PYROSOFT

IO System and Pyrometers

DIAS Infrared GmbH

Copyright © 1995-2021 by
DIAS Infrared GmbH
Pforzheimer Straße 21
D-01189 Dresden
www.dias-infrared.de
info@dias-infrared.de
Dokumenten-Nummer:
01.98-D-28-46/004
Revision: February 2021
English

Table of Contents

CHAPTER 1

General information

This manual provides information on how to integrate IO systems and pyrometers into **PYROSOFT**. Thus, information such as alarm states, trigger conditions or process values can be exchanged easily and individually.

Should you still have any open questions, noticed any errors in this manual or wish to pass on any tips and suggestions for improvement, please inform your supplier or contact us directly:

DIAS Infrared GmbH
Pforzheimer Straße 21
D-01189 DRESDEN
Germany

Phone: +49 351 896 740
Fax: +49 351 896 7499
Email: info@dias-infrared.de
www: http://www.dias-infrared.de

This way, you help us to provide you with the best possible software and documentation.

Terms of Guarantee

All information and instructions for the operation of this device and all safety instructions are given to the best of our knowledge.

DIAS accepts no liability for the examples and procedures described in this manual, for any damage that may result from this, or in the event that the contents of this document may be incomplete or incorrect. DIAS reserves the right to make changes to this document and to the products described therein without the obligation to notify any person of such changes.

DIAS assumes no liability for damage or loss resulting from the use or manufacturing defects of the equipment, including financial losses and consequential damages.

The Windows software has been tested to the best of knowledge in several Windows operating systems in several languages. However, it can not be ruled out that there is a configuration of PC and Windows operating system or other circumstances in which it does not work properly.

No liability or warranty claims can be derived from the use of the PC software. Any liability for direct, indirect, consequential or consequential damages that may arise from the use of this program is excluded.

No guarantee is given for the completeness or correctness of the content of this documentation.

Overview of compatible IO systems

PYROSOFT supports the following IO systems:

- Network based IO system **WAGO®**
- **PROFIBUS®** with PCI card
- **PROFINET®** with PCI card
- **TCP-Socket** (Client)
- **MODBUS**

- Text file based IO system **Text-IO**

The maximum number of channels are:

	WAGO	PROFIBUS	PROFINET	MODBUS	Socket-IO	Text-IO
DI	128	512	512	512	512	512
AI	64	88	128	128	128	128
DO	128	512	512	512	512	512
AO	200	88	128	128	8192	128
AO as block	-	-	-	-	8	-
Required hardware	-	PCI card	PCI card	-	-	-

KAPITEL 2

Using an IO System or a Pyrometer in PYROSOFT

In diesem Kapitel

Using an IO System in PYROSOFT

The necessary settings for **PYROSOFT** can be configured and tested using the software tool **IOConfig.exe** (siehe Seite 7). Beforehand, all open programs that are still connected to the camera must be closed.

In **PYROSOFT,** the channels of the IO system can be used in the following ways:

- Digital outputs

Echo	For output of an echo signal of a digital input (is reflected back to the IO system, for testing the connection)
VOI Alarm	Output of VOI alarms
Status System OK	All system functions are working error-free and all <Status Camera Line> (see below) are OK
Toggle	For output of a toggle signal (alternating 0/1 with approx. 1 Hz), if <Status System> is OK (1)
Status Disk Space OK	1 = The free hard disk space is at least 0.3 GB. It takes into account the disk space of the Windows system and the application data.
Status Pyrometer	1 = Pyrometers are active and working error-free
Status Camera Line OK	1 = The camera line functions is working error-free and the data acquisition is running
Camera Status OK	1 = No camera error occured
Camera Temperature OK	1 = Camera is in the allowed temperature range
Camera Temperature	Output of camera temperature*10 (according to the temperature unit selected in PYROSOFT)
Camera Connection	1 = Camera is online
Data Acquisition	1 = Data acquisition is running
Shutter State	Only for bolometer cameras: 0 = shutter is open 1 = shutter is closed
Data Saving	1 = Data recording is active and running error-free
Data Saving for 2D Line	1 = Data recording for 2D line image is active and

Image	running error-free
Data Saving for History	1 = Data recording for history is active and running error-free
Alarm Data Saving OK	1 = Alarm data recording is active and running error-free
Temperature Calculation OK	1 = Temperature calculation in PYROSOFT running error-free
Bitmap Export	1 = Bitmap export is active and running error-free
Calculation Running	1 = The pulse trigger and / or the calculation of the phase image is active
Image Counter	Output of image counter
Difference Image Counter	Output of difference image counter
Time Stamp	Time stamp of the last image in day milliseconds

- Digital outputs for product control (**PYROSOFT Automation** only)

Product ID	Output of the current product ID
Product ID valid	1 = Current product ID is valid (product has been loaded successfully)
Enabling	1 = Enabled (output of VOI values and VOI alarms is active)

- Analog outputs

VOI Value	Output of VOI values
Block 1 - 8	Only for Socket-IO via TCP! Output of up to 8 blocks for VOI line values. The 1st value contains the number of line values.

- Digital inputs

Echo	As input for an echo signal for a digital output (is reflected back to the IO system, for testing the connection)
Reset VOI	To reset all VOI value outputs to default values (LH edge)
Print	To print the current infrared image in full screen mode (LH edge)
VOI Alarm	For a VOI alarm input
Start/Stop Data Acquisition	To start (LH edge) or stop (HL edge) the data acquisition
Single Trigger Data Acquisition	To trigger a single image (level or edge, depending on the setting in PYROSOFT)
Sequence Trigger Data Acquisition	To trigger a line sequence using a line camera (level or edge, depending on the setting in PYROSOFT)
Trigger Reference Image	To trigger a new reference image (LH edge)
Trigger Difference Image	To trigger a new difference image (LH edge)
Trigger Pulse	To trigger a pulse (LH edge: Start, HL edge: Stop) for the calculation of a triggered phase image
Shutter	Only for bolometer cameras: To execute a shutter procedure in the camera (LH edge)

Disable Shutter	Only for bolometer cameras: For blocking the automatic shutter procedures in the camera, (LH edge: blocking, HL edge: enabling automatic shutter)
Single Data Saving	To record a single image (LH edge)
Channel for Single Data Saving	Channel number for single data storage in different destination folders
Bitmap Export	To execute a bitmap export (LH edge); the index for the file name is generated from the active bits (1).
Saving Name	To pass 8-bit characters (ASCII) to form filenames for online saving functions in PYROSOFT

- Digital inputs for product control (**PYROSOFT Automation** only)

Product ID	Product ID for loading the product template (will be applied at the next start of data acquisition)

- Analog inputs

VOI Value	As input for VOI values
Reference Temperature	As input for an external reference temperature

- Analoge Eingänge (nur **PYROSOFT Automation / Automation SC**)

UserData1 - UserData4	Inputs for displaying and storing up to 4 customer-specific values

Details on the above-mentioned **PYROSOFT** functions can be found in the **PYROSOFT** software documentation.

