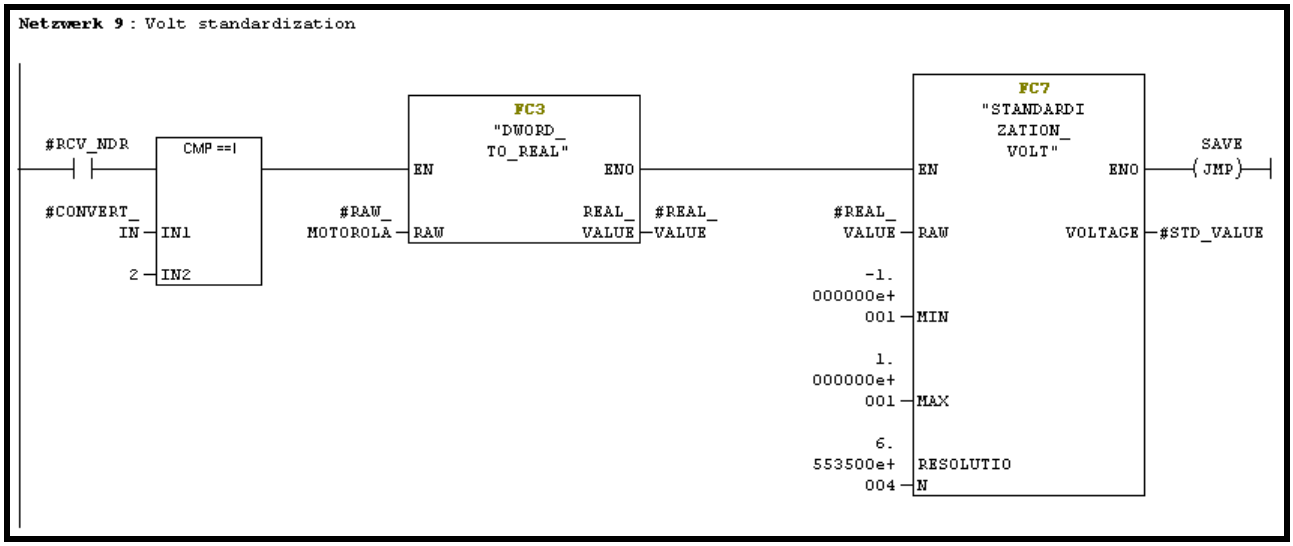
The system returns raw values. With FC3, the DWORD raw values are converted to the REAL data type, and with FC4 they are converted into the unit millimetres. Subsequently, the program jumps into the network with the marking SAVE.



4.2.10 Network 9: Converting to volts

Description:

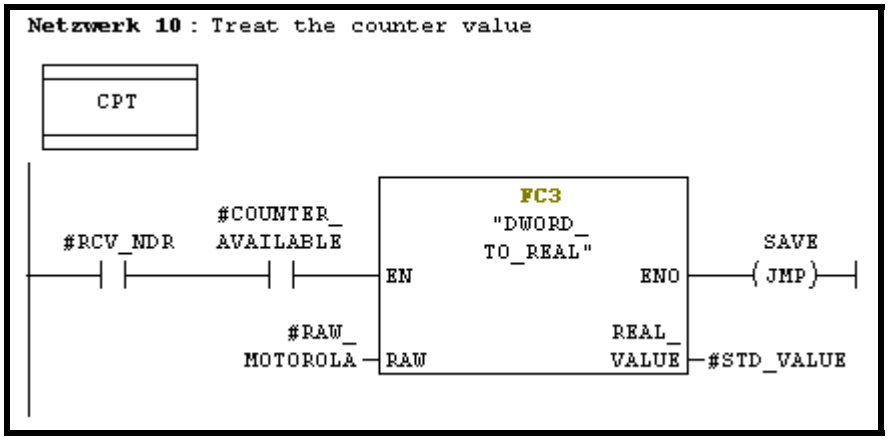
The system returns raw values. With FC3, the DWORD raw values are converted to the REAL data type, and with FC7 they are converted into the unit volts. The program then jumps into the network that contains the SAVE marking.



4.2.11 Network 10: Processing the sequence counter value

Description:

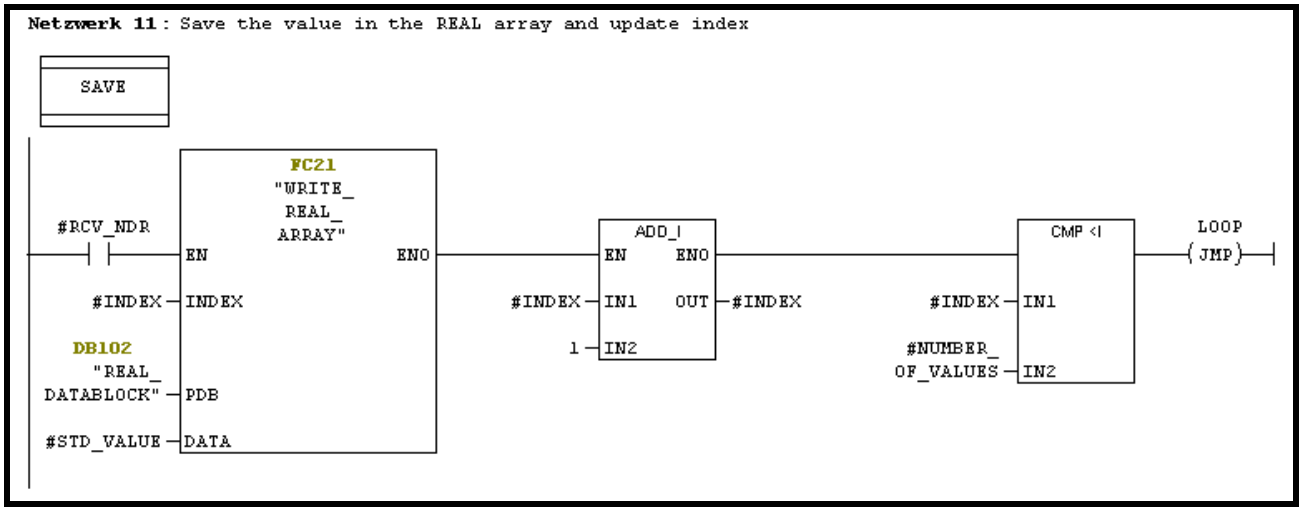
The sequence counter value is converted as a DWORD data type into REAL. Afterwards, the program jumps into the network with the marking SAVE.



4.2.12 Network 11: Save and check

Description:

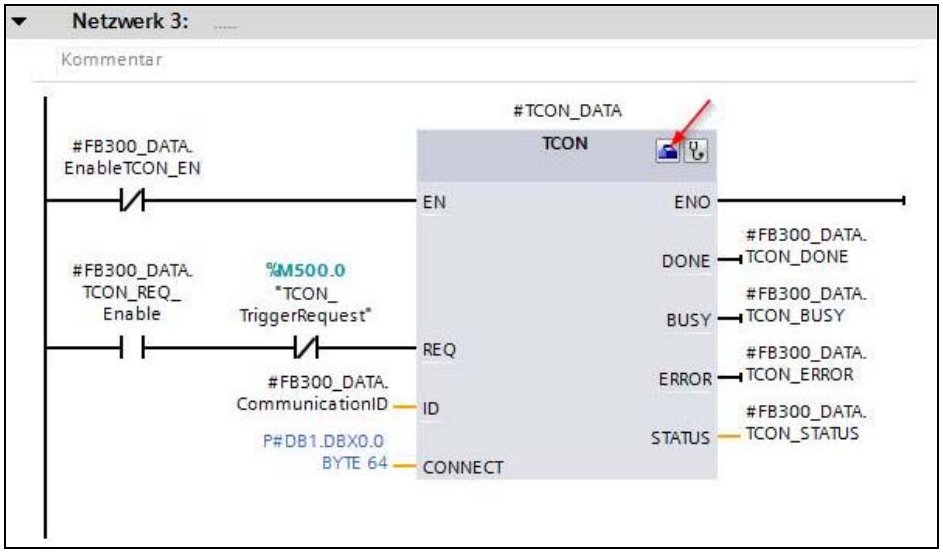
FC21 saves the value in the REAL table. ADD_I increases the INDEX variable so that the next raw value can be read. CMP < I checks whether all raw values have been read. If not, the program jumps into the network with the marking LOOP.



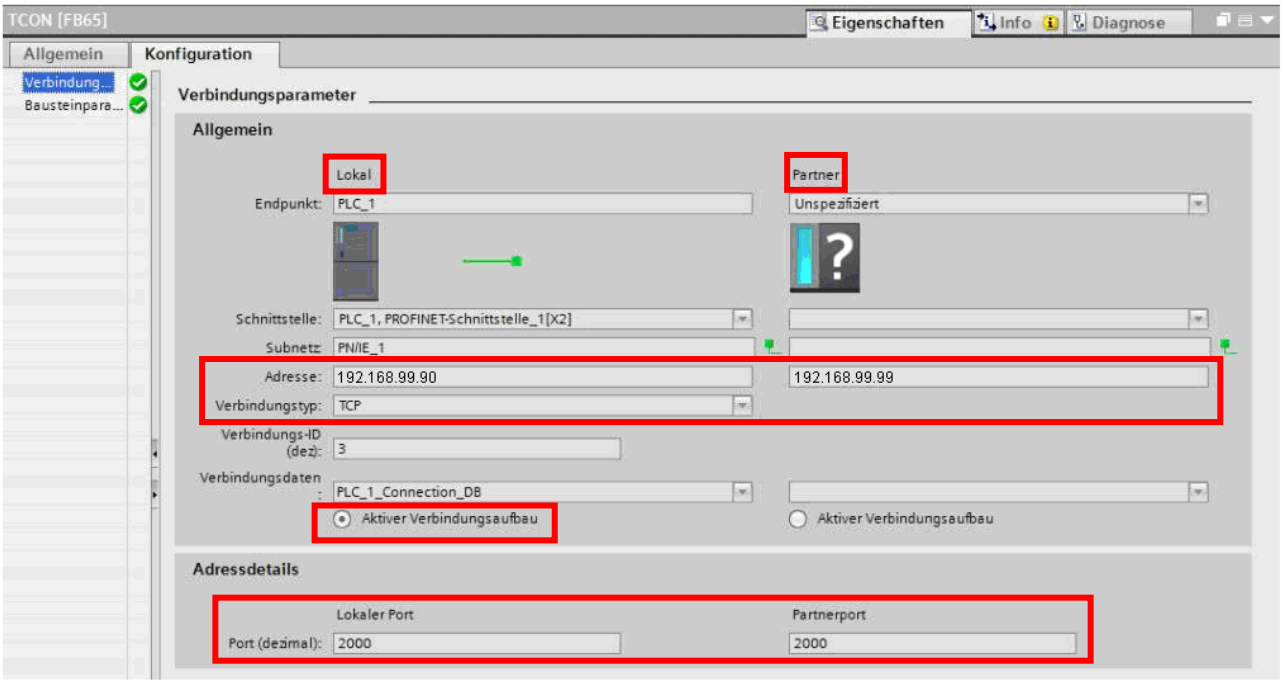
4.3 TIA software environment with VIPA CPU 015

Below, you will find a short description of the sample program **CPU015_TRECV** to read MSX-E data using a PLC with T-blocks.
The sample program has been generated with **TIA Portal V13** and can be viewed in the programming languages AWL (STL), KOP (LAD) and FUP (FBD).

- Use the PLC network interface **X4 PN**.
- Click on the toolbox icon (see red arrow) to set the communication parameters under **FB300 Netzwerk 3 (TCON)**.

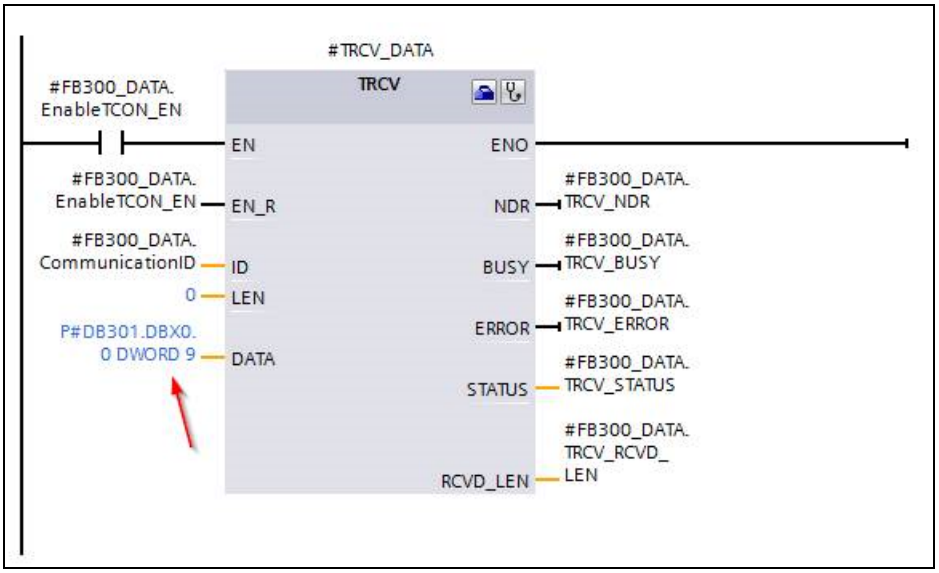


The following window is displayed:



- In the section **Allgemein** [General], in the field **Adresse** [Address], enter the IP address of the PLC (**Lokal** area) and the one of the MSX-E system (**Partner** area). These two IP addresses must be located in the same network area.
- Select the connection type (Verbindungstyp) **TCP**.
- Activate the PLC mode **Aktiver Verbindungsaufbau** [Active connection establishment] (**Lokal** area).
- In the section **Adressdetails**, in the areas **Lokaler Port** (PLC) and **Partnerport** (MSX-E system), enter the port numbers **2000**. These two port numbers must be identical and must correspond to the port number defined on the web interface of the MSX-E system (see Chapter 2.2).

The amount of data to be received must be set in **Netzwerk 10** [Network 10]. **LEN** is to equal 0.
In the **DATA** pointer, the number of DWORDs is indicated.
Example: For an MSX-E acquisition in Auto-Refresh mode with 8 channels, 9 DWORDs must be defined (1 counter value + 8 channel values each correspond to 1 DWORD).



5 Recommendations

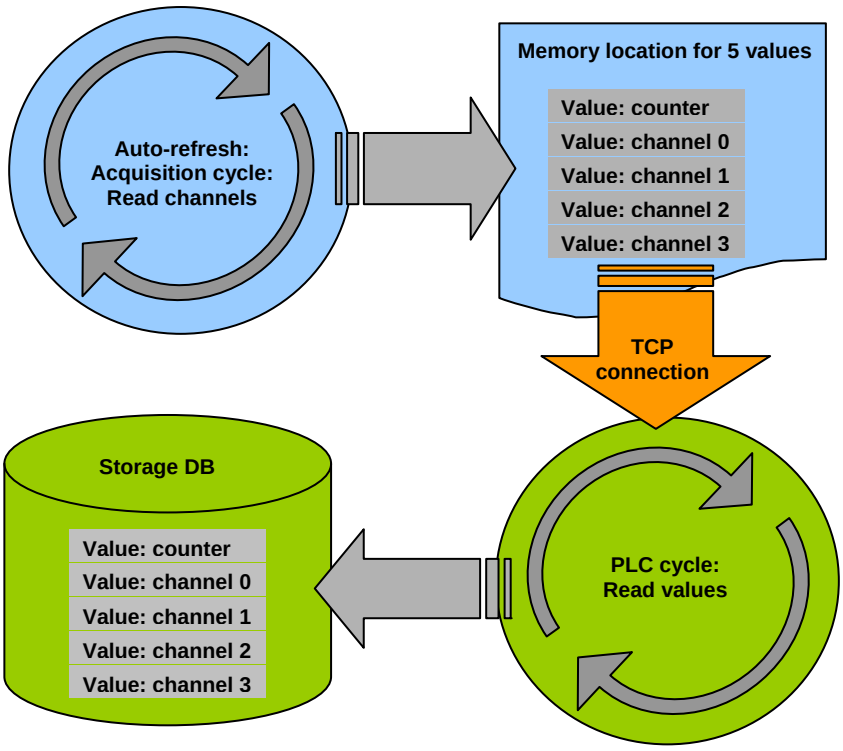
5.1 Auto-refresh and Sequence modes

NOTICE!

If you do not configure an average value calculation in Auto-refresh mode or a delay in Sequence mode, the PLC requires a certain time to read the values from the TCP connection (PLC cycle > acquisition cycle). This is also the case if a small **Division factor** is configured in both modes. So this may result in a delay between the actual status of the channels and the measurement values that are read.
If the packets from the TCP connection are not read quickly enough, there may be a FIFO overrun which causes the MSX-E system to cancel the connection to the PLC.