Using pyrometers in PYROSOFT

PYROSOFT IO supports the connection to DIAS pyrometers. Their measurement data can be integrated into the online data analysis as a reference temperature value

The necessary settings can be configured and tested using the software tool **IOConfig.exe** (siehe Seite 7). Beforehand, all measurement documents must be closed.

IOConfig.exe

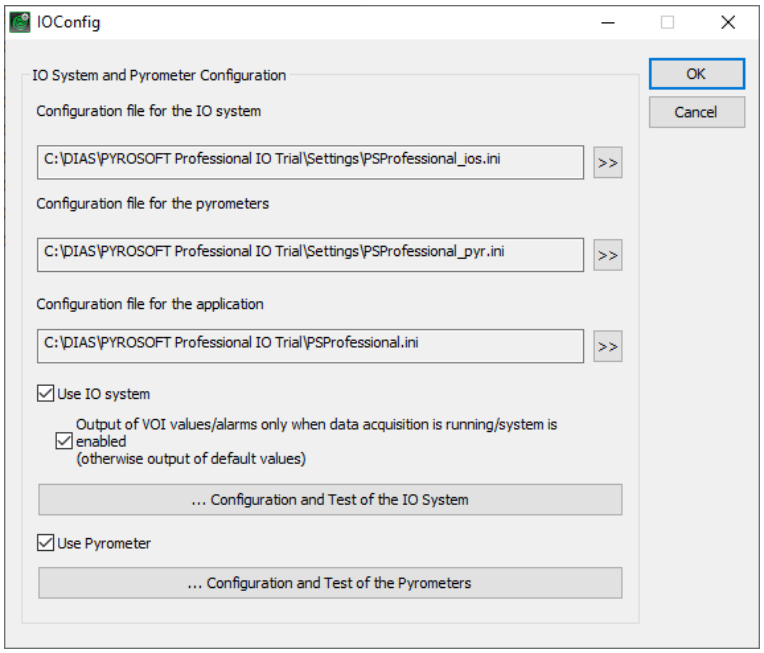
IOConfig.exe is an add-on program, which allows configuration and testing of a connected IO system or pyrometer.

How to start **IOConfig.exe:**

- In program directory **"DIAS\PYROSOFT...\Tools"**

In order to use **IOConfig.exe** it is required to close **PYROSOFT** beforehand.

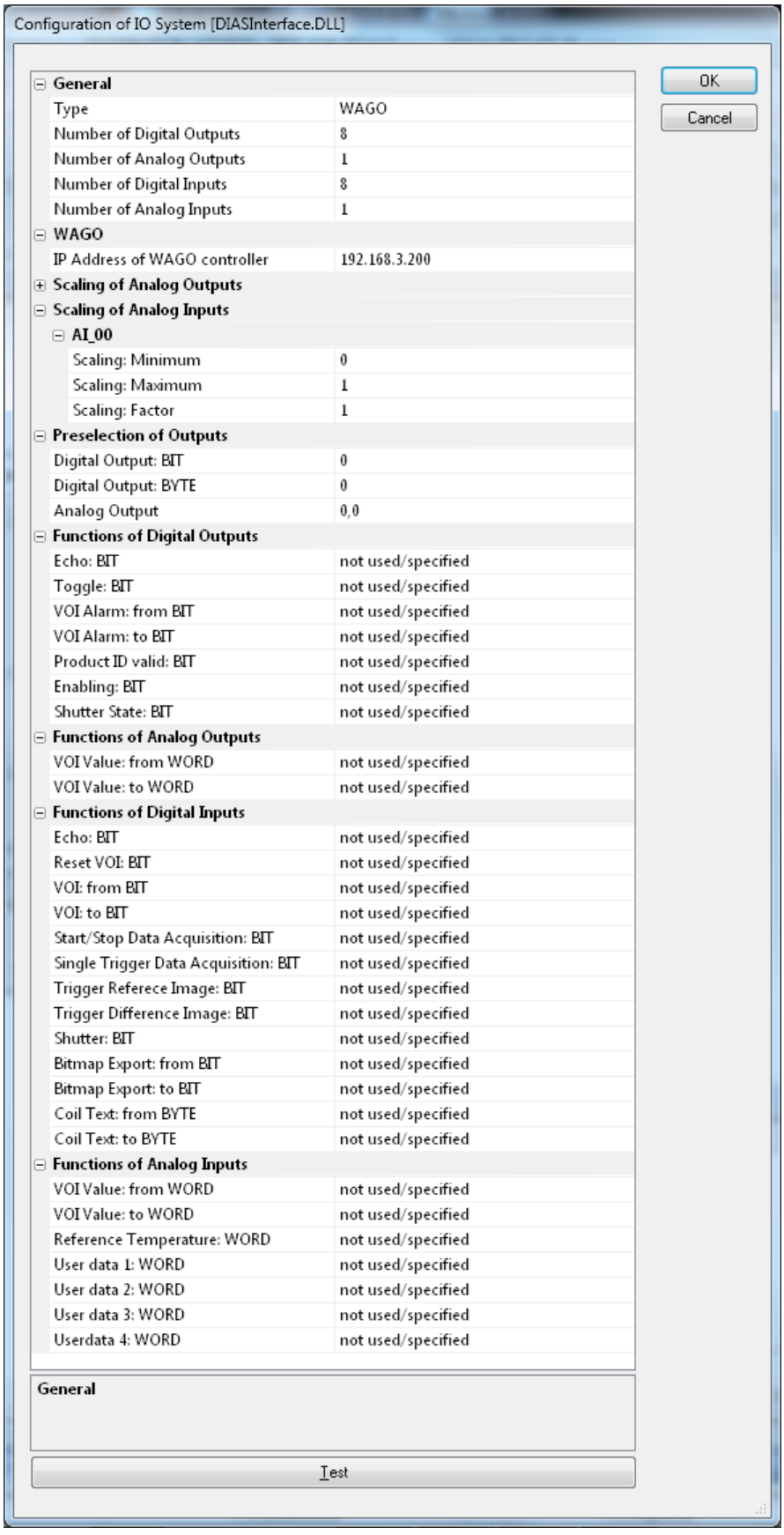
After starting **IOConfig.exe** the following dialog is displayed:



If **IOConfig.exe** is started from the **PYROSOFT** program directory (standard), the configuration file specifications are identified automatically. If your directory structure deviates from standard, please specifiy the file path of "PS*_ios.ini", PS*_pyr.ini and "PS*.ini" here.