5.1.1 Acquisition in Auto-refresh mode

Fig. 5-1: Auto-refresh mode: Acquisition cycle



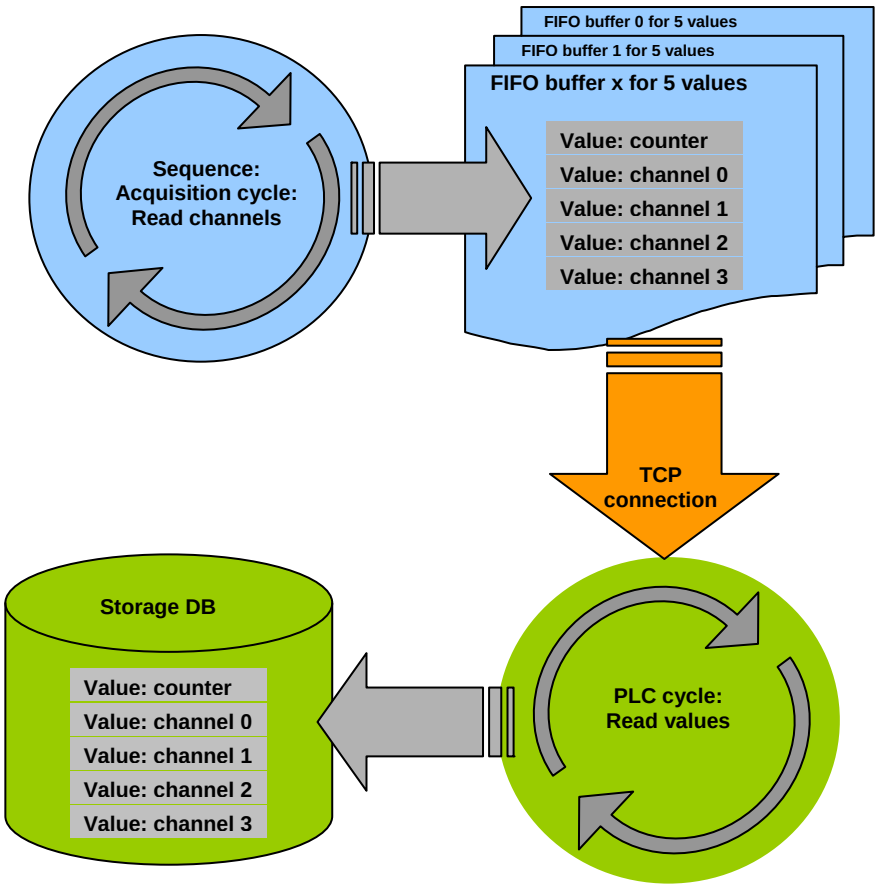
The acquisition cycle is displayed with four channels of an MSX-E system in Auto-refresh mode (light blue). As only one memory location is available for the five values, after each cycle, the values of the previous cycle are overwritten with those of the current cycle.
The MSX-E system sends the values to the PLC as soon as the PLC is connected. The values from the TCP connection are read in the PLC cycle (green).

MSX-E acquisition cycle (Auto-refresh mode) < PLC cycle -> values may be lost
MSX-E acquisition cycle (Auto-refresh mode) > PLC cycle -> all values are received
MSX-E acquisition cycle (Auto-refresh mode) = PLC cycle -> all values are received

NOTICE!
With a socket FIFO overrun, values may be lost.

5.1.2 Acquisition in Sequence mode

Fig. 5-2: Sequence mode: Acquisition cycle



The acquisition cycle is displayed with 4 channels of an MSX-E system in Sequence mode (light blue). The values of the current cycle are written into a new FIFO buffer, i.e. no values are overwritten. The MSX-E system sends the values to the PLC as soon as the PLC is connected. The values from the TCP connection are read in the PLC cycle (green).

MSX-E acquisition cycle (Sequence mode) < PLC cycle -> all values are received
MSX-E acquisition cycle (Sequence mode) > PLC cycle -> all values are received
MSX-E acquisition cycle (Sequence mode) = PLC cycle -> all values are received

NOTICE!
With a socket FIFO overrun, values may be lost.

5.2 Trigger

With the MSX-E systems, the measurement can be started by a trigger.
More information on this can be found in the general manual of the MSX-E systems (see PDF link) and in the respective system-specific MSX-E manual.

The following information describes how the acquisition start can be controlled using a hardware trigger. Each time a rising edge is identified at the MSX-E system's trigger input, the system sends a packet with a sequence counter value and eight measurement values.

5.2.1 System configuration

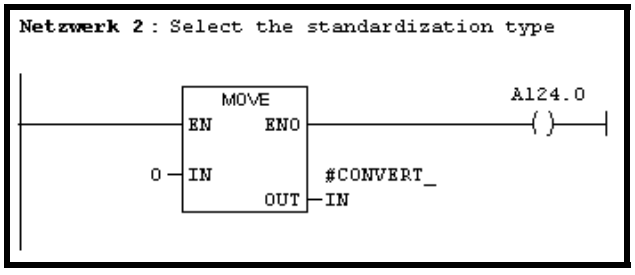
The configuration described in Chapter 4.1.1 (Auto-refresh mode) or Chapter 4.1.2 (Sequence mode) can be used as a basis for defining the following trigger configuration:

	Trigger source	Trigger mode	Number of sequences per trigger
Description	Trigger mask (API)		Number of sequences to be acquired at each trigger event
Value	Hardware	Sequence	1 (1 - 65535)

	Hardware trigger active edge	Hardware trigger count
Description		Number of trigger events before the acquisition starts
Value	Rising	1 (1 - 65535)

5.2.2 BasicSample: Example of trigger enhancement

To start the acquisition, you can set a digital output (A124.0) for the system in **Network 2**. This has to be connected with the system's digital trigger input **(Trig/Sync In)**.



5.3 Querying the current measurement values in Auto-refresh mode

The PLC has the option of querying one or more current measurement values asynchronously (without a hardware trigger). This method can be used if only one or more current measurement values of the system are required at a certain time.

The following chapter describes how a current packet containing a sequence counter value and eight measurement values is read. The **BasicSample** is adapted accordingly for this purpose (see Chapter 5.3.2).

Please note that the transmission time is extended if several PLCs access an MSX-E system simultaneously.

5.3.1 System configuration

The configuration described in Chapter 4.1.1 (Auto-refresh mode) or Chapter 4.1.2 (Sequence mode) can be used as a basis.

- On the web interface, click on the menu item **Data server**.
- On the **Blocking transfer** tab, with **Activate blocking TCP/IP transfer**, select the option **Yes**.

For **TCP/IP transfer timeout**, you can set a timeout of 2 seconds so that the data server does not permanently block the data.

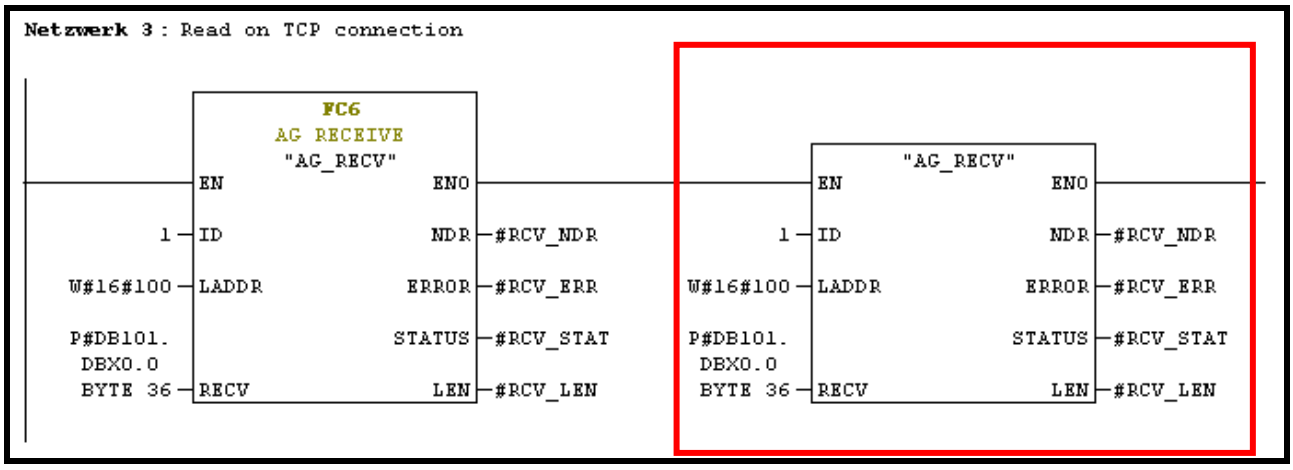
- In the **Action** section, click on **Save** and then on **Restart**.

5.3.2 BasicSample: Adjustment example for querying the measurement values

So that you can read the current values, two data packets have to be acquired. Only the second packet is taken into account, as the first packet contains previous measurement values.

The **BasicSample** is adjusted as follows:

- In Network 3, copy the function block FC6 and insert it to the right of the original block.

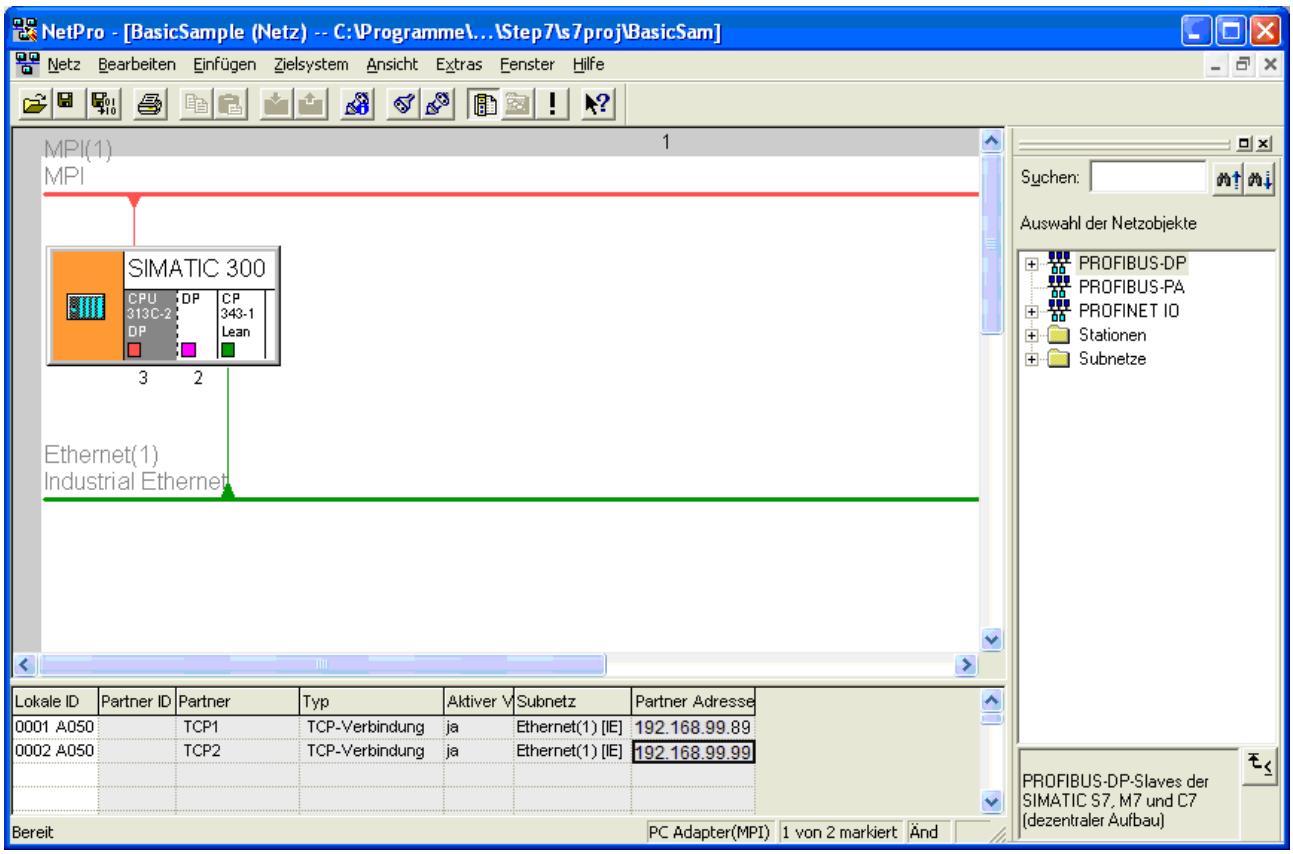


5.4 Connecting several systems to a PLC

You can connect several MSX-E systems to a PLC. In this case, for each system, you have to create a new TCP connection (socket).

- To do so, follow the instructions in Chapter 2.2 and enter a different port number for each system in the window **Eigenschaften – TCP-Verbindung** [Properties – TCP connection] (see Fig. 2-4).

Fig. 5-3: SIMATIC Manager: S7 Ethernet configuration



6 Appendix

6.1 Glossary

AWL
= Anweisungsliste

AWL is a programming language according to IEC 61131-3 for PLC programming.
It corresponds to the instruction list language STL for Siemens PLCs.

Counter
A counter is a circuit which counts pulses or measures pulse duration.

Data block (DB)
Data blocks are data areas in the Siemens STEP 7 system, in which the data of the user program is stored. There are two data block types: instance data blocks and global data blocks. The former contain static local data, the latter contain information that can be accessed from all code blocks. Instance data blocks are generated together with function blocks (FBs), whereas global data blocks must be programmed by the user.

Ethernet
The Ethernet is a baseband bus system originally developed in order to connect mini-computers. It is based on the CSMA/CD access method. Coaxial cables or twisted-pair cables are used as the transmission medium.
The transmission speeds are 10 Mbit/s (Ethernet), 100 Mbit/s (Fast Ethernet) and 1 Gbit/s or 10 Gbit/s (Gigabit-Ethernet).
This widely used technology for computer networking in a LAN has been standardised since 1985 (IEEE 802.3 and ISO 8802-3).
Ethernet technology is now common practice in the office environment. After making even very tough real-time requirements possible and adapting the device technology (bus cables, patch fields, junction boxes) to the harsh application conditions of the industrial environment, Ethernet is now also increasingly used in the field areas of automation technology.

FIFO
= First in, First out

Data that is written first into the FIFO memory buffer gets first out of the buffer.

Function (FC)
In the Siemens STEP 7 system, there is the code block type “functions”, which contains functions of structured user programs. This code block type is parameterisable and can store local data temporarily.

Function block (FB)
In the Siemens STEP 7 system, function blocks of structured user programs are part of the code blocks and contain task-specific subroutines. A function block always includes an instance data block, i.e. a particular memory area for special data. FBs are parameterisable and can store local data both statically and temporarily.

IP address
The IP address is a unique string of numbers separated by full stops that identifies each computer attached to the Internet. Usually, it also has a version containing words separated by full stops.

LSB
= Least Significant Bit

LSB is the lowest order bit in a digital quantity.

MSB
= Most Significant Bit

MSB is the highest order bit in a digital quantity.

Organisation block (OB)
In the Siemens STEP 7 system, organisation blocks belong to the code blocks and are the interface between the operating system and the user program. OBs can be called up by the operating system in certain situations when OB1, which is otherwise endlessly executed, is interrupted.

PLC
= Programmable Logic Controller

The PLC is a computer-based control unit whose functionality is defined by an application program. With standardised technical languages, this application program is relatively easy to produce. Because of its serial mode of operation, reaction times of PLCs are slower than those of VPS. As a family of devices with graduated and matched components, PLCs can now cover all levels of an automation hierarchy.

Socket
A socket is a bidirectional software interface to interprocess (IPC) or network communication. Sockets are a standardised interface (API) between the network protocol implementation of the operating system and the actual application software.

Trigger
A trigger is a pulse or signal for starting or stopping a special task. Triggers are often used for controlling data acquisition.

6.2 Index

7 Contact and support

Do you have any questions? Write or call us:

Address: ADDI-DATA GmbH
Airpark Business Center
Airport Boulevard B210
77836 Rheinmünster
Germany

Phone: +49 7229 1847-0
Fax: +49 7229 1847-222

E-mail: info@addi-data.com

Manual and software download from the Internet:

www.addi-data.com

ADDI-DATA®
PARTNER FÜR PRÄZISION

Instruction Manual

Intelligent Ethernet I/O systems (Part 3)

ADDI-DATA Modbus TCP client library
for a SIMATIC® S7® PLC

DIN EN ISO 9001:2015 certified

Edition: 02.02-09/2018

Product information

This manual contains the technical installation and important instructions for correct commissioning and usage, as well as production information according to the current state before printing.
The content of this manual and the technical product data may be changed without prior notice.
ADDI-DATA GmbH reserves the right to make changes to the technical data and the materials included herein.

Warranty and liability

The user is not authorised to make changes to the product beyond the intended use, or to interfere with the product in any other way.
ADDI-DATA shall not be liable for obvious printing and phrasing errors. In addition, ADDI DATA, if legally permissible, shall not be liable for personal injury or damage to materials caused by improper installation and/or commissioning of the product by the user or improper use, for example, if the product is operated despite faulty safety and protection devices, or if notes in the operating instructions regarding transport, storage, installation, commissioning, operation, thresholds, etc. are not taken into consideration. Liability is further excluded if the operator changes the product or the source code files without authorisation and/or if the operator is guilty of not monitoring the permanent operational capability of working parts and this has led to damage.

Copyright

ADDI-DATA software product licence

Please read this licence carefully before using the standard software. The customer is only granted the right to use this software if he/she agrees with the conditions of this licence.
The software may only be used to set up the ADDI-DATA products.
Reproduction of the software is forbidden (except for back-up and for exchange of faulty data carriers). Disassembly, decompilation, decryption and reverse engineering of the software are forbidden. This licence and the software may be transferred to a third party if this party has acquired a product by purchase, has agreed to all the conditions in this licence contract and the original owner does not keep any copies of the software.

Trademarks

- ADDI-DATA, APCI-1500, MSX-Box and MSX-E are registered trademarks of ADDI-DATA GmbH.
- Turbo Pascal, Delphi, Borland C, Borland C++ are registered trademarks of Borland Software Corporation.
- Microsoft .NET, Microsoft C, Visual C++, MS-DOS, Windows XP, Windows 7, Windows 8, Windows 10, Windows Server 2000, Windows Server 2003, Windows Embedded and Internet Explorer are registered trademarks of Microsoft Corporation.
- LabVIEW, LabWindows/CVI, DASYLab, DIAdem are registered trademarks of National Instruments Corporation.
- CompactPCI is a registered trademark of PCI Industrial Computer Manufacturers Group.
- VxWorks is a registered trademark of Wind River Systems, Inc.
- RTX is a registered trademark of IntervalZero.
- Mozilla Firefox is a registered trademark of Mozilla Foundation.
- SIMATIC S7, S7-300, STEP7, TIA Portal, CPU313C-2DP and CP343-1 Lean are registered trademarks of Siemens AG.

Warning!

The following risks result from the improper implementation of the Ethernet system and from use contrary to the regulations:

Personal injury

Damage to the Ethernet system, the PC and peripherals

Pollution of the environment.

- Protect yourself, others and the environment!
- Read the safety precautions (yellow leaflet) carefully!

  If this leaflet is not enclosed with the documentation, please contact us and ask for it.
- Observe the instructions of this manual!

  Make sure that you do not forget or skip any step!
  We are not liable for damages resulting from the wrong use of the Ethernet system.
- Pay attention to the following symbols:

NOTICE!
Designates hints and other useful information.

NOTICE!
Designates a possibly dangerous situation.
If the instructions are ignored, the Ethernet system, the PC and/or peripherals may be **destroyed**.

WARNING!
Designates a possibly dangerous situation.
If the instructions are ignored, the Ethernet system, the PC and/or peripherals may be **destroyed** and persons may be **endangered**.

Contents

Figures

Chapter overview

In this manual, you will find the following information:

Chapter	Content
1	General information on the ADModbus library, the Modbus TCP protocol, etc.
2	Description of the MSX-E Modbus TCP functions
3	Information on the ADModbus blocks
4	Information on integrating the ADModbus library
5	Procedure for configuring the PLC
6	Importing and compiling the MSX-E Modbus TCP functions
7	Explanation of how to read data from the MSX-E data server
8	Notes on the S7 programming sample
9	Information on data acquisition in Auto-refresh or Sequence mode
10	Appendix with glossary and index
11	Contact and support address

1 ADDI-DATA Modbus TCP client library

1.1 Brief description

Modbus TCP is a non-proprietary communication protocol, which is supported by different device types in the world of industry, like for example by a PLC.
The MSX-E systems from ADDI-DATA have a Modbus TCP server and can be directly managed and controlled by a PLC using a Modbus TCP client library[1]. Via telegrams, the PLC can, for instance, read system information and start or stop measurements.
A function block (FB) according to the standard IEC 61131-3 is available for each function of the MSX-E systems and can be imported into the standard SIMATIC Manager. As the function blocks are adapted to the Modbus TCP server of the MSX-E systems, it is possible to configure the MSX-E systems quickly in the SIMATIC Manager.

1.2 Modbus TCP protocol

The Modbus TCP protocol is a query/response protocol, which can execute different tasks with the aid of defined classes (FC = Function Code).
The Modbus TCP server of the MSX-E systems supports the basic "Class 0". This consists of the Modbus TCP functions that should at least be used by a client and a server:

- **FC3** ("Read multiple registers")
- **FC16** ("Write multiple registers")

1.3 Software package

The software package supplied on a CD has the following structure:

- **MXXXXX{0000}_XXXX_modbus_client**
  - **Documentation**
    - **html** (description of the MSX-E Modbus TCP functions)
  - **S7**
    - **src** (AWL/STL source code files of the MSX-E Modbus TCP functions)
    - **samples.zip** (S7 programming sample for using the ADModbus library)

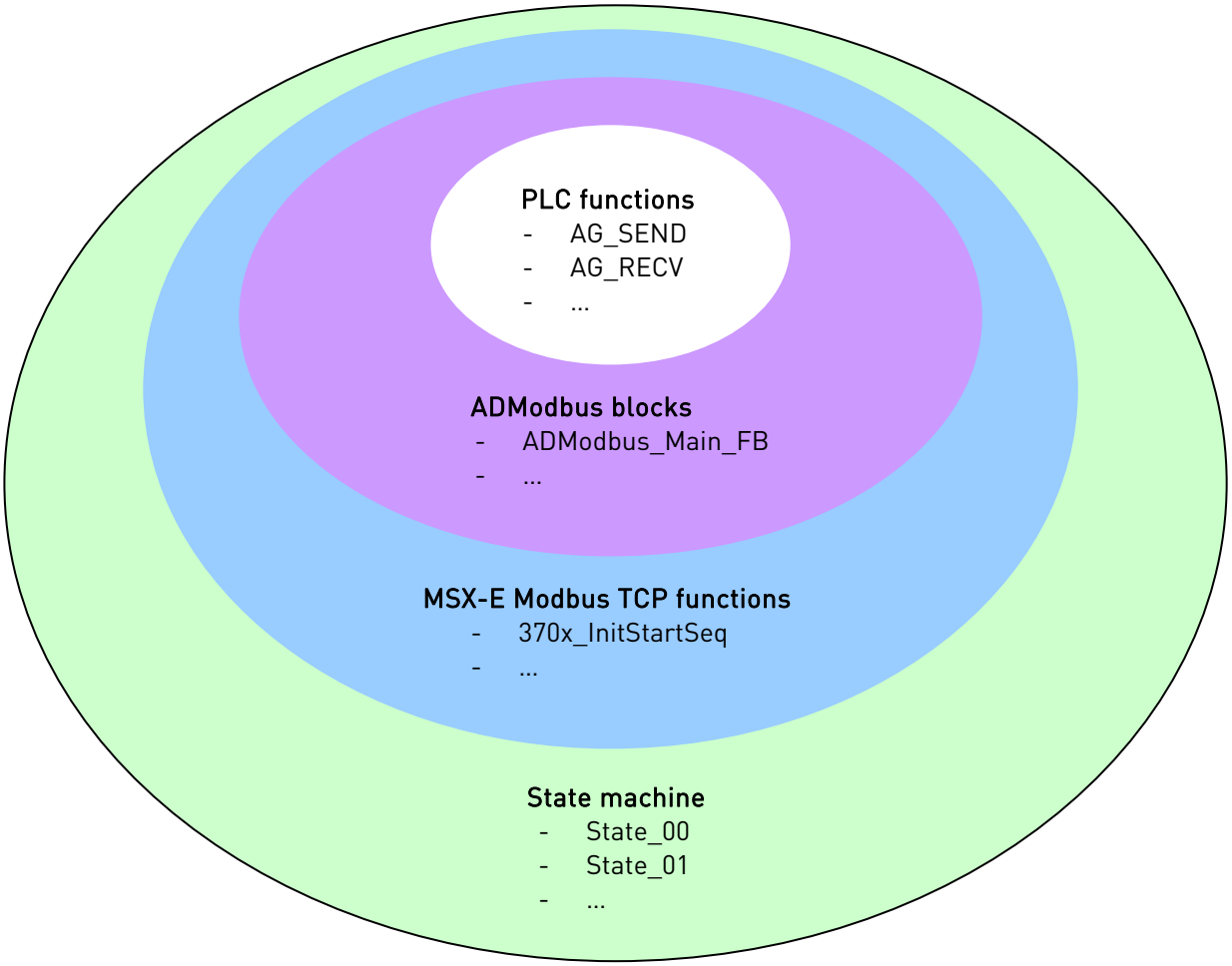
[1] Short form: ADModbus library

1.4 Structure of the ADModbus library

The ADModbus library comprises three programming levels: PLC functions, ADModbus blocks and MSX-E Modbus TCP functions. ADDI-DATA provides the latter for programming the communication between a PLC and an MSX-E system via Modbus TCP.

The Modbus TCP protocol is automatically managed by the ADModbus library. This means that the user does not need any Modbus TCP knowledge to use the MSX-E Modbus TCP functions.

Fig. 1-1: ADModbus library: Structure overview

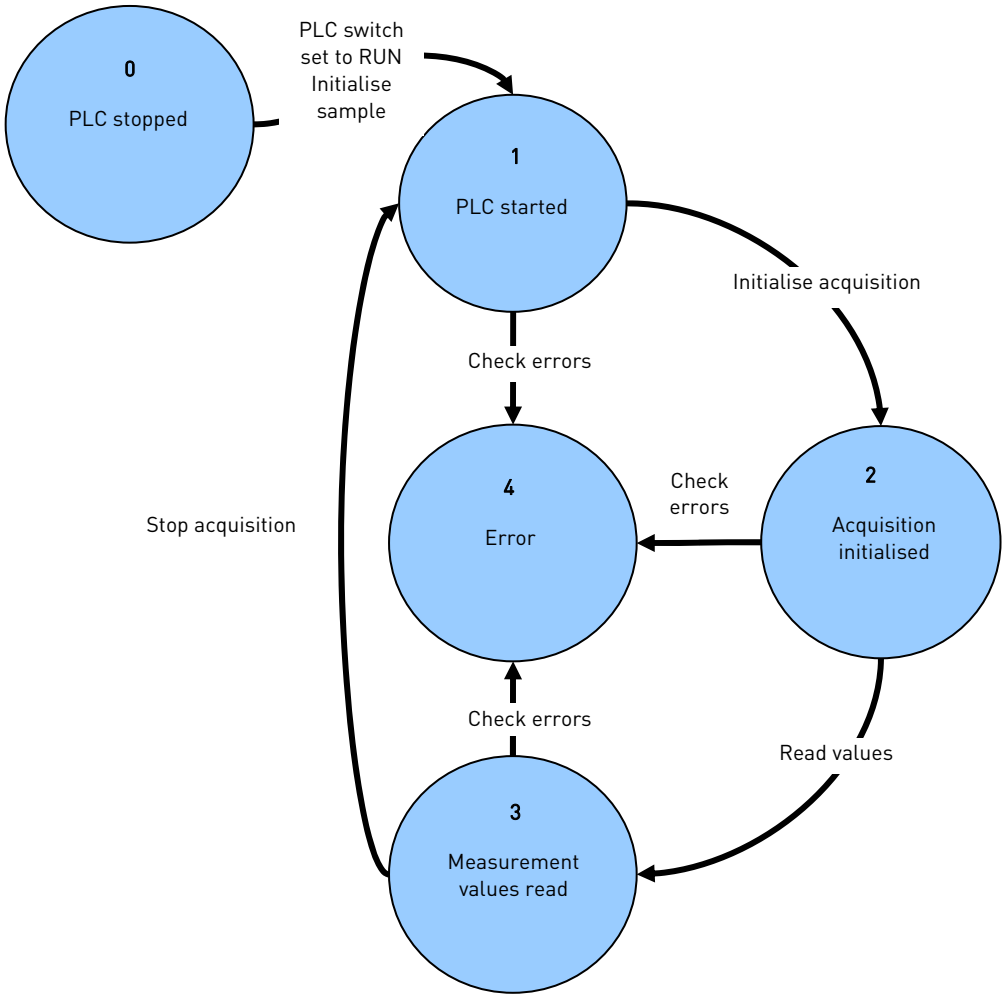


1.5 State machine

The state machine belongs only indirectly to the ADModbus library. However, it plays an important role because it manages the function calls in a defined order.

As an example, the acquired data of an **MSX-E-370x** system can be read with the following state machine:

Fig. 1-2: State machine: Example



NOTICE!

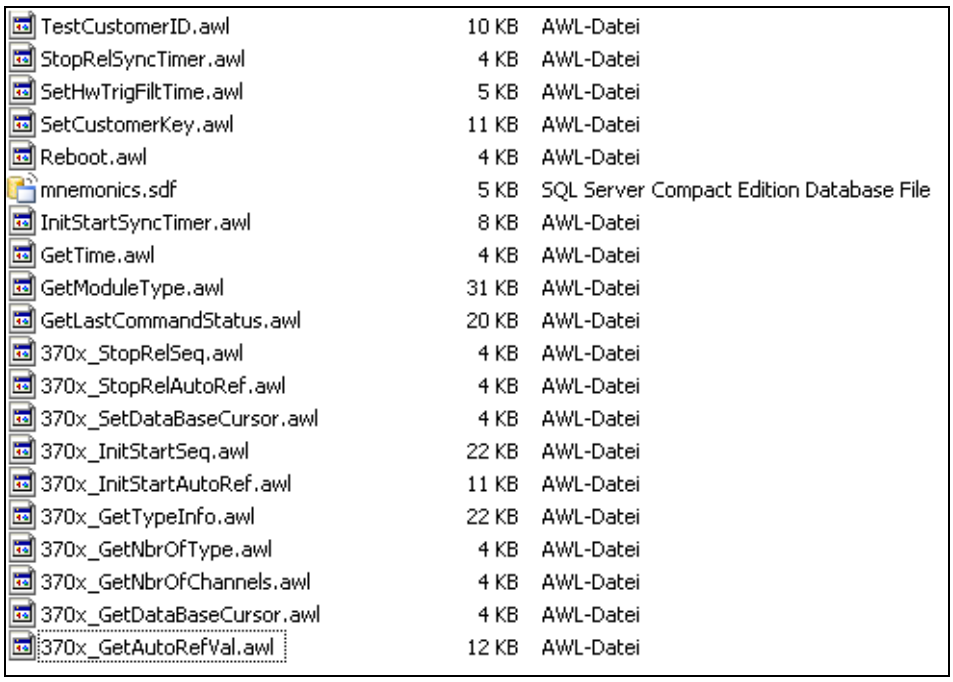
A maximum of one MSX-E Modbus TCP function may be called up for each state.

2 MSX-E Modbus TCP functions

2.1 Function blocks

For each MSX-E system that is equipped with a Modbus server, the MSX-E Modbus TCP functions are available in the form of AWL/STL source code files.

Fig. 2-1: MSX-E370x functions: AWL/STL source code files



The AWL/STL source code files can be imported into the SIMATIC Manager and be compiled into function and data blocks. Each function block works with an instance data block (name = I like **I**nstance + name of the MSX-E Modbus TCP function) and a global data block (name = SR like **S**end **R**eceived + name of the MSX-E Modbus TCP function). The global data block includes the query and response telegrams.
A part of the MSX-E Modbus TCP functions contained in the programming sample is already compiled. More information on importing the MSX-E Modbus TCP functions is to be found in Chapter 6.

Fig. 2-2: MSX-E370x functions: Function blocks

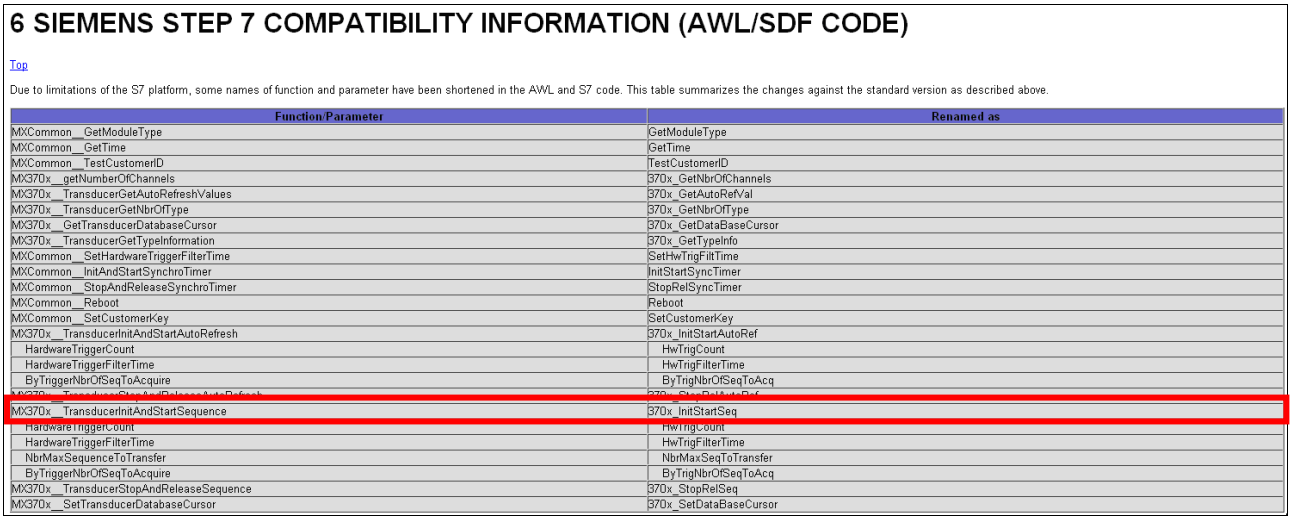
2.2 Function names

Due to the character limit in the SIMATIC Manager, the names of the MSX-E Modbus TCP functions have been automatically abbreviated.
To find the description of a specific function in the "msxeXXXX_modbus.html" documentation (see Chapter 1.3), you may first have to check in the table in the Chapter "Siemens STEP 7 compatibility information (AWL/STL/SDF code)" (see the following figure), which PLC function block belongs to which MSX-E Modbus TCP function.

Example:

"370x_InitStartSeq" is the name of a PLC function block, which can be found in the right-hand column of the table. The column on the left contains the name of the corresponding MSX-E Modbus TCP function.

Fig. 2-3: MSX-E Modbus TCP functions and corresponding PLC function blocks



6 SIEMENS STEP 7 COMPATIBILITY INFORMATION (AWL/SDF CODE)

Top

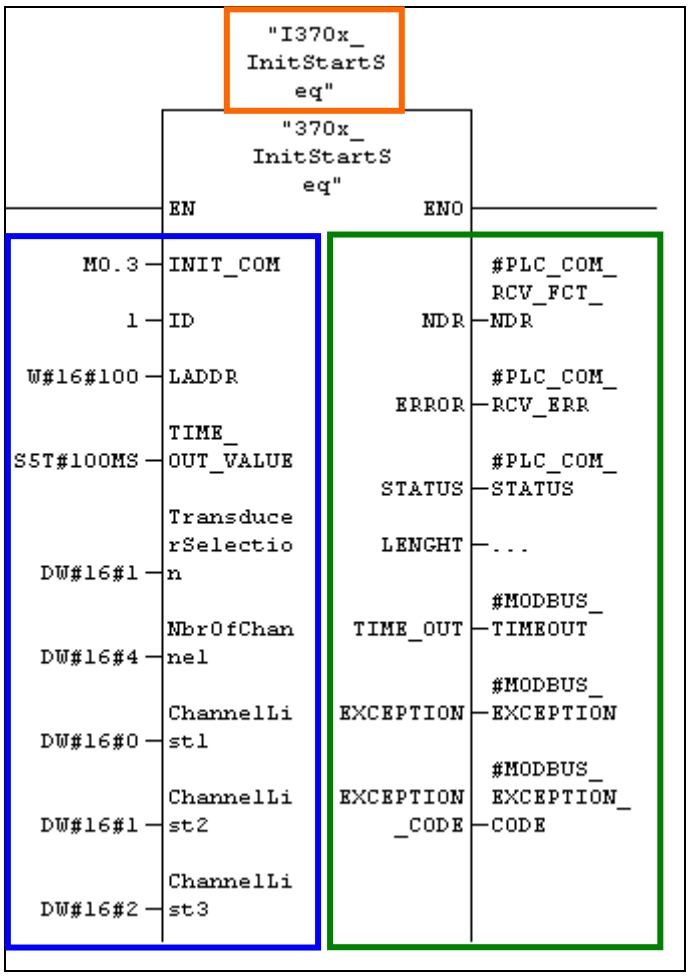
Due to limitations of the S7 platform, some names of function and parameter have been shortened in the AWL and S7 code. This table summarizes the changes against the standard version as described above.