IO System: PYROSOFT Configuration and Test

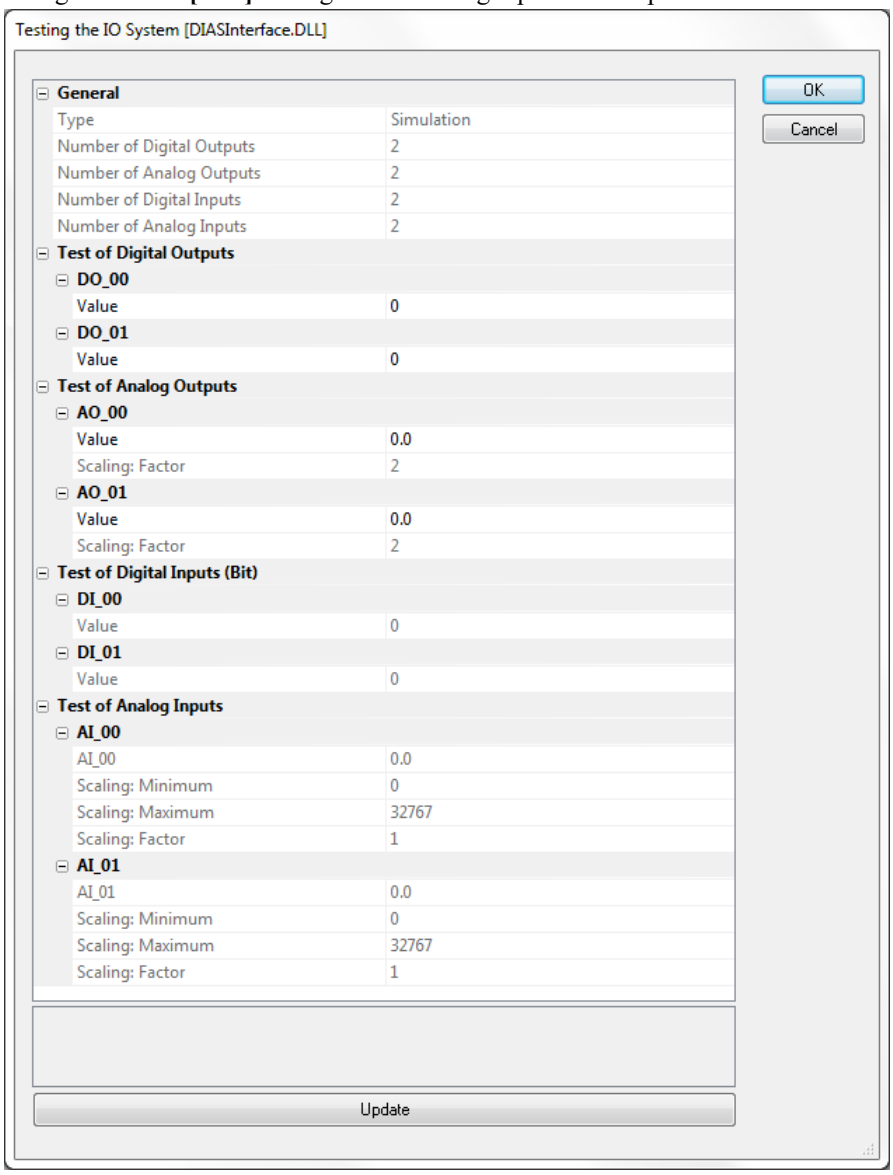
Activate "Use IO System" in **IOConfig.exe** and click the button **[Configuration and Test of the IO System]** in order to configure the use of the IO system. The following dialog is displayed:



Here the following parameters can be defined:

- System Type:
  Simulation, WAGO®, PROFIBUS®, PROFINET®, TCP socket, Text IO via file, MODBUS
- Number of digital and analog inputs and outputs
- Number of camera and currently selected camera (for software operation with multiple cameras only)
- various configuration parameters depending on the selected type of IO system
- Scaling of analog outputs
- Scaling of analog inputs
- Default values of outputs
- Functions of digital and analog outputs (see **IO System in PYROSOFT** on page **5**)
- Functions of digital and analog inputs (see **IO System in PYROSOFT** on page **5**)

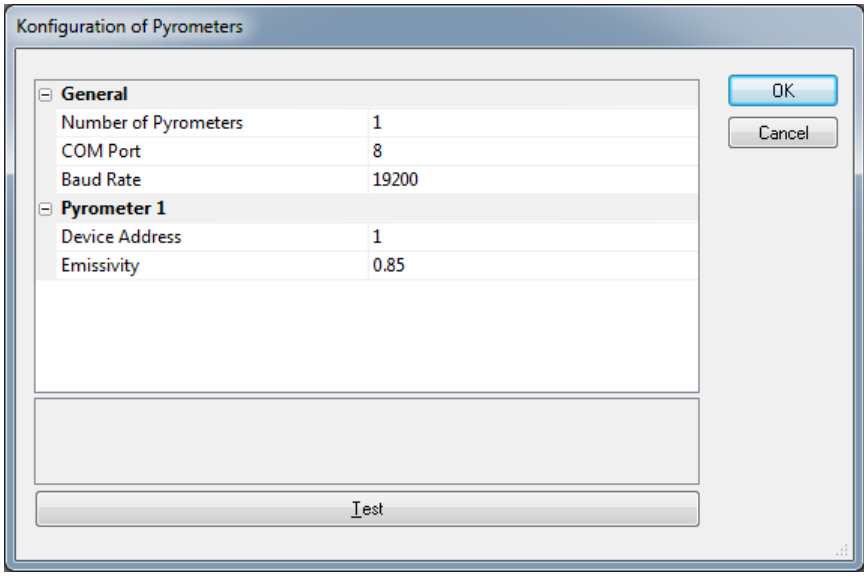
Using the button **[Test]** the digital and analog inputs and outputs can be tested:



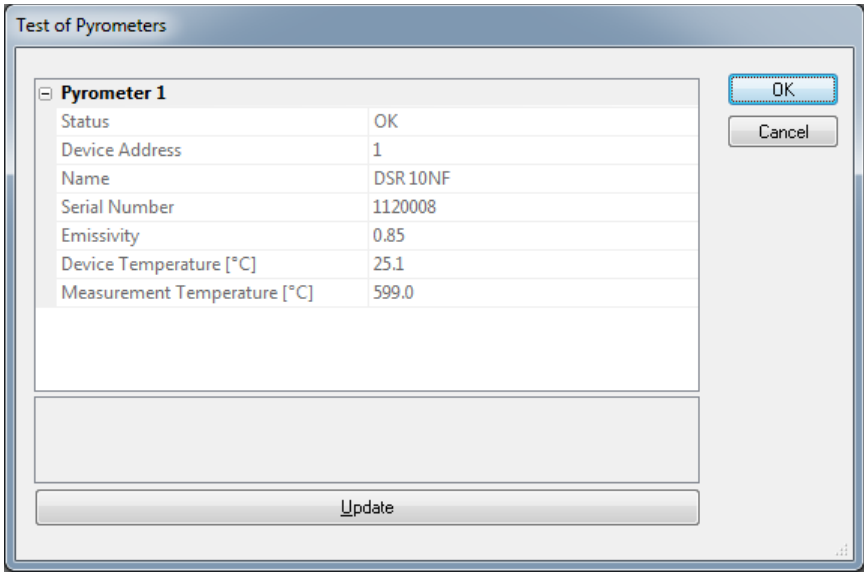
You can output test values to the IO system and display the currently applied input values. If the data in the IO system and in **IOConfig.exe** match, the data exchange is configured correctly. If not, you should recheck the configuration on both sides and adjust if necessary.

Pyrometer: Configuration and Test

Activate "Use Pyrometer" in **IOConfig.exe** and click the button **[Configuration and Test of the Pyrometers]** in order to configure the use of pyrometers. The following dialog is displayed:



The settings can be verified using the button **[Test]**. It opens a test connection and downloads and displays the pyrometer data:



CHAPTER 3

Profibus

In this Chapter

Integration

The PC is connected to the Profibus network via the PCI Profibus card from Hilscher, which operates as a slave.

The PLC must be configured as master. In order for the PLC to communicate with the PCI card, the GSD file belonging to the Profibus PCI card must first be imported. You can then configure the card's inputs and outputs.

To set up the PCI card on the PC, it is necessary to install the appropriate drivers and configuration tools from the enclosed CD (SYCON.net and CIFX drivers).

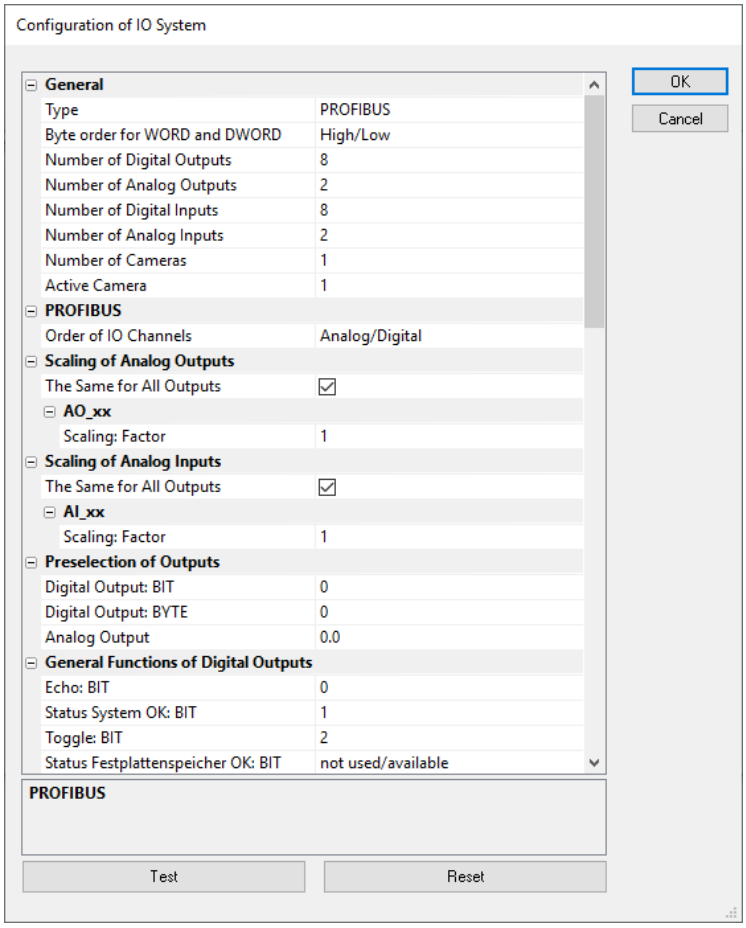
If a signal exchange list is available, a configuration file (SPJ file) for the PCI card is created by DIAS. This can be loaded in SYCON.net via **[File> Open]**. In the configuration dialog (right-click on the icon in the project view), the corresponding firmware must now be loaded (NXF file) and the card declared as a Profibus slave ("Device Assignment"). Close the configuration dialog and select **[Download]** to transfer the settings.

Settings

Configuration of PYROSOFT

Start **IOConfig.exe** (see **IOConfig.exe** on page 7), activate **[Use IO System]** and click on the **[Configuration and Test of the IO System**] button.

The following dialog appears:



In **[General > Type]** choose the option "PROFIBUS". Specify the required number of inputs and outputs.

If required, you can also change the order for the IO channels and the byte order.

Finally, define the functions of the individual inputs and outputs. For an overview of the available functions for inputs and outputs see **IO System in PYROSOFT** on page **5**.