Function/Parameter	Renamed as
MXCommon__GetModuleType	GetModuleType
MXCommon__GetTime	GetTime
MXCommon__TestCustomerID	TestCustomerID
MX370x__getNumberOfChannels	370x_GetNbrOfChannels
MX370x__TransducerGetAutoRefreshValues	370x_GetAutoRefVal
MX370x__TransducerGetNbrOfType	370x_GetNbrOfType
MX370x__GetTransducerDatabaseCursor	370x_GetDataBaseCursor
MX370x__TransducerGetTypeInformation	370x_GetTypeInfo
MXCommon__SetHardwareTriggerFilterTime	SetHwTrigFiltTime
MXCommon__InitAndStartSynchroTimer	InitStartSyncTimer
MXCommon__StopAndReleaseSynchroTimer	StopRelSyncTimer
MXCommon__Reboot	Reboot
MXCommon__SetCustomerKey	SetCustomerKey
MX370x__TransducerInitAndStartAutoRefresh	370x_InitStartAutoRef
HardwareTriggerCount	HwTrigCount
HardwareTriggerFilterTime	HwTrigFilterTime
ByTriggerNbrOfSeqToAcquire	ByTrigNbrOfSeqToAcq
MX370x__TransducerInitAndStartSequence	370x_InitStartSeq
HardwareTriggerCount	HwTrigCount
HardwareTriggerFilterTime	HwTrigFilterTime
NbrMaxSequenceToTransfer	NbrMaxSeqToTransfer
ByTriggerNbrOfSeqToAcquire	ByTrigNbrOfSeqToAcq
MX370x__TransducerStopAndReleaseSequence	370x_StopRelSeq
MX370x__SetTransducerDatabaseCursor	370x_SetDataBaseCursor

2.3 Description of the function parameters

The function block of the MSX-E Modbus TCP function “370x_InitStartSeq” serves as an example.

Fig. 2-4: Part of the function block “370x_InitStartSeq”

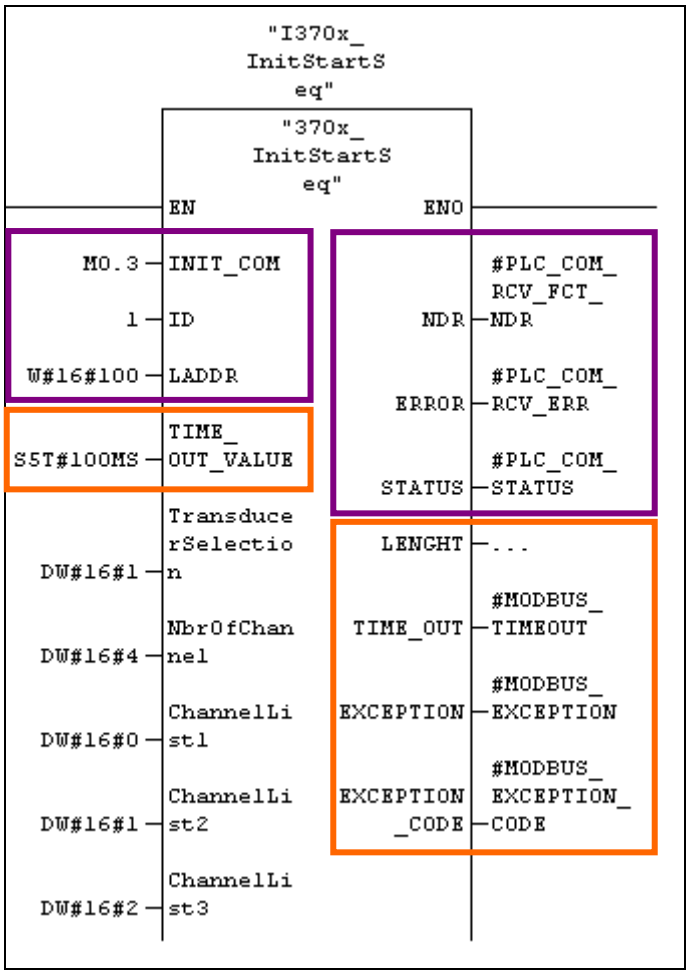


The input parameters **[Eingabe-Parameter]** are always indicated on the left; the output parameters **[Ausgabe-Parameter]** always on the right. The instance data block **[Instanz-Datenbaustein]** is to be found in the middle above.

2.3.1 PLC- and Modbus TCP-specific parameters

The MSX-E Modbus TCP functions contain **S7-specific parameters**. These parameters belong to the PLC functions “AG_SEND” and “AG_RECV” respectively.
The parameters highlighted in orange are **Modbus TCP-specific parameters**, which are identical with all MSX-E Modbus TCP functions.
The specific parameters are always capitalised.

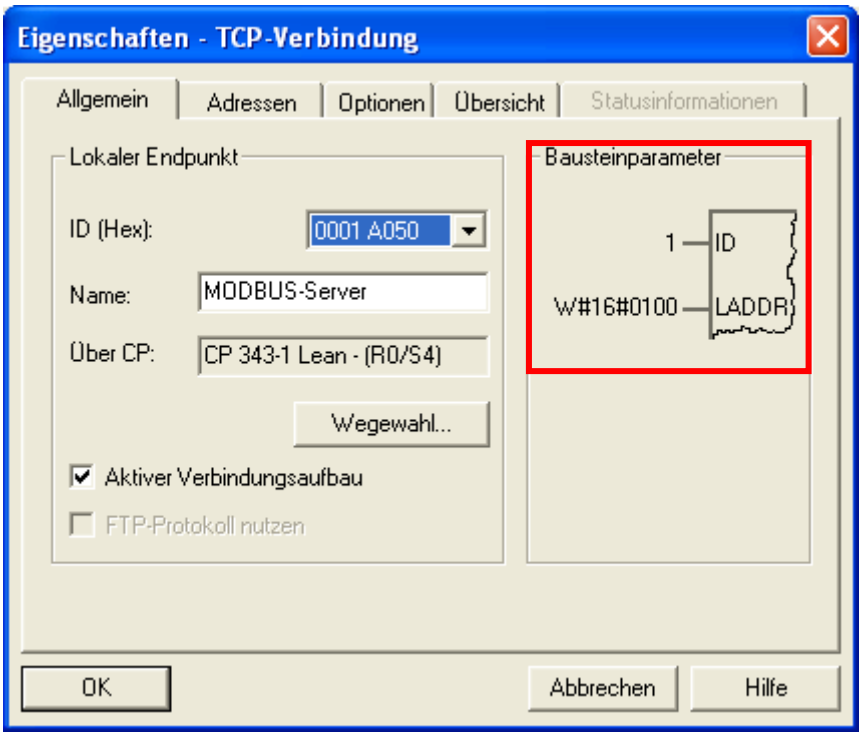
Fig. 2-5: Specific parameters of the function block „370x_InitStartSeq”



Input parameters:

- **INIT_COM:** If this bit is set to 1, the input parameters of the “AG_SEND” function are taken over. If this bit is set to 0, the output parameters are updated (see SIMATIC Manager Help: “AG_SEND ACT” parameters).
- **ID:** Communication ID relating to the Modbus server TCP connection (see NetPro properties)
- **LADDR:** Address of the blocks relating to the Modbus server TCP connection (see NetPro properties)

Fig. 2-6: TCP connection properties: Input parameters



- **TIME_OUT_VALUE:** You can set a time out whose value is greater than 0 (example: 50 ms = S5#T#50MS). This allows you to determine if the time between sending a Modbus request and receiving the response or exception through a partner device is within the defined time out. Moreover, the time out informs you if the network traffic is too large or if there is a disconnection.

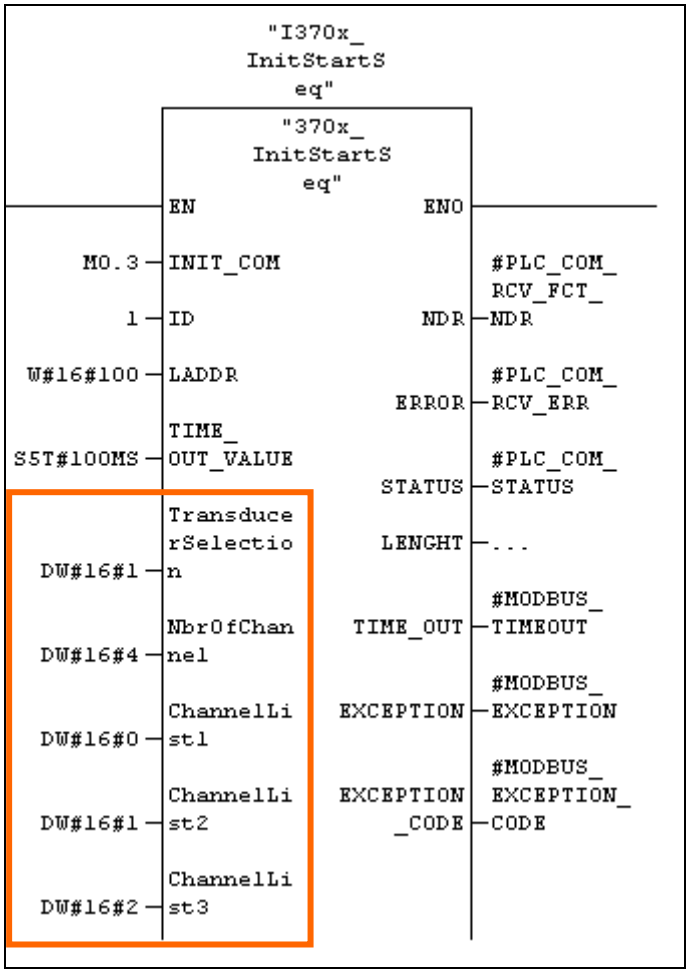
Output parameters:

- **NDR:** If the function call (sending a Modbus request and receiving a response or exception) is completed, **N**ew **D**ata **R**eceive is set to TRUE; otherwise, it is set to FALSE.
- **ERROR:** Response if an error occurs with the "AG_RECV" function
- **STATUS:** This parameter returns the error status of the "AG_RECV" function (see SIMATIC Manager Help: "AG_RECV").
- **LENGTH:** This parameter indicates the length in bytes of the received Modbus telegrams (only "payload" from the response telegram without any header).
- **TIME_OUT:** After a time out, TRUE is displayed (see **TIME_OUT_VALUE**).
- **EXCEPTION:** In case of a Modbus error, TRUE is displayed.
- **EXCEPTION_CODE:** This code can be read to get the error number. The error numbers are explained in the "msxeXXXX_modbus.html" documentation (see Chapter 1.3), in the Chapter "Exception code description".
  If the **EXCEPTION_CODE** equals 9, no Modbus error has occurred, but an error in the MSX-E system. By means of the MSX-E common function "GetLastCommandStatus", you can view details about this error.

2.3.2 General parameters of the MSX-E Modbus TCP functions

All the parameter names containing small letters are general parameters of the MSX-E Modbus TCP functions.

Fig. 2-7: General parameters of the function block “370x_InitStartSeq”



The description of the general parameters is to be found in the “msxeXXXX_modbus.html” documentation (see Chapter 1.3), in the Chapter “Siemens STEP 7 compatibility information (AWL/STL/SDF code)”.

Notes on the function names are available in Chapter 2.2 of this manual.

Example:

The PLC function block “370x_InitStartSeq” corresponds to the MSX-E Modbus TCP function “MX370x__TransducerInitAndStartSequenceEx”.

The input parameters are marked by the prefix “[Query frame layout]”.

Fig. 2-8: 370x_InitStartSeq: Description of the input parameters

3.16 Function MX370x__TransducerInitAndStartSequenceEx

Description

Initialise and start the transducer sequence acquisition mode

Parameters:

[Query frame layout] **TransducerSelection :** Transducer type selection

[Query frame layout] **NbrOfChannel :** Number of channel in the sequence

[Query frame layout] **ChannelList :** List of the channel index (0 to MaxChannel-1) who compose the sequence

[Query frame layout] **DivisionFactor :** Division factor (min: 5, max: 255)

[Query frame layout] **NbrOfSequence :** Number of sequence to acquire :

- 0 : continuous mode
- > 0 : number of sequence

[Query frame layout] **NbrMaxSequenceToTransfer :** This parameter defined the minimal number of sequences to acquire between each send of data by the modul.

Warning : They are two possibilities that the number of sequences sent doesn't reach the minimal number:
- By the end of the acquisition.
- If the memory capicity is not big enough.

[Query frame layout] **DelayMode :** Delay Mode :

- ADDIDATA_DELAY_NOT_USED 0 : Delay is not used.
- ADDIDATA_DELAY_MODE1_USED 1 : The delay time defines the time between 2 sequence beginnings.
- ADDIDATA_DELAY_MODE2_USED 2 : The delay time defines the time between the end of a sequence until the beginning of the next sequence.

[Query frame layout] **DelayTimeUnit :** Selection of the delay time unit

- 0: ms
- 1: s

The output parameters include the prefix “[Response frame layout]”.

Fig. 2-9: 370x_InitStartSeq: Description of the output parameters

2.10 Function MX370x__getNumberOfChannelsEx

Description

Return the number of transducer channels on the module (4,8 or 16)

Parameters:

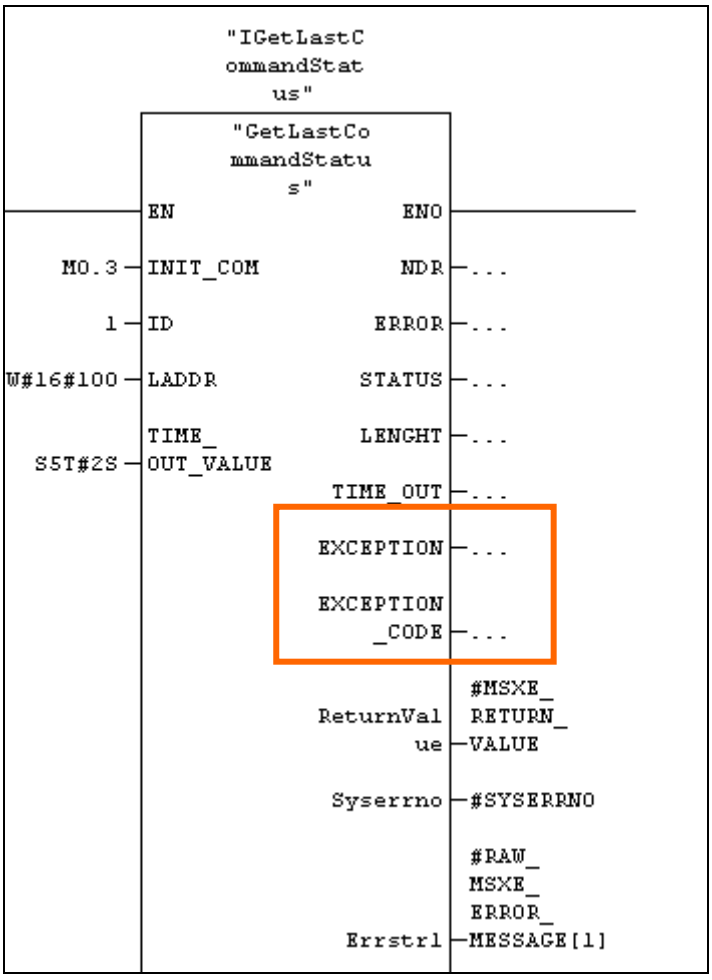
[Response frame layout]***ChannelNumber:*** Number of channels

2.3.3 MSX-E system errors

To find out if a function has produced an error after it had been called up, you can check the output parameters **EXCEPTION** and **EXCEPTION_CODE**:

- **EXCEPTION:** In case of a Modbus error, TRUE is displayed.
- **EXCEPTION_CODE:** This code can be read to get the error number. The error numbers are explained in the "msxeXXXX_modbus.html" documentation (see Chapter 1.3), in the Chapter "Exception code description".
  If the **EXCEPTION_CODE** equals 9, no Modbus error has occurred, but an error in the MSX-E system. By means of the MSX-E common function "GetLastCommandStatus", you can view details about this error.

Fig. 2-10: GetLastCommandStatus: Function block



- **ReturnValue:** This parameter returns a raw value that is always equal to or greater than 0.
  If the function parameter **TransducerSelection** in the function block **InitStartSeq** is not correctly indicated, **ReturnValue** equals 0xfffffffe. The return value is calculated as follows:

ReturnValue - 0x100000000 = Return value

Example:
0xfffffffe - 0x100000000 = -2

The return values can be checked in the “msxeXXXX_modbus.html” documentation (see Chapter 1.3), in the description of the function “MX370x__TransducerInitAndStartSequenceEx”.