Data Exchange

For the correct data exchange it is important that the settings in the PLC, the Profibus PCI card and in **PYROSOFT** match. An example table is shown below:

	Profibus PCI Card (Slave)	**PYROSOFT**	**PLC (Master)**
DI	8	8	64
AI	0	0	8
DO	64	64	8
AO	8	8	0

CHAPTER 4

Profinet

In this Chapter

Integration

The PC is connected to the Profinet network via the PCI Profibus card from Hilscher, which operates as a slave.

The PLC must be configured as master. In order for the PLC to communicate with the PCI card, the GSD file belonging to the Profinet PCI card must first be imported. You can then configure the card's inputs and outputs.

To set up the PCI card on the PC, it is necessary to install the appropriate drivers and configuration tools from the enclosed CD

Afterwards, in the netX configuration tool the card has to be configured appropriately and the firmware has to be installed.

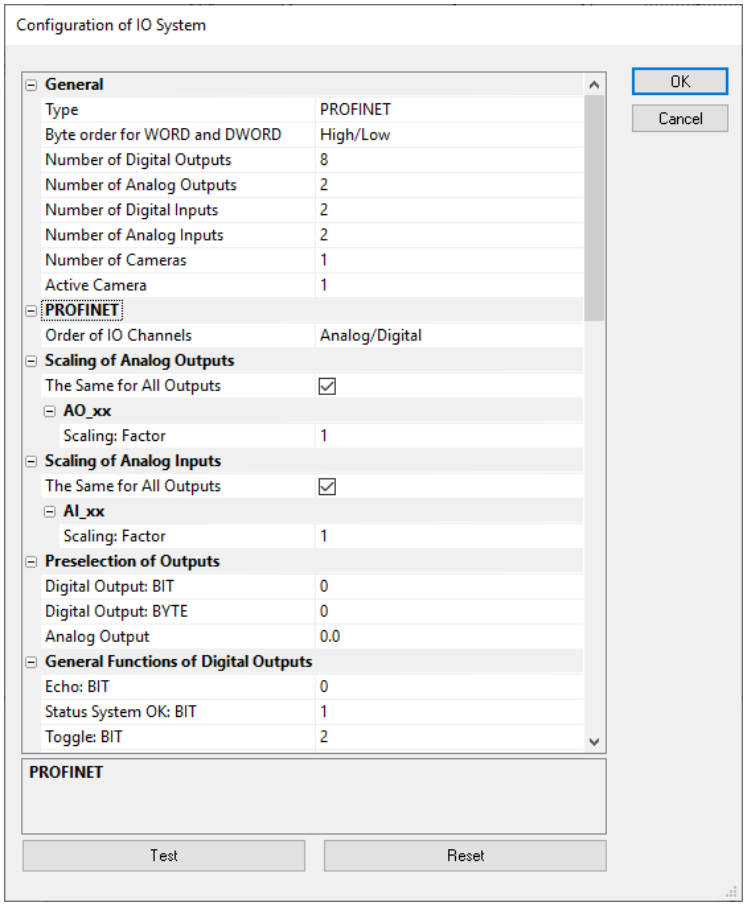
If a signal exchange list is available, a configuration file (SPJ file) for the PCI card is created by DIAS. This can be loaded in SYCON.net via **[File> Open]**. Select **[Download]** to transfer the settings (right-click on the icon in the project view).

Settings

Configuration of PYROSOFT

Start **IOConfig.exe** (see **IOConfig.exe** on page 7), activate **[Use IO System]** and click on the **[Configuration and Test of the IO System**] button.

The following dialog appears:



In **[General > Type]** choose the option "PROFINET". Specify the required number of inputs and outputs.

If required, you can also change the order for the IO channels and the byte order.

Finally, define the functions of the individual inputs and outputs. For an overview of the available functions for inputs and outputs see **IO System in PYROSOFT** on page **5**.

Data Exchange

For the correct data exchange it is important that the settings in the PLC, the Profinet PCI card and in **PYROSOFT** match. Therefore, an example table is shown below:

	Profinet PCI Card (Slave)	**PYROSOFT**	**PLC (Master)**
DI	8	8	64
AI	0	0	8
DO	64	64	8
AO	8	8	0

CHAPTER 5

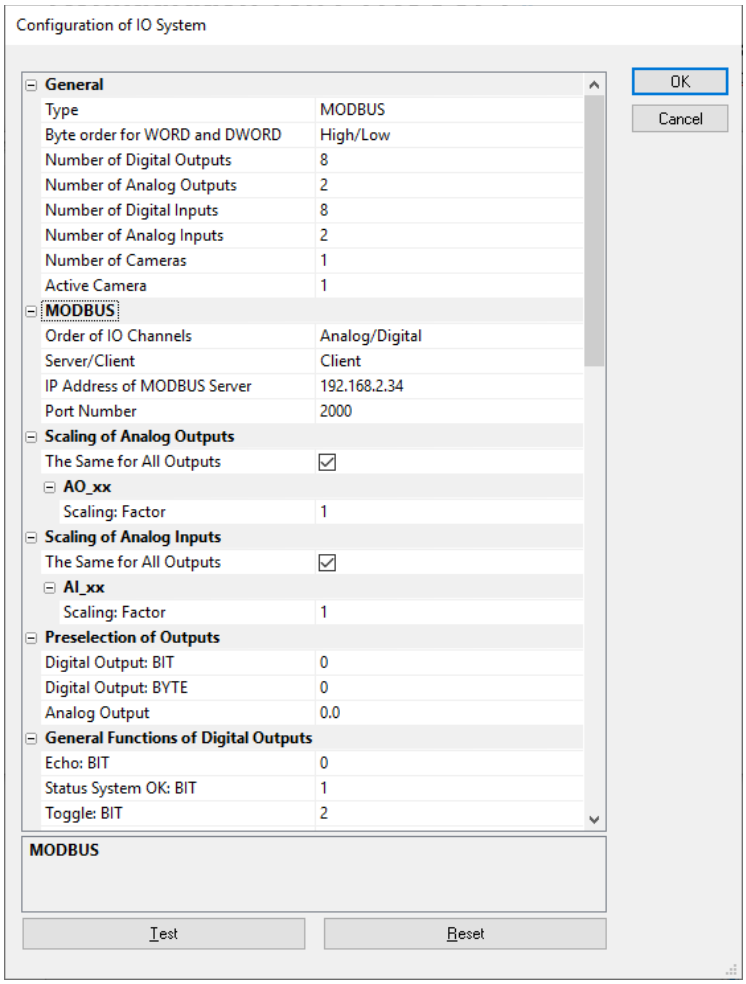
Modbus

In this Chapter

Settings

Configuration of PYROSOFT

Start **IOConfig.exe** (see IOConfig.exe on page 7), activate **[Use IO System]** and click on the **[Configuration and Test of the IO System**] button.
The following dialog appears:



In **[General > Type]** choose the option "MODBUS". Specify the required number of inputs and outputs.

In category [**MODBUS**], choose whether **PYROSOFT** should work as server or client. For clients, you must specify the IP address and associated port number of the Modbus server. For servers, specify the port that **PYROSOFT** should open for transmission.

If required, you can also change the order for the IO channels and the byte order.

Finally, define the functions of the individual inputs and outputs. For an overview of the available functions for inputs and outputs see **IO System in PYROSOFT** on page **5**.

Data structure

An input is set by the server and output is read by the server. The following mapping applies to reading and writing the data:

PYROSOFT		**MODBUS Access**	
Inputs		**Read**	**Write**
DI		Bits FC01 HoldingRegs FC03 (grouped)	Bit FC05 HoldingRegs FC06 (grouped)
AI		HoldingRegs FC03	HoldingRegs FC06
Outputs			
DO		InputBits FC02 InputRegs FC04 (grouped)	
AO		InputRegs FC04	

All registers are counted from address 0.

Bits are counted from address 0 and, depending on the setting, mirrored in front of or behind the analog registers (16 bits in one register).

CHAPTER 6

TCP Socket

In this Chapter

Integration

The data is transmitted via TCP. The PLC works as server, **PYROSOFT** as client.

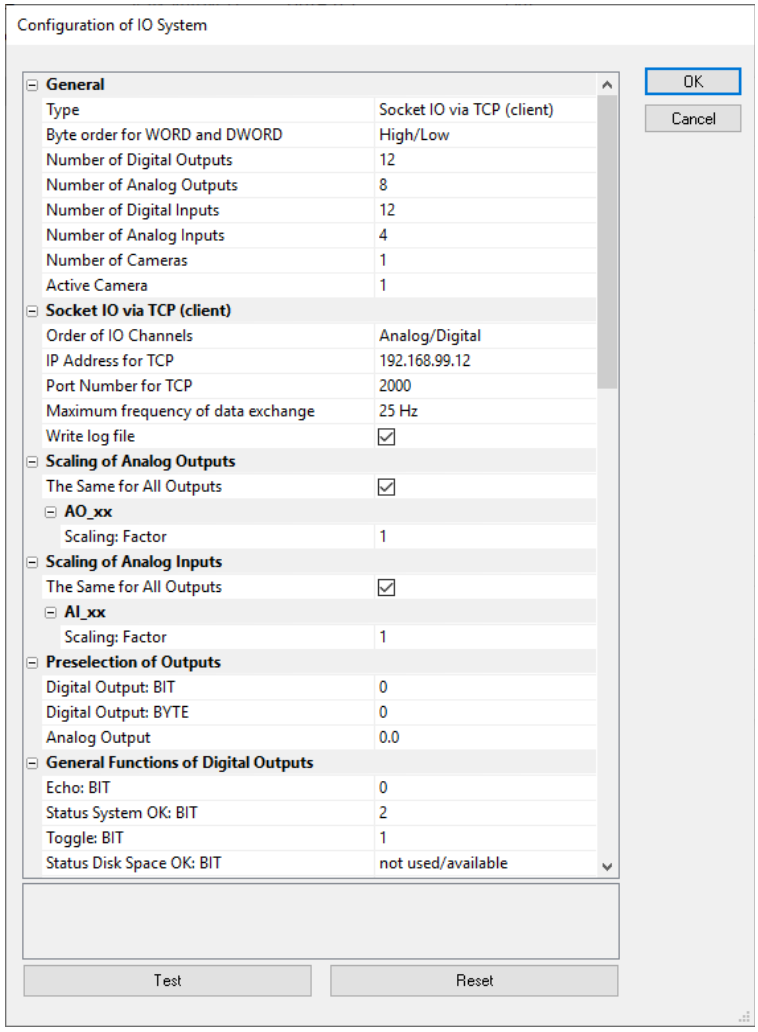
In parallel, it is possible for **PYROSOFT** to simultaneously transmit analog data over an UDP channel. See chapter **Configuration of PYROSOFT** (on page **19**) for more information.

Settings

Configuration of PYROSOFT

Start **IOConfig.exe** (see **IOConfig.exe** on page **7**), activate **[Use IO System]** and click on the **[Configuration and Test of the IO System]** button.

The following dialog appears:



In **[General > Type]** choose the option "Socket IO via TCP". Specify the required number of inputs and outputs.

In category [**Socket IO via TCP]**, specify the IP address of the server, the associated port and the frequency of the data exchange (1Hz to max.100Hz).

If required, you can also change the order for the IO channels and the byte order.

Finally, define the functions of the individual inputs and outputs. For an overview of the available functions for inputs and outputs see **IO System in PYROSOFT** on page **5**.

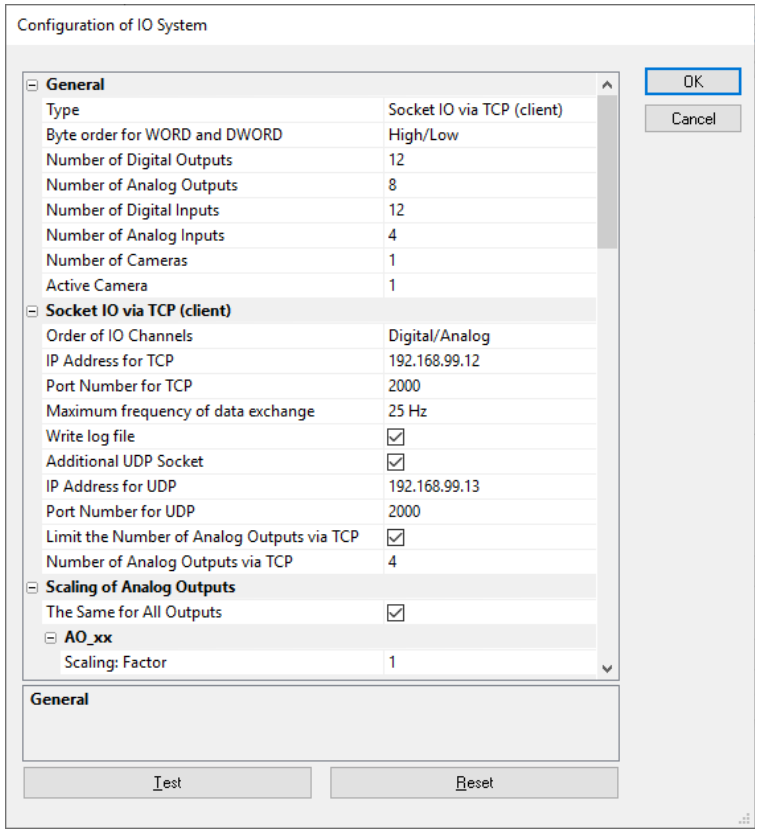
Additionally, it is possible for **PYROSOFT** to simultaneously transmit analog data via an UDP channel. The analog data are then output both via TCP and via UDP.