Fig. 2-11: InitStartSeq: Return values

Returns:

Possible return value on the remote system (read them with GetLastCommandStatus)

- 0 : success
- -1: means an system error occured
- -2: Tranducer selection error
- -3: Number of channel error
- -4: Channel array selection error
- -5: Division factor error
- -6: Incorrect value for Hardware Trigger Mode
- -7: Incorrect value for Hardware Trigger Front
- -8: Incorrect value for Synchro Trigger Mode
- -9: Incorrect value for Hardware Trigger Count
- -10: Incorrect value for Hardware Trigger filter time
- -11: Incorrect value for "trigger number of sequences to acquire"
- -12: Delay Mode selection error
- -13: Delay time unit selection error
- -14: Delay value
- -15: Wrong data format parameter (ulOption1)
- -16: A value for Hardware Trigger front was defined but Hardware Trigger Mode is not set
- -17: Cannot use both triggers at the same time
- -18: Incorrect value for the hardware trigger stop front
- -19: Hardware trigger stop can not be used by this configuration of hardware trigger start
- -100: TransducerInit kernel function error
- -101: InitConvertTimeDivisionFactor kernel function error
- -102: InitEnableDisableSequenceDelay kernel function error
- -103: InitDigitalInputFilter kernel function error
- -104: InitEnableDisableHardwareTrigger kernel function error
- -105: InitEnableSynchroTrigger kernel function error
- -106: DisableSynchroTrigger kernel function error
- -107: SetTriggerSequenceCount kernel function error
- -108: InitSequence kernel function error
- -109: StartStopSequence kernel function error

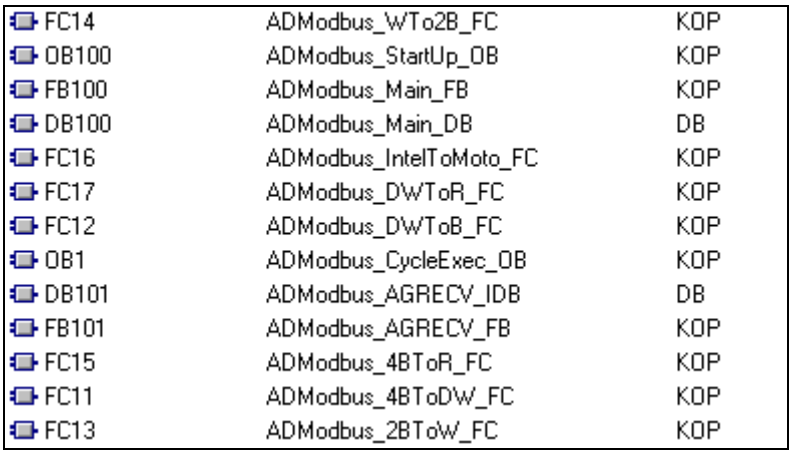
- **Syserrno:** This parameter works like **ReturnValue**. If a Syserrno error occurs, you can read the error message via the output parameter **ErrstrX**.
- **ErrstrX:** Each ErrstrX parameter corresponds to an ASCII value or a letter of the error message. If the ASCII code of the error message is decoded, you receive the error message in letters.

3 ADModbus blocks

3.1.1 Use

In the programming sample, the ADModbus blocks are available as PLC blocks. These are used to automatically manage the Modbus TCP protocol between the PLC and the MSX-E systems.

Fig. 3-1: ADModbus blocks



If you use the MSX-E Modbus TCP functions, the ADModbus blocks are processed in the background.

3.1.2 OB1 and OB100

Depending on the project, you need to adapt the organisation blocks OB1 and OB100.

NOTICE!

The ADModbus library uses the flags M0.1, M0.2, M0.3. If these are already being used by another application, you have to make sure that the alternative flags are defined like in OB1 or OB100 of the programming sample.

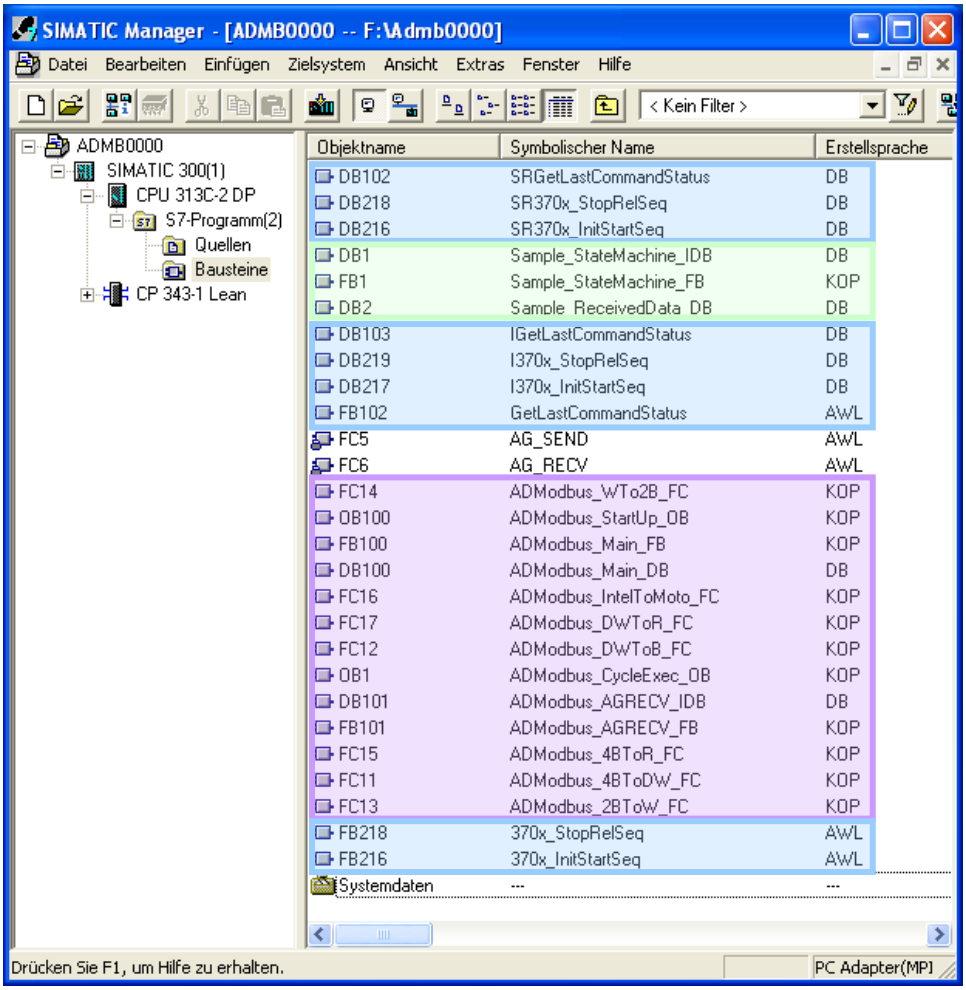
4 Integration of the ADModbus library

4.1 Functions and blocks

A part of the ADModbus library is supplied in the programming sample. You can use this sample as a project basis, or you can import or copy the required MSX-E Modbus TCP functions into your own project.

The following figure refers to a sequence acquisition of the Ethernet system **MSX-E370x**.

Fig. 4-1: ADModbus library: Functions and blocks



- All the blocks with the "ADModbus" prefix (see purple highlighting) belong to the ADModbus blocks and are supplied as Siemens blocks in the programming sample.
- The MSX-E Modbus TCP functions are supplied as AWL/STL source code files. The blocks highlighted in blue are already supplied as blocks in the programming sample. The remaining MSX-E Modbus TCP functions can be imported into the SIMATIC Manager and be compiled as blocks (see Chapter 6).
- The Siemens blocks highlighted in green belong to the programming sample of the MSX-E system.

4.2 Requirements

Please ensure that the following requirements are fulfilled:

- Siemens **CPU313C-2DP** (or compatible CPU)
- Siemens **CP343-1 Lean** (or compatible Ethernet system for the PLC)
- FC5 (AG_SEND) for **S7-300** (function for asynchronous communication)
- FC6 (AG_RECV) for **S7-300** (function for asynchronous communication)
- S7 sample for **STEP 7**.

4.3 Integration into the SIMATIC Manager

As already mentioned in Chapter 4.1, you can use the programming sample as a project basis, or you can copy the required ADModbus blocks from the programming sample into your project.
To integrate the ADModbus library into the SIMATIC Manager, the following functions and blocks are needed:

- S7 functions: “AG_SEND” and “AG_RECV” (see SIMATIC Manager Help)
- All ADModbus blocks
- MSX-E Modbus TCP functions (as required)
- State machine to define the order of the function calls.

NOTICE!
The ADModbus library uses the flags M0.1, M0.2, M0.3. If these are already being used by another application, you have to make sure that the alternative flags are defined like in OB1 or OB100 of the programming sample.

5 PLC configuration

The configuration below is based on an **S7-300** PLC with **CP343-1**. Different menus may be displayed with your PLC, but the settings are the same.

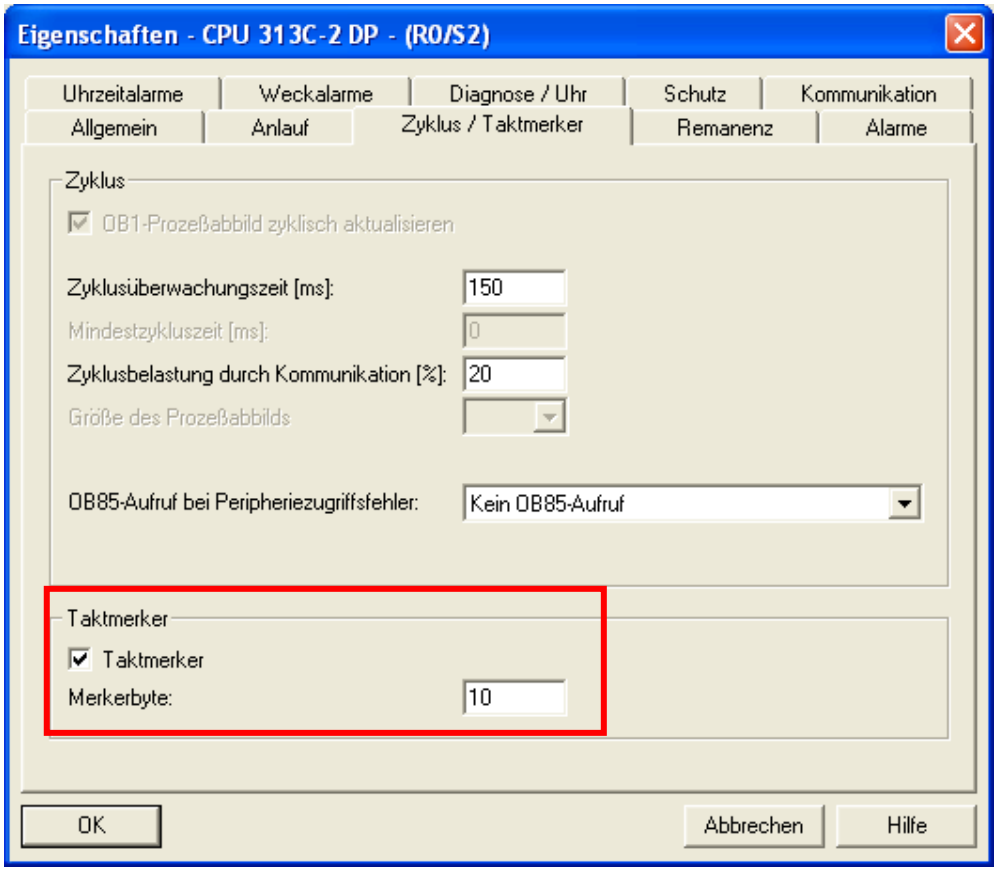
5.1 Clock memory

To use the ADModbus library, the clock memory needs to be configured first.

- Click on “Station SIMATIC”.
- Double-click on “Hardware”.
- In the “Baugruppe” [Device] column, right-click on “CPU” and select the menu item “Objekteigenschaften” [Object properties].

The following window is displayed:

Fig. 5-1: SIMATIC Manager: S7 clock memory configuration



- With “Memory byte”, enter the value “10”.

If you are already using a clock memory, you can adapt the following ADModbus blocks:

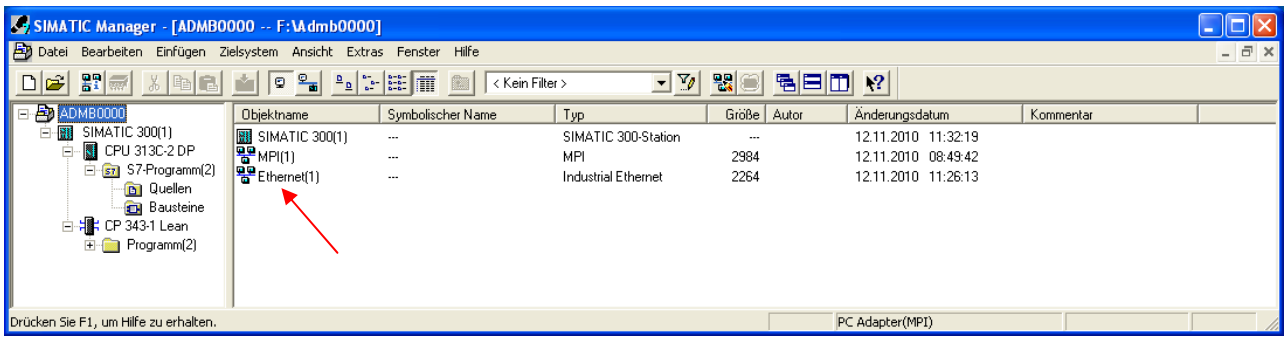
- ADModbus_Main_FB
- ADModbus_StartUp_OB.

5.2 TCP connection

5.2.1 Modbus server

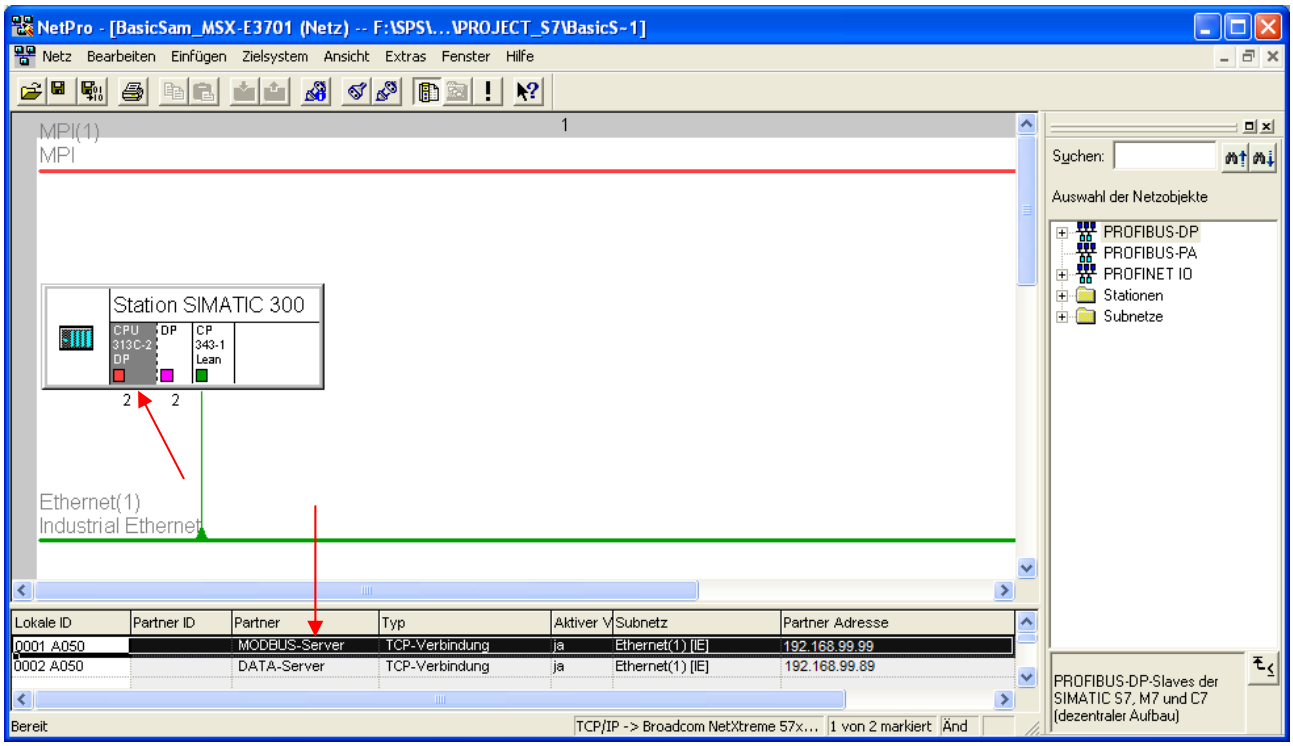
By default, the ADModbus library needs a TCP connection to access the Modbus server of the MSX-E system.

Fig. 5-2: Ethernet configuration: TCP connection



- Open the SIMATIC manager and then the “ADMB0000” project.
- To configure the TCP connection, double-click on the object name (Objektname) “Ethernet” (see arrow).

Fig. 5-3: Ethernet configuration: Modbus server

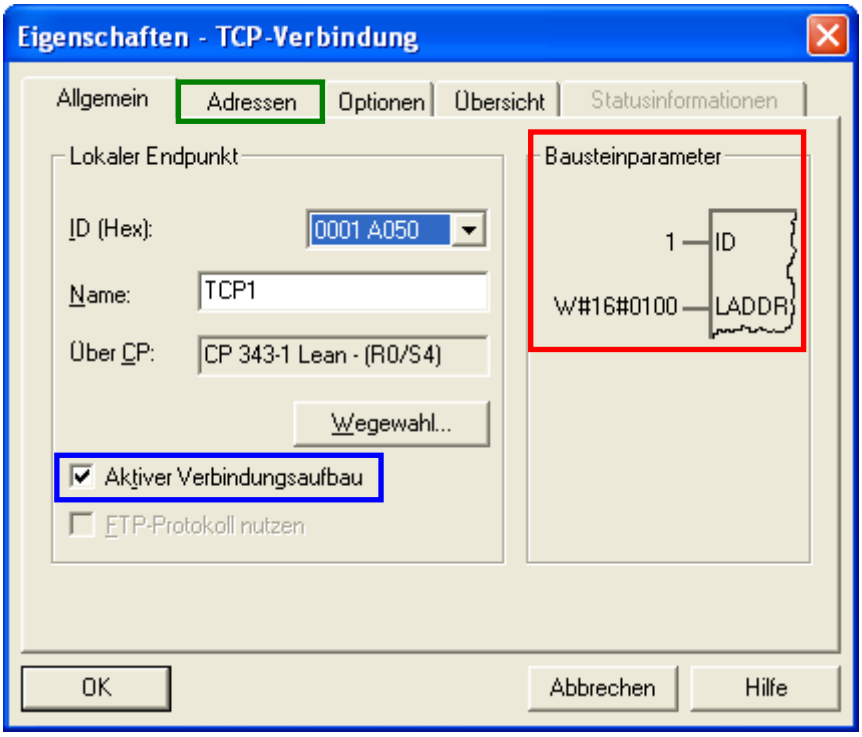


- Click on “CPU313C-2DP” (see top arrow).
- In the selected line of the table at the bottom, double-click on “MODBUS-Server” (see bottom arrow).