To do that, proceed as follows:

In **[Socket IO via TCP > Order of IO Channels]** select the option "Digital/Analog". Then check "Additional UDP Socket" and enter the IP address and port number for the destination of the UDP connection.

If fewer analog channels are required for TCP than for UDP, select the option "Limit number of analog outputs via TCP". Then only the first portion (according to your specification) of the analog data is sent via TCP, whereas all configured analog data is transmitted via UDP.

Example of a configuration with additional UDP socket:



Data Structure

An input is set by the server and output is read by the server.

A data packet to the server contains the information of the digital and analog outputs; a data packet from the server contains the digital and analog inputs. The length of an incoming data packet at the server will be calculated as follows:

data length in Bytes = (amount of digital outputs – 1) / 8 + 1 + amount of analog outputs *2.

For example, an IO configuration with 12 digital outputs and 6 analog outputs the data length of the packet will be 14 bytes.

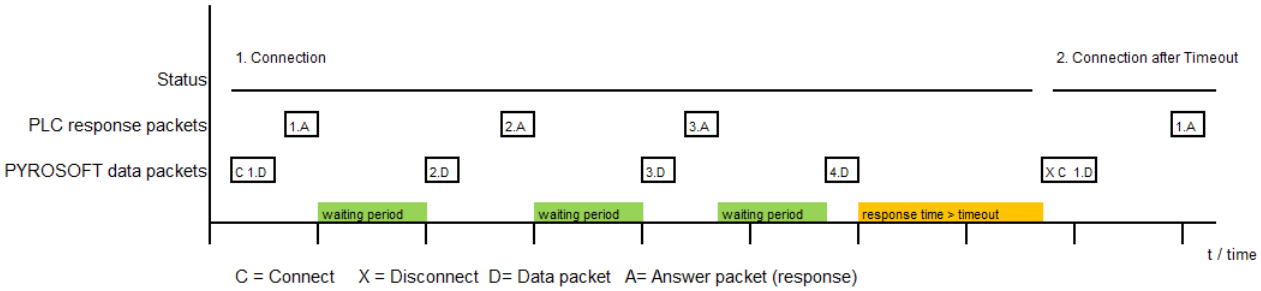
The same calculation is valid for the outgoing data packet of the server.

Timing

PYROSOFT (as client) establishes the connection to the server PLC and sends the defined data package (incoming data at the server). Afterwards it waits for the answer of the server PLC (outgoing data of the server). Only when the answer has been received the next packet is sent after the agreed time interval. If there is no answer by server in a time interval of 30 seconds the communication will be stopped and restarted by **PYROSOFT**.

The following figure illustrates this process.

Flowchart



Example

Required Function Blocks (Server, PLC)

For example, the following blocks are required for a Beckhoff PLC (TwinCAT):

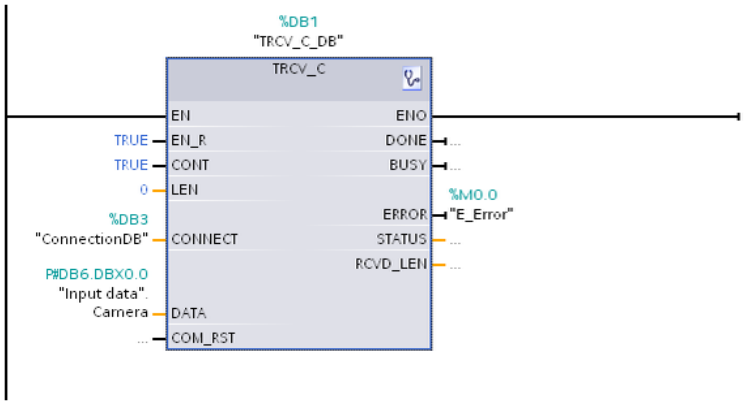
The FB_ServerClientConnection function block manages (establishes and dismantles) a server connection. FB_ServerClientConnection simplifies the implementation of a server application by internally encapsulating the functionality of the three function blocks FB_SocketListen, FB_SocketAccept and FB_SocketClose internally. In addition, a minimal server application requires one instance of the FB_SocketSend and one instance of the FB_SocketReceive function block.

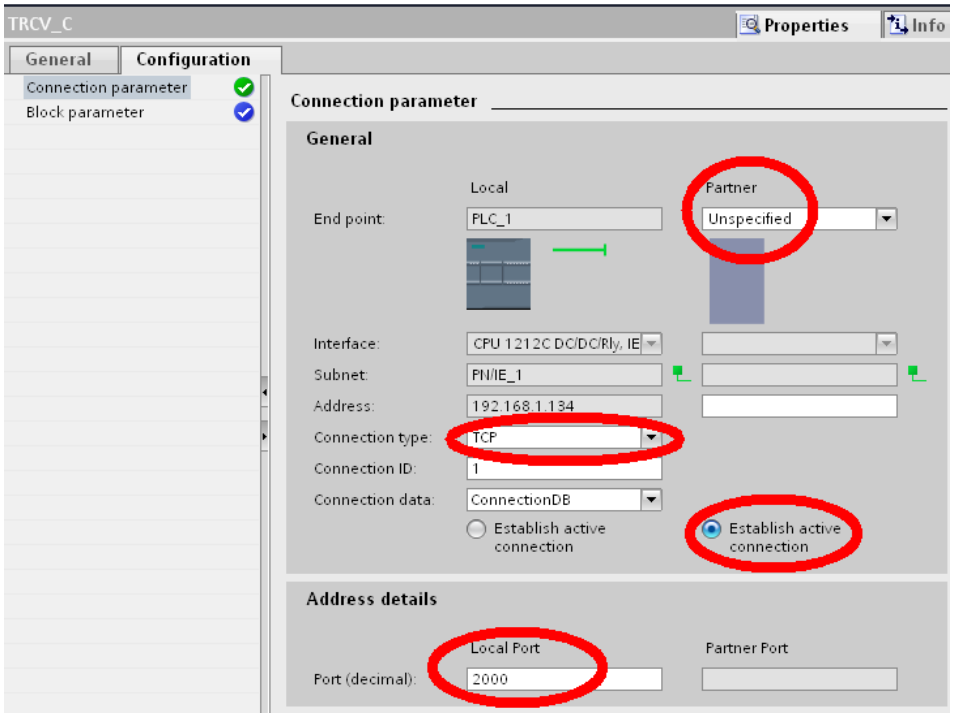
A typical server application starts with the function block ServerClientConnection by accepting an incoming communication request by the client. The function block FB_SocketReceive receives the process data sent by the client. After every dataset the server has to execute the function block FB_SocketSend to send a response to the client PC (e.g. start / stop information).