The table column “Typ” [Type] says that the Ethernet connection is a TCP connection (TCP-Verbindung).

Afterwards, the following window is displayed:

Fig. 5-4: Properties – TCP connection: General Information

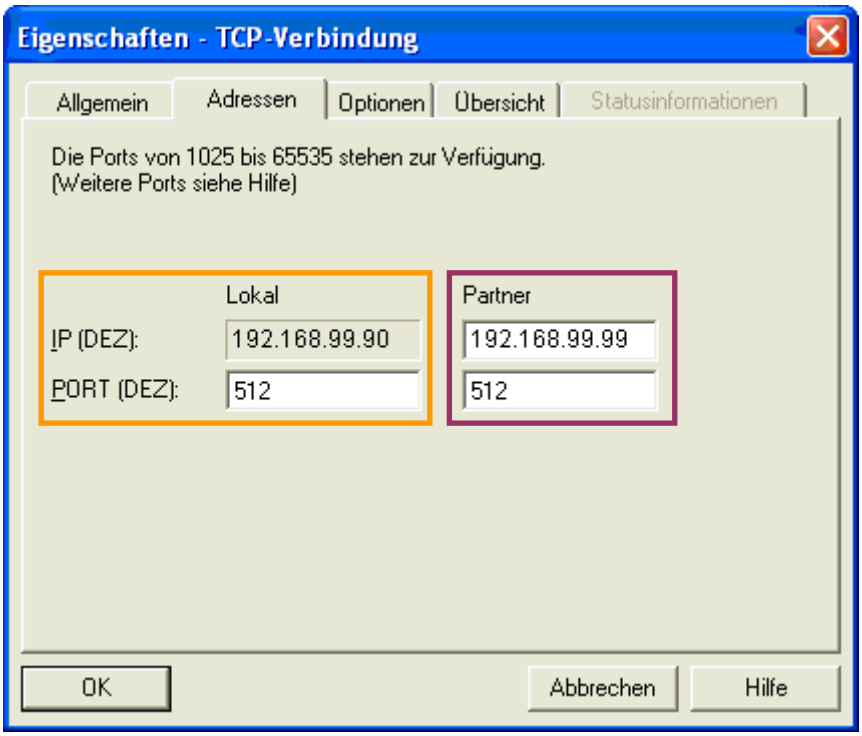


In order for the S7 to create the connection to the MSX-E system, **Aktiver Verbindungsaufbau** [Active connection establishment] must be selected.

On the right-hand side of the window, you see the block parameters **(Bausteinparameter)** that have to be used together with the function block "ADModbus_Main_FB". The connection ID (socket ID) and the address are specified.

- Click on the **Adressen** [Addresses] tab.

Fig. 5-5: Properties – TCP connection: Addresses



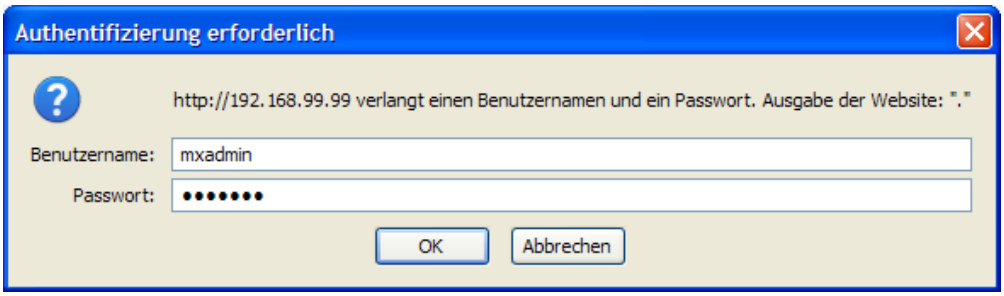
- In the **Lokal** [Local] area, in the "PORT" field, enter the port number of the S7. This must correspond to the port number of the MSX-E system and can be used only once.
  In order to use several MSX-E systems, a local port number has to be indicated for each system.

- In the **Partner** [Remote] area, in the "IP" field, enter the IP address of the MSX-E system, and in the "Port" field, the port number of the MSX-E system.
  By default, the partner port number is always set to 512 as this port is reserved for Modbus communication. This port number must correspond to the one that has been defined on the web interface of the MSX-E system.

- To check the port number of the MSX-E system, open a web browser (such as Mozilla Firefox, Internet Explorer, etc.) and enter the following address: "http://[IP address of the MSX-E system]".

A login window is displayed:

Fig. 5-6: Login window



- Enter “mxadmin” as the user name and password.
- On the web interface of the MSX-E system, click on the menu item “Modbus server”.

Fig. 5-7: Modbus server: Configuration

- In the “Configuration” section, with “Endianness”, select the “Big” option, and with “Protocol”, the “TCP/IP” option.
- In the “Port number” field, change the port number if you cannot use the default port number (512).

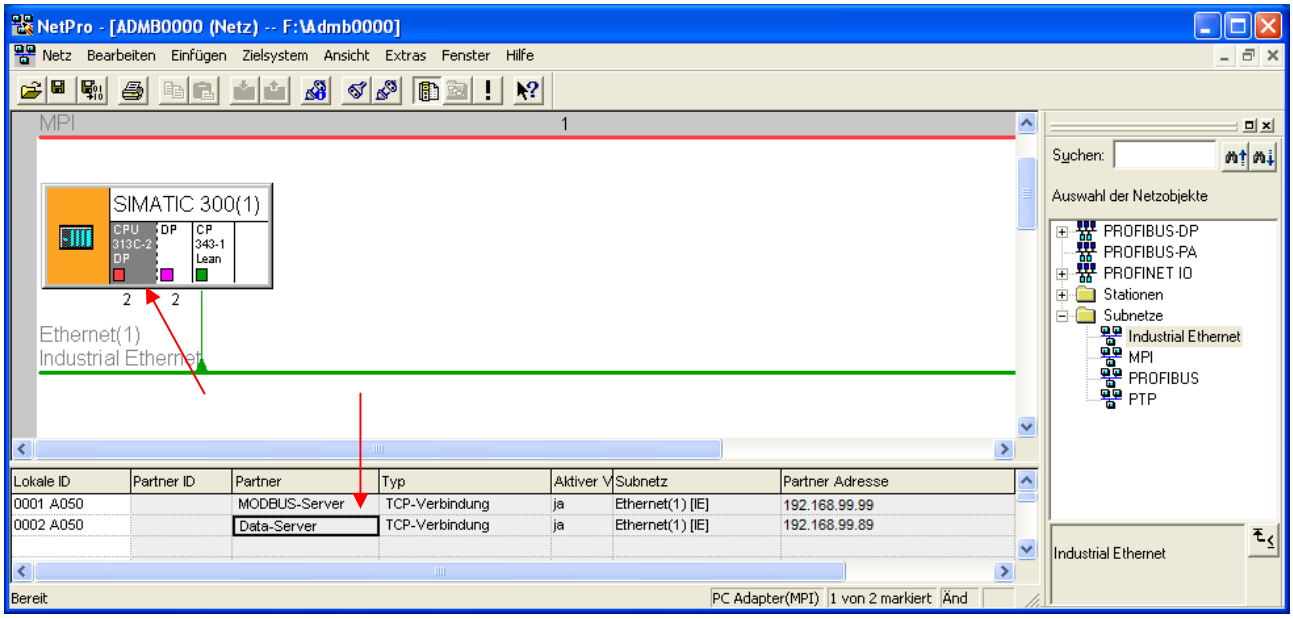
Fig. 5-8: Modbus server: Set and save

- After changing the port number, click on “Set and Save” and then on “Reload”.

5.2.2 Data server

If you want to read the measurement values from the data server of the MSX-E system, you need to set up another TCP connection because the Modbus server is used only for commands.

Fig. 5-9: Ethernet configuration: Data server

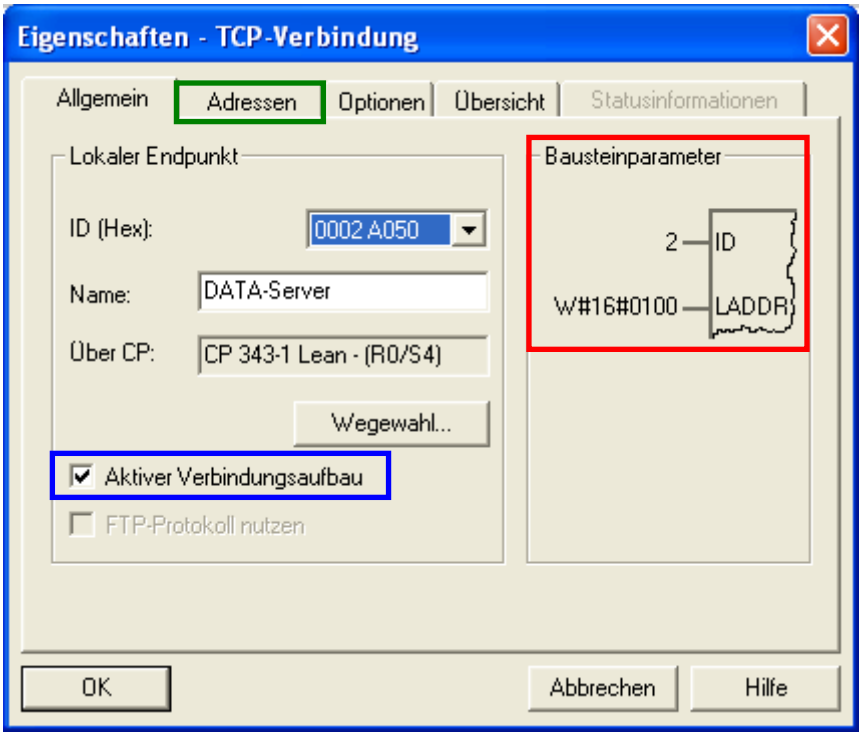


- Click on “CPU313C-2DP” (see top arrow).
- In the selected line of the table at the bottom, double-click on “Data-Server” (see bottom arrow).

The table column “Typ” [Type] says that the Ethernet connection is a TCP connection (TCP-Verbindung).

Afterwards, the following window is displayed:

Fig. 5-10: Properties – TCP connection: General information

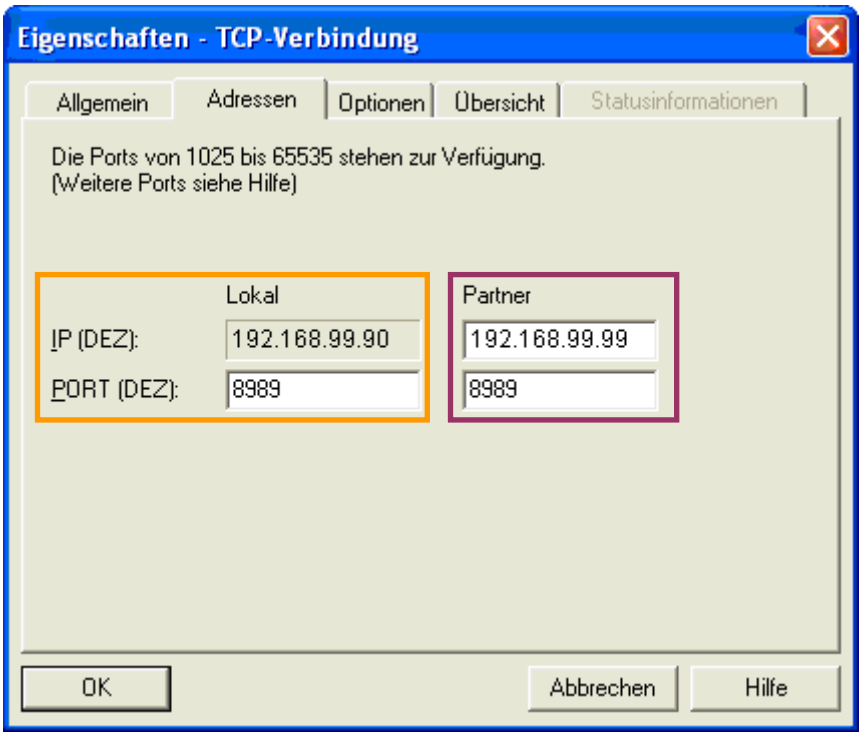


In order for the S7 to create the connection to the MSX-E system, **Aktiver Verbindungsaufbau** [Active connection establishment] must be selected.

On the right-hand side of the window, you see the block parameters **(Bausteinparameter)** that have to be used together with the "FC6 (AG_RECV)" function. The connection ID (socket ID) and the address are specified.

- Click on the **Adressen** [Addresses] tab.

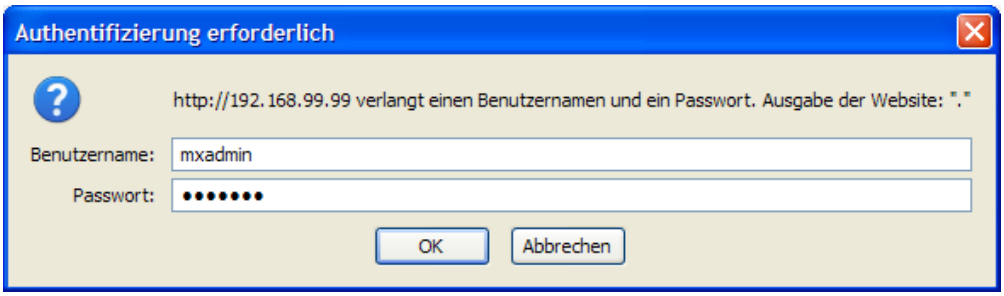
Fig. 5-11: Properties – TCP connection: Addresses



- In the **Lokal** [Local] area, in the "PORT" field, enter the port number of the S7. This must correspond to the port number of the MSX-E system.
- In the **Partner** [Remote] area, in the "IP" field, enter the IP address of the MSX-E system, and in the "Port" field, the port number of the MSX-E system. This port number must correspond to the one that has been defined on the web interface of the MSX-E system.
- To check the port number of the MSX-E system, open a web browser (such as Mozilla Firefox, Internet Explorer, etc.) and enter the following address: "http://[IP address of the MSX-E system]".

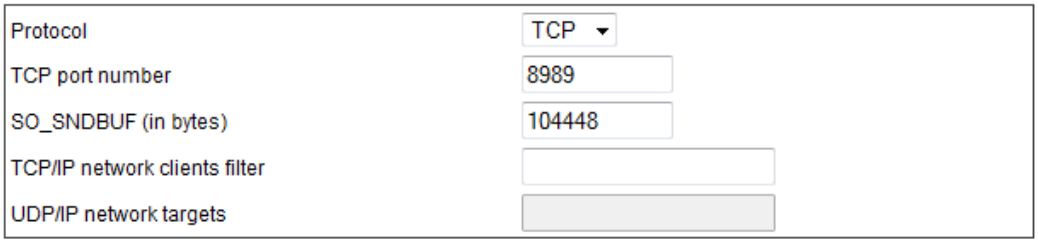
A login window is displayed:

Fig. 5-12: Login window



- Enter "mxadmin" as the user name and password.
- On the web interface of the MSX-E system, click on the menu item "Data server".

Fig. 5-13: Data server: Configuration



- In the “Configuration” section, in the “TCP port number” field, change the port number if you cannot use the default port number (8989).

Fig. 5-14: Data server: Set and save

- After changing the port number, click on “Set and Save” and then on “Reload”.
- Click on the “Blocking transfer” tab.

Fig. 5-15: Data server: Blocking transfer

- Ensure that in the “Configuration” section, with “Activate blocking TCP/IP transfer”, the “No” option is selected.
- When you have made this change, click on “Set and Save” and then on “Reload”.

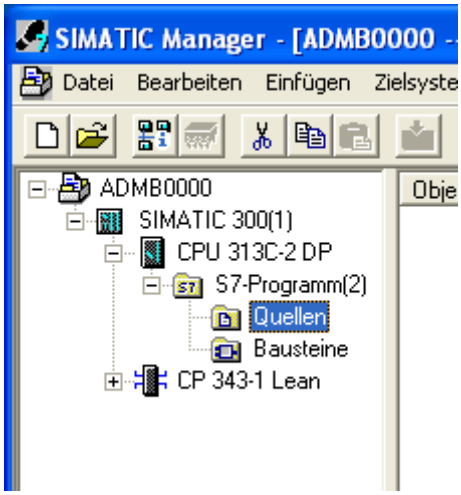
6 Function blocks

6.1 Implementation of AWL/STL source code files into function blocks

The MSX-E Modbus TCP functions that are not available in the programming sample can be imported into the SIMATIC Manager via the AWL/STL source code files.
Each MSX-E Modbus TCP function which is implemented in the MSX-E system has a function block afterwards that accesses the function together with an instance data block and a global data block. As there is an AWL/STL source code for each MSX-E Modbus TCP function, three blocks are therefore implemented in this code. The AWL/STL source codes can be opened and compiled in the SIMATIC Manager.

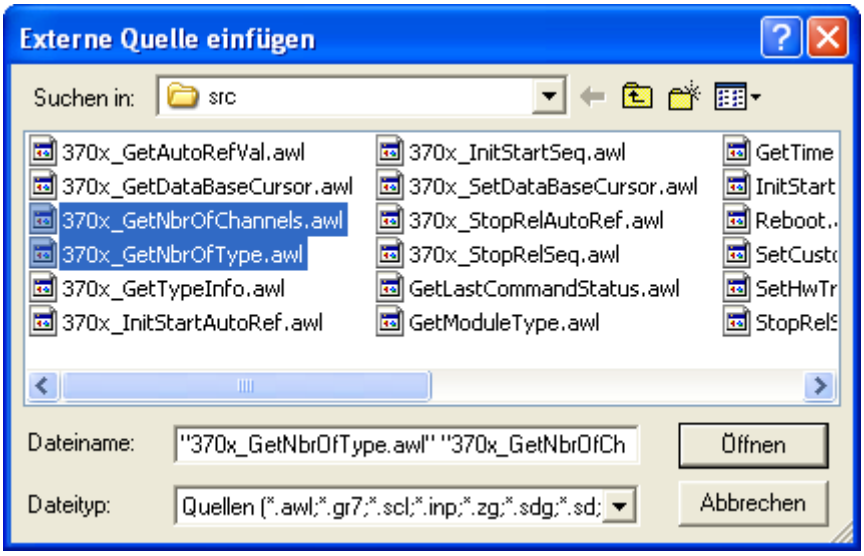
- In the SIMATIC Manger, right-click on “Quellen” [Sources].
- Select the menu item “Neues Objekt einfügen” [Insert new object] and then click on “Externe Quelle” [External source].

Fig. 6-1: SIMATIC Manager: Source directory



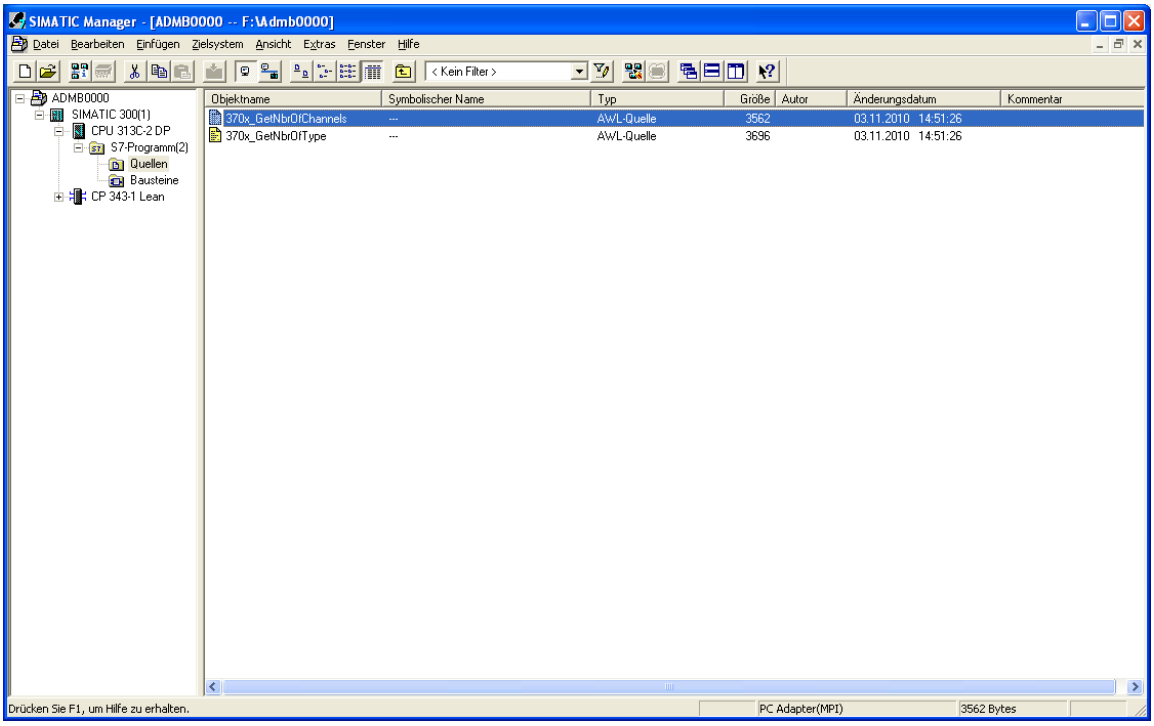
- Select the folder in which the AWL/STL source code files containing the source codes of the MSX-E Modbus TCP functions are to be found. Select the files to be imported and click on “Öffnen” [Open].

Fig. 6-2: SIMATIC Manager: Selection of the AWL/STL source code files



The selected AWL/STL source code files are now stored in the “Quellen” [Sources] folder.

Fig. 6-3: SIMATIC Manager: AWL/STL source code files



- To import single MSX-E Modbus TCP functions into the SIMATIC Manager, double-click on the respective AWL/STL source code file. This allows you to access the AWL/STL source code.

NOTICE!

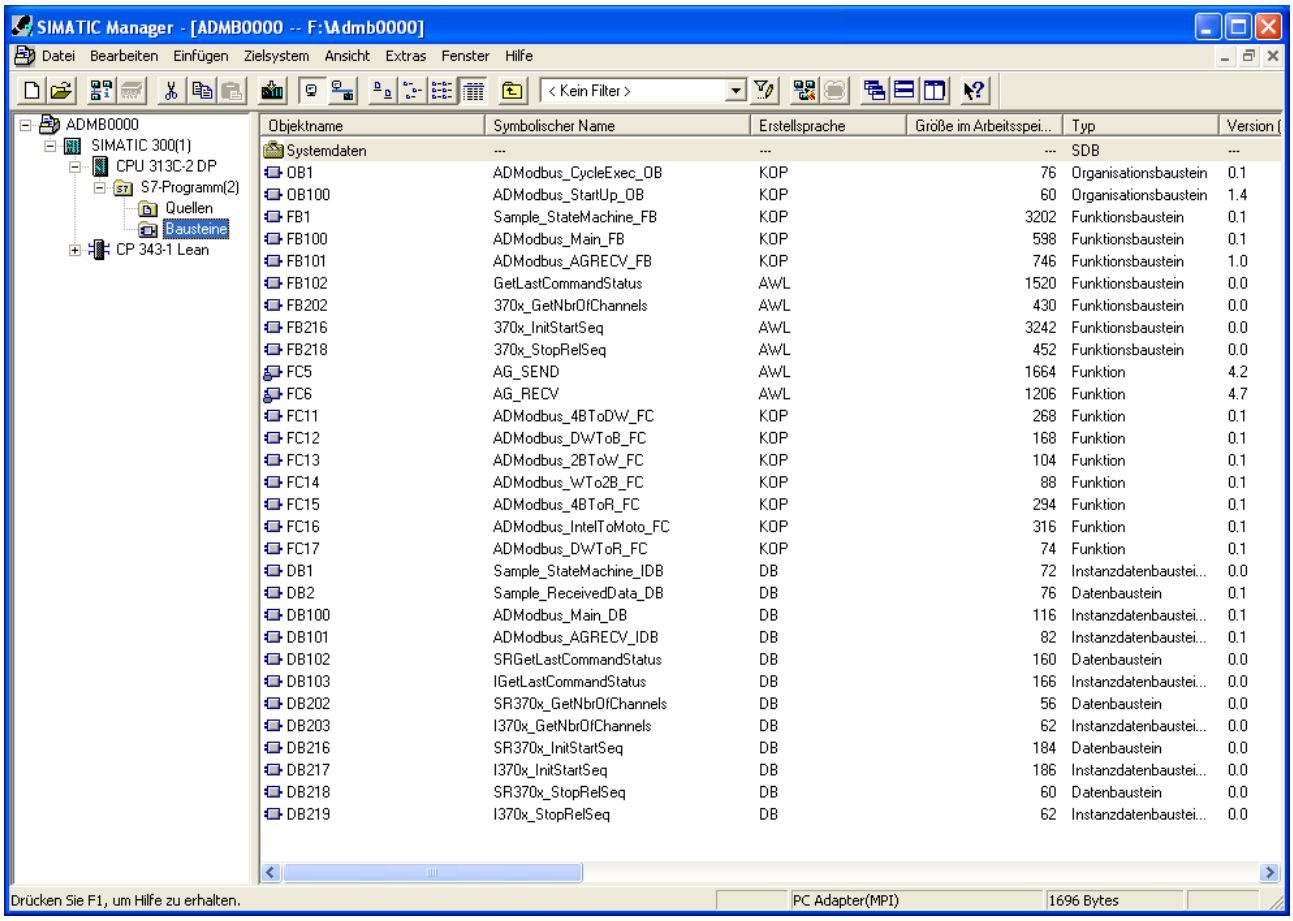
The numbering of the MSX-E Modbus TCP functions is identical for all MSX-E systems even if the functions are different.
If the AWL/STL source code file is opened, you can modify the number of each of the three blocks. Please note that a block number may be assigned only once, as otherwise, the block with the same number is overwritten during the AWL/STL compilation!
Each MSX-E Modbus TCP function is indexed as follows (example):

- **Function block: FB202 370x_GetNbrOfChannels**
- **Global data block: DB202 SR370x_GetNbrOfChannels (query/response Modbus TCP telegrams)**
- **Instance data block: DB203 I370x_GetNbrOfChannels.**

- After completing the source code modifications in a file, in the menu bar, click on “Datei” [File] and select the menu item “Übersetzen” [Compile].

The compiled files are now stored as blocks in the “Bausteine” [Blocks] folder.

Fig. 6-4: SIMATIC Manager: Blocks



6.2 Mnemonic symbols

For each MSX-E system, there is a symbol file (.sdf), in which all available symbols or MSX-E Modbus TCP functions with three blocks each are listed.
The symbol file can be used, for example, to name blocks. This means that the symbolic name is used instead of the block number from the source code. For this, you have to import the supplied file “mnemonics.sdf” into the project via the Symbol Editor.

- Open the Symbol Editor, and in the menu bar, click on “Tabelle” [Table].
- Select the menu item “Importieren” [Import] and then the “mnemonics.sdf” file. After that, click on “Öffnen” [Open].
- In the displayed symbol file, select the required functions and delete those that are not to be used.

Fig. 6-5: SIMATIC Manager: Mnemonic symbols

Symbol Editor - [S7-Programm(2) (Symbole) -- ADMB0000\SIMATIC 300(1)\CPU 313C-2 DP]

Tabelle Bearbeiten Einfügen Ansicht Extras Fenster Hilfe

Alle Symbole

	Status	Symbol	Adresse	Datentyp	Kommentar
1		370x_GetNbrOfChannels	FB 202	FB 202	
2		370x_InitStartSeq	FB 216	FB 216	MX370x__TransducerInitAndStartSequence
3		370x_StopRelSeq	FB 218	FB 218	MX370x__TransducerStopAndReleaseSequence
4		ADModbus_2BToW_FC	FC 13	FC 13	
5		ADModbus_4BToDW_FC	FC 11	FC 11	
6		ADModbus_4BToR_FC	FC 15	FC 15	
7		ADModbus_AGRECV_FB	FB 101	FB 101	
8		ADModbus_AGRECV_IDB	DB 101	FB 101	
9		ADModbus_CycleExec_OB	OB 1	OB 1	
10		ADModbus_DWToB_FC	FC 12	FC 12	
11		ADModbus_DWToR_FC	FC 17	FC 17	
12		ADModbus_IntelToMoto_FC	FC 16	FC 16	
13		ADModbus_Main_DB	DB 100	FB 100	
14		ADModbus_Main_FB	FB 100	FB 100	
15		ADModbus_StartUp_OB	OB 100	OB 100	
16		ADModbus_WTo2B_FC	FC 14	FC 14	
17		AG_RECV	FC 6	FC 6	AG RECEIVE
18		AG_SEND	FC 5	FC 5	AG SEND
19		GetLastCommandStatus	FB 102	FB 102	GetLastCommandStatus
20		I370x_GetNbrOfChannels	DB 203	FB 202	
21		I370x_InitStartSeq	DB 217	FB 216	MX370x__TransducerInitAndStartSequence.Instance
22		I370x_StopRelSeq	DB 219	FB 218	MX370x__TransducerStopAndReleaseSequence.Instance
23		IGetLastCommandStatus	DB 103	FB 102	GetLastCommandStatusInstance
24		Instanz_DB_FB3	DB 3	DB 3	
25		Sample_ReceivedData_DB	DB 2	DB 2	
26		Sample_StateMachine_FB	FB 1	FB 1	
27		Sample_StateMachine_IDB	DB 1	FB 1	
28		SR370x_GetNbrOfChannels	DB 202	DB 202	
29		SR370x_InitStartSeq	DB 216	DB 216	MX370x__TransducerInitAndStartSequence.Send/Recevice
30		SR370x_StopRelSeq	DB 218	DB 218	MX370x__TransducerStopAndReleaseSequence.Send/Recevice
31		SRGetLastCommandStatus	DB 102	DB 102	GetLastCommandStatus.Send/Receive
32		STATUS	FB 5	FB 5	FMS STATUS
33					

Drücken Sie F1, um Hilfe zu erhalten. NUM

NOTICE!

After making the modifications, just click on “Speichern” [Save].

7 Reading data from the MSX-E data server

7.1 Data conversion

Please note that the MSX-E systems send data from the data server in the Little Endian format (Intel format), whereas an S7 PLC works with the Big Endian format (Motorola format).
As a result, the values of the MSX-E systems have to be converted in the PLC into the Big Endian format (Motorola format). This data coding plays an important role, as otherwise, the measurement results will be interpreted incorrectly. ADDI-DATA provides you with a corresponding conversion function:

- **Function:** FC1
- **Name:** ADModbus_IntelToMoto_FC

7.1.1 Little Endian format (Intel format

The least significant bit (LSB) is transferred first:

16-bit	0x1234	>>	0x34	0x12		
32-bit	0x12345678	>>	0x78	0x56	0x34	0x12

7.1.2 Big Endian Format (Motorola Format

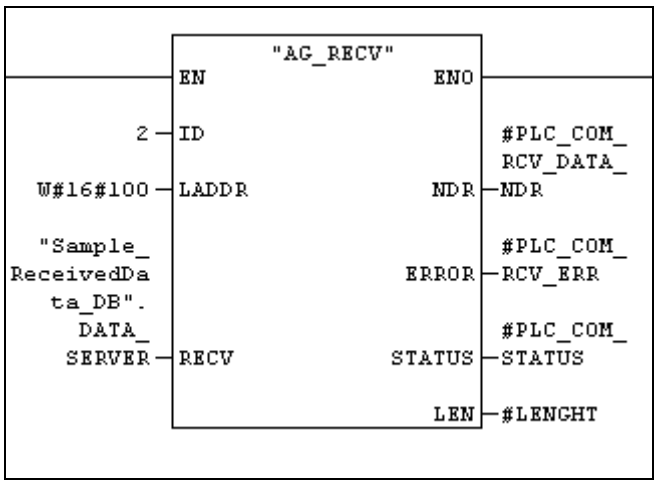
The most significant bit (MSB) is transferred first:

16-bit	0x1234	>>	0x12	0x34		
32-bit	0x12345678	>>	0x12	0x34	0x56	0x78

7.2 „FC6 (AG_RECV)“ function

Description:

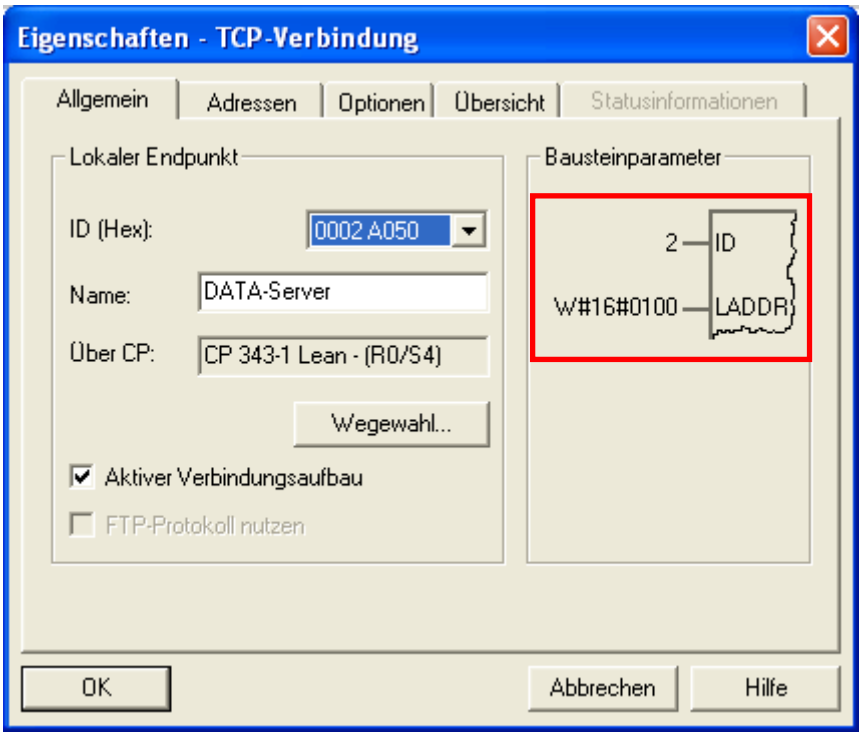
This function has been created by Siemens and enables you to read data from a network connection.



Input parameters:

- **ID:** 2 is the number of the data server’s communication identifier.
- **LADDR:** The value “W#16#0100” is the address of the function block.

Fig. 7-1: Properties – TCP connection: Input parameters



- **RECV:** Pointer in which the data has to be saved; in this case, in **Sample_ReceivedData_DB.DATA_SERVER**. This means that a pointer to the "DATA_SERVER" offset is passed to the DB2 data block ("Sample_ReceivedData_DB") and that 20 bytes (5 DWORDs) are expected. The number of bytes expected may change as required.
- **DATA_SERVER:** Table that is 5 DWORDs wide. Make sure that the table in which the values are saved is sufficiently wide. The number of bytes always corresponds to a data packet.

The size of a data packet is specified on the web interface of the MSX-E system:

- On the web interface, click on the menu item "Acquisition".
- Select the Auto-refresh or Sequence mode.

In the "Data server frame format" section, the size of the data packet is displayed.

Fig. 7-2: Acquisition: Data server frame format

Size	Field	Description
4 Bytes	Inductive transducer 1	Digital value, encoded on 24 bits.
4 Bytes	Inductive transducer 6	Digital value, encoded on 24 bits.
4 Bytes	Inductive transducer 3	Digital value, encoded on 24 bits.

When the function block has received the values from the MSX-E system, you will find them in the following order in the table of the target data block:

DATA_SERVER [0]: contains the value "channel"
DATA_SERVER [1]: contains the value "channel 1"
DATA_SERVER [2]: contains the value "channel 2"
…
DATA_SERVER [4]: contains the value "channel 8"

The following example is based on an S7 screenshot ("Data server frame format") and a data block containing the "DATA_SERVER" table as a DWORD data type with a width of 20 bytes.

Fig. 7-3: Data block with the "DATA_SERVER" table

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	DATA_SERVER	ARRAY[0..4]	DW#16#0	RAW Data
*4.0		DWORD		
+20.0	COUNT_MOTOROLA	DWORD	DW#16#0	Converted sequence counter
+24.0	CHANNEL	ARRAY[0..3]	0.000000e+000	Converted channels values
*4.0		REAL		
=40.0		END_STRUCT		

Fig. 7-4: "DATA_SERVER" table in the data block

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
0.0	DATA_SERVER[0]	DWORD	DW#16#0	DW#16#16000000	RAW Data
4.0	DATA_SERVER[1]	DWORD	DW#16#0	DW#16#00000040	
8.0	DATA_SERVER[2]	DWORD	DW#16#0	DW#16#01C835BA	
12.0	DATA_SERVER[3]	DWORD	DW#16#0	DW#16#015828BA	
16.0	DATA_SERVER[4]	DWORD	DW#16#0	DW#16#01582FBA	
20.0	COUNT_MOTOROLA	DWORD	DW#16#0	DW#16#00000016	Converted sequence counter
24.0	CHANNEL[0]	REAL	0.000000e+000	1.073742e+009	Converted channels values
28.0	CHANNEL[1]	REAL	0.000000e+000	-1.170881e+009	
32.0	CHANNEL[2]	REAL	0.000000e+000	-1.171761e+009	
36.0	CHANNEL[3]	REAL	0.000000e+000	-1.171302e+009	

Output parameters:

- **NDR:** If the number of bytes has been read and NDR equals 1, no error has occurred. If NDR equals 0, then check ERROR and STATUS to find the error that is displayed.
- **LEN:** Number of read bytes. The number must be 20 (= 5 x 4 bytes).

8 S7 programming sample

8.1 Introduction

ADDI-DATA provides you with a software package to allow for the connection between an MSX-E system and an S7 PLC via Modbus TCP. The programming sample is based on a **SIMATIC S7** from Siemens.

If you want to apply an S7-compatible PLC from a different manufacturer (such as IBHsoftec, VIPA, etc.), you have to use the relevant manufacturer's original FCs. You may also have to adapt the data blocks, etc. (Siemens: 5 DWORDs; VIPA: 20 bytes).

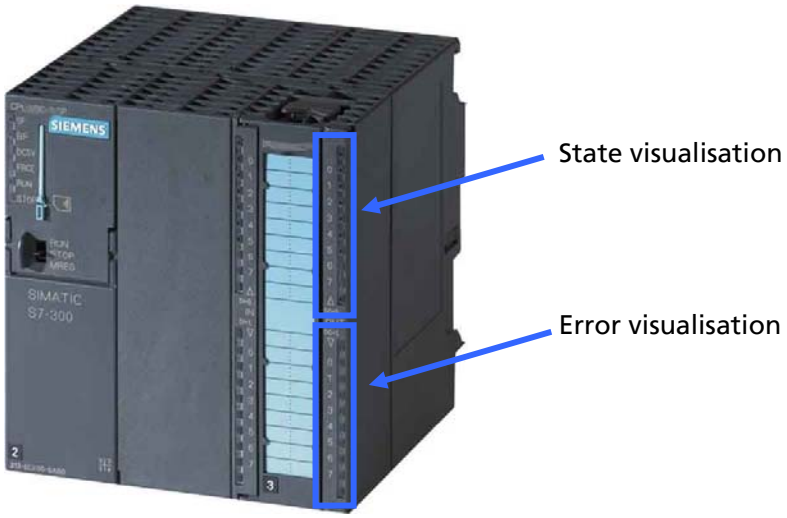
The programming sample „Sample MSXE-3701 (ADMB0000)" has been developed for the Ethernet communication of the **S7-300** with **CP343-1** via the Modbus TCP protocol. It serves for reading values from the TCP connection (TCP socket).

8.2 Requirements

The programming sample uses a state machine to manage the function calls. There is the possibility to visualise the current state of the digital PLC outputs by means of LEDs. These LEDs can be used as well to indicate errors and warnings during the development stage.

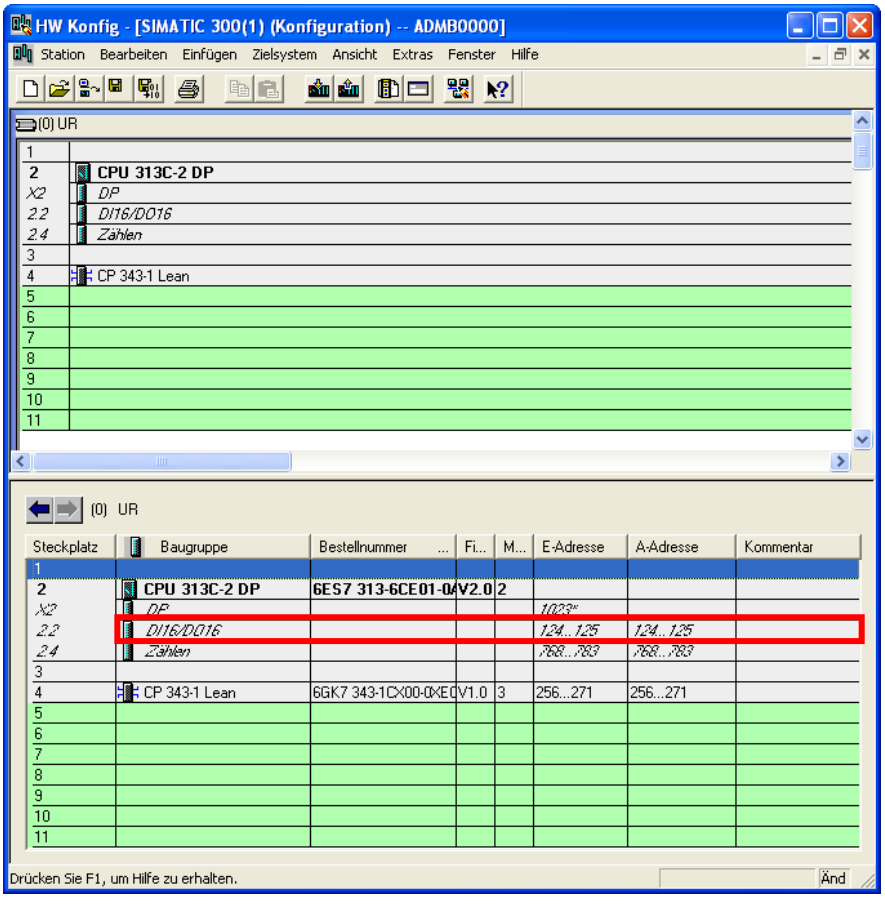
If you want to use the LEDs of the digital outputs, in the programming sample, you have to adapt the register address of the digital outputs to your hardware configuration.

Fig. 8-1: S7: LED visualisation of the digital outputs



The meaning of each LED is explained in the commentary of the programming sample.

Fig. 8-2: SIMATIC Manager: LED configuration of the digital outputs



OB1 calls up the state machine FB1, in which all the steps of the programming sample are described.

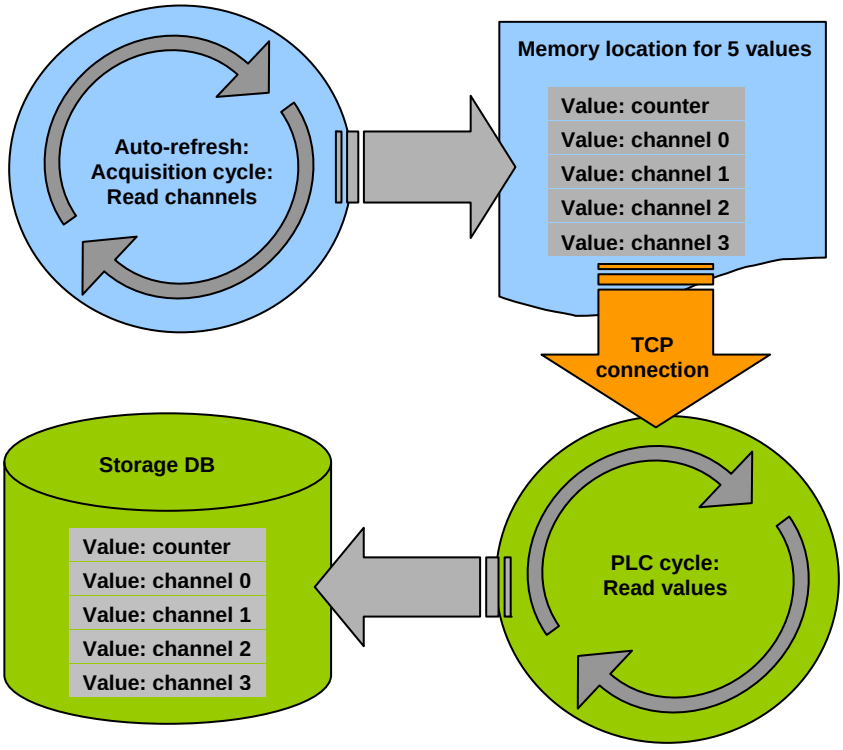
9 Auto-refresh and Sequence modes

NOTICE!

If you do not configure an average value calculation in Auto-refresh mode or a delay in Sequence mode, the PLC requires a certain time to read the values from the TCP connection (PLC cycle > acquisition cycle). This is also the case if a small “division factor” is configured in both modes. So this may result in a delay between the actual status of the channels and the measurement values that are read.
If the packets from the TCP connection are not read quickly enough, there may be a FIFO overrun which causes the MSX-E system to cancel the connection to the PLC.

9.1 Acquisition in Auto-refresh mode

Fig. 9-1: Auto-refresh mode: Acquisition cycle



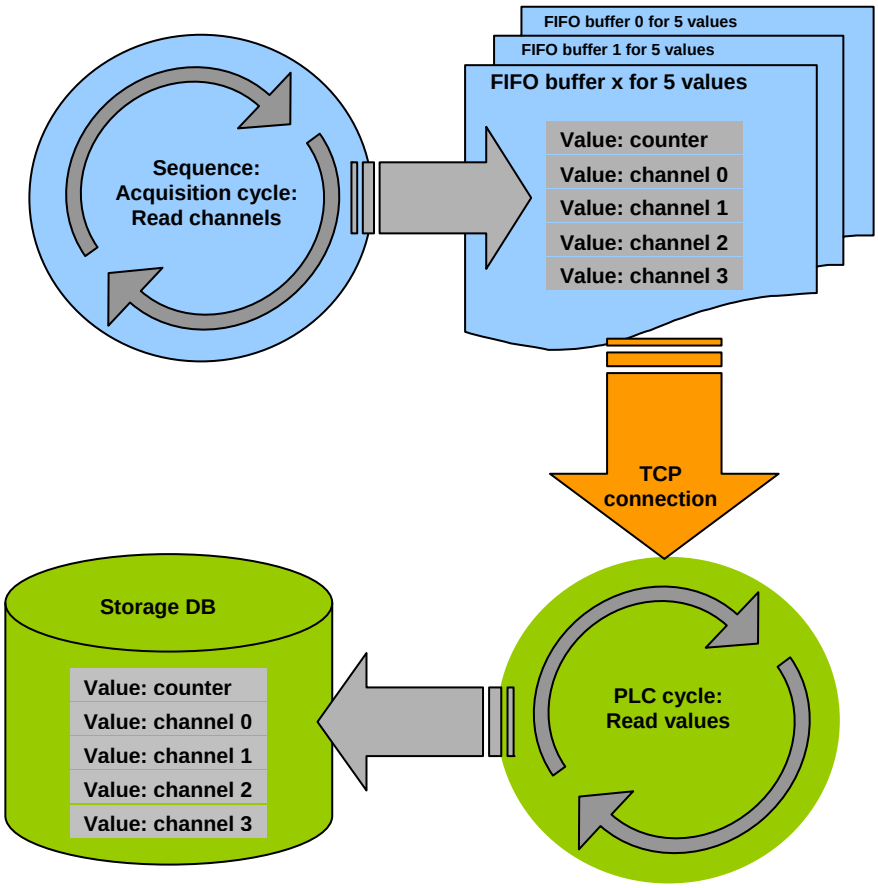
The acquisition cycle is displayed with four channels of an MSX-E system in Auto-refresh mode (light blue). As only one memory location is available for the five values, after each cycle, the values of the previous cycle are overwritten with those of the current cycle.
The MSX-E system sends the values to the PLC as soon as the PLC is connected. The values from the TCP connection are read in the PLC cycle (green).

MSX-E acquisition cycle (Auto-refresh mode) < PLC cycle -> values may be lost MSX-E acquisition cycle (Auto-refresh mode) > PLC cycle -> all values are received MSX-E acquisition cycle (Auto-refresh mode) = PLC cycle -> all values are received

NOTICE!
With a socket FIFO overrun, values may be lost.

9.2 Acquisition in Sequence mode

Fig. 9-2: Sequence mode: Acquisition cycle



The acquisition cycle is displayed with 4 channels of an MSX-E system in Sequence mode (light blue). The values of the current cycle are written into a new FIFO buffer, i.e. no values are overwritten. The MSX-E system sends the values to the PLC as soon as the PLC is connected. The values from the TCP connection are read in the PLC cycle (green).

MSX-E acquisition cycle (Sequence mode) < PLC cycle -> all values are received
MSX-E acquisition cycle (Sequence mode) > PLC cycle -> all values are received
MSX-E acquisition cycle (Sequence mode) = PLC cycle -> all values are received

NOTICE!
With a socket FIFO overrun, values may be lost.

10 Appendix

10.1 Glossary

AWL
= Anweisungsliste

AWL is a programming language according to IEC 61131-3 for PLC programming.
It corresponds to the instruction list language STL for Siemens PLCs.

Code block
In the Siemens STEP 7 system, code blocks correspond to organisation units of the user program. A distinction is drawn between the following code block types: organisation block (OB), function block (FB) and function (FC).

Counter
A counter is a circuit which counts pulses or measures pulse duration.

Data block (DB)
Data blocks are data areas in the Siemens STEP 7 system, in which the data of the user program is stored. There are two data block types: instance data blocks and global data blocks. The former contain static local data, the latter contain information that can be accessed from all code blocks. Instance data blocks are generated together with function blocks (FBs), whereas global data blocks must be programmed by the user.

Ethernet
The Ethernet is a baseband bus system originally developed in order to connect mini-computers. It is based on the CSMA/CD access method. Coaxial cables or twisted-pair cables are used as the transmission medium.
The transmission speeds are 10 Mbit/s (Ethernet), 100 Mbit/s (Fast Ethernet) and 1 Gbit/s or 10 Gbit/s (Gigabit-Ethernet).
This widely used technology for computer networking in a LAN has been standardised since 1985 (IEEE 802.3 and ISO 8802-3).
Ethernet technology is now common practice in the office environment.
After making even very tough real-time requirements possible and adapting the device technology (bus cables, patch fields, junction boxes) to the harsh application conditions of the industrial environment, Ethernet is now also increasingly used in the field areas of automation technology.

FIFO
= First in, First out

Data that is written first into the FIFO memory buffer gets first out of the buffer.

Function (FC)
In the Siemens STEP 7 system, there is the code block type “functions”, which contains functions of structured user programs. This code block type is parameterisable and can store local data temporarily.

Function block (FB)
In the Siemens STEP 7 system, function blocks of structured user programs are part of the code blocks and contain task-specific subroutines. A function block always includes an instance data block, i.e. a particular memory area for special data. FBs are parameterisable and can store local data both statically and temporarily.

IP address
The IP address is a unique string of numbers separated by full stops that identifies each computer attached to the Internet. Usually, it also has a version containing words separated by full stops.

LSB
= Least Significant Bit

LSB is the lowest order bit in a digital quantity.

Modbus TCP
The Modbus TCP protocol is an open query/response protocol for the communication between master and slave or client and server. A PLC, for example, can act as a master that initialises the communication processes.
The data is transferred in the form of TCP/IP packets.

MSB
= Most Significant Bit

MSB is the highest order bit in a digital quantity.

Organisation block (OB)
In the Siemens STEP 7 system, organisation blocks belong to the code blocks and are the interface between the operating system and the user program. OBs can be called up by the operating system in certain situations when OB1, which is otherwise endlessly executed, is interrupted.

PLC
= Programmable Logic Controller

The PLC is a computer-based control unit whose functionality is defined by an application program. With standardised technical languages, this application program is relatively easy to produce. Because of its serial mode of operation, reaction times of PLCs are slower than those of VPS. As a family of devices with graduated and matched components, PLCs can now cover all levels of an automation hierarchy.

Socket
A socket is a bidirectional software interface to interprocess (IPC) or network communication. Sockets are a standardised interface (API) between the network protocol implementation of the operating system and the actual application software.

STL
= Statement List

STL is the instruction list language for Siemens PLCs according to IEC 61131-3.

10.2 Index

11 Contact and support

Do you have any questions? Write or call us:

Address: ADDI-DATA GmbH
Airpark Business Center
Airport Boulevard B210
77836 Rheinmünster
Germany

Phone: +49 7229 1847-0
Fax: +49 7229 1847-222

E-mail: info@addi-data.com

Manual and software download from the Internet:

www.addi-data.com