The PLC (the server) has to answer all incoming data packages with its own outgoing data. The packet length for sending and receiving is static!

Thus, the timing is controlled by the client (camera) and it is ensured that each image from the camera can be evaluated by the server.

Sample program for S7-1200

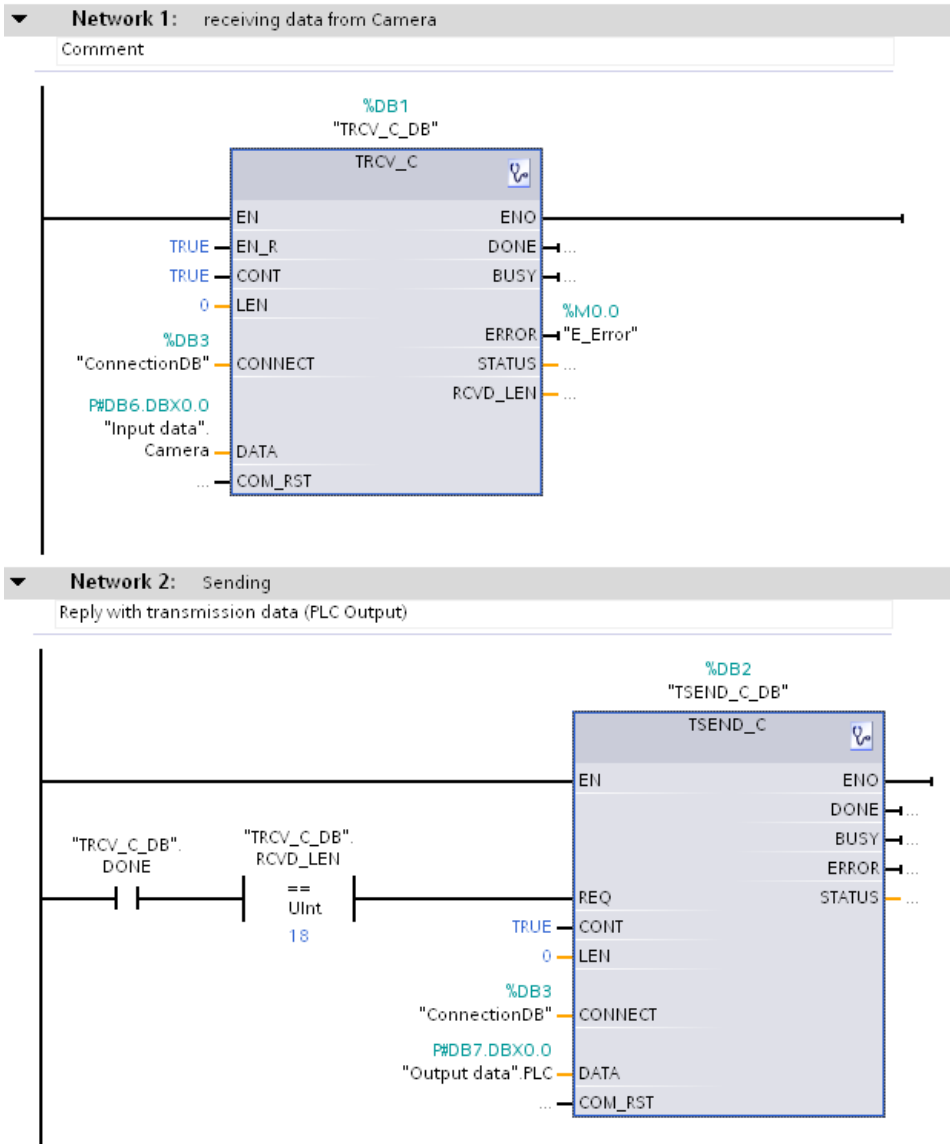




The PLC works as a server and the connection will be initiated by **PYROSOFT** PC.

Network 1: The receive block TRCV_C waits for a TCP packet of 18 bytes and stores all in DB6 ("Input data Camera").

Network 2: Every received packet is answered with transmission data DB7 ("Output data PLC"), but only if the received data length was 18 bytes.


Network 1: receiving data from Camera
Comment
%DB1
"TRCV_C_DB"
TRCV_C
EN
ENO
TRUE
EN_R
DONE
TRUE
CONT
BUSY
0
LEN
%M0.0
"E_Error"
%DB3
"ConnectionDB"
CONNECT
ERROR
STATUS
RCVD_LEN
P#DB6.DBX0.0
"Input data".
Camera
DATA
COM_RST
Network 2: Sending
Reply with transmission data (PLC Output)
%DB2
"TSEND_C_DB"
TSEND_C
EN
ENO
DONE
BUSY
ERROR
"TRCV_C_DB".
DONE
"TRCV_C_DB".
RCVD_LEN
==
UInt
18
REQ
STATUS
TRUE
CONT
0
LEN
%DB3
"ConnectionDB"
CONNECT
P#DB7.DBX0.0
"Output data".PLC
DATA
COM_RST

CHAPTER 7

Text IO

In this Chapter

Functionality

Text IO is a file based IO system. Communication is carried out by reading and writing text files saved on the computers hard disk.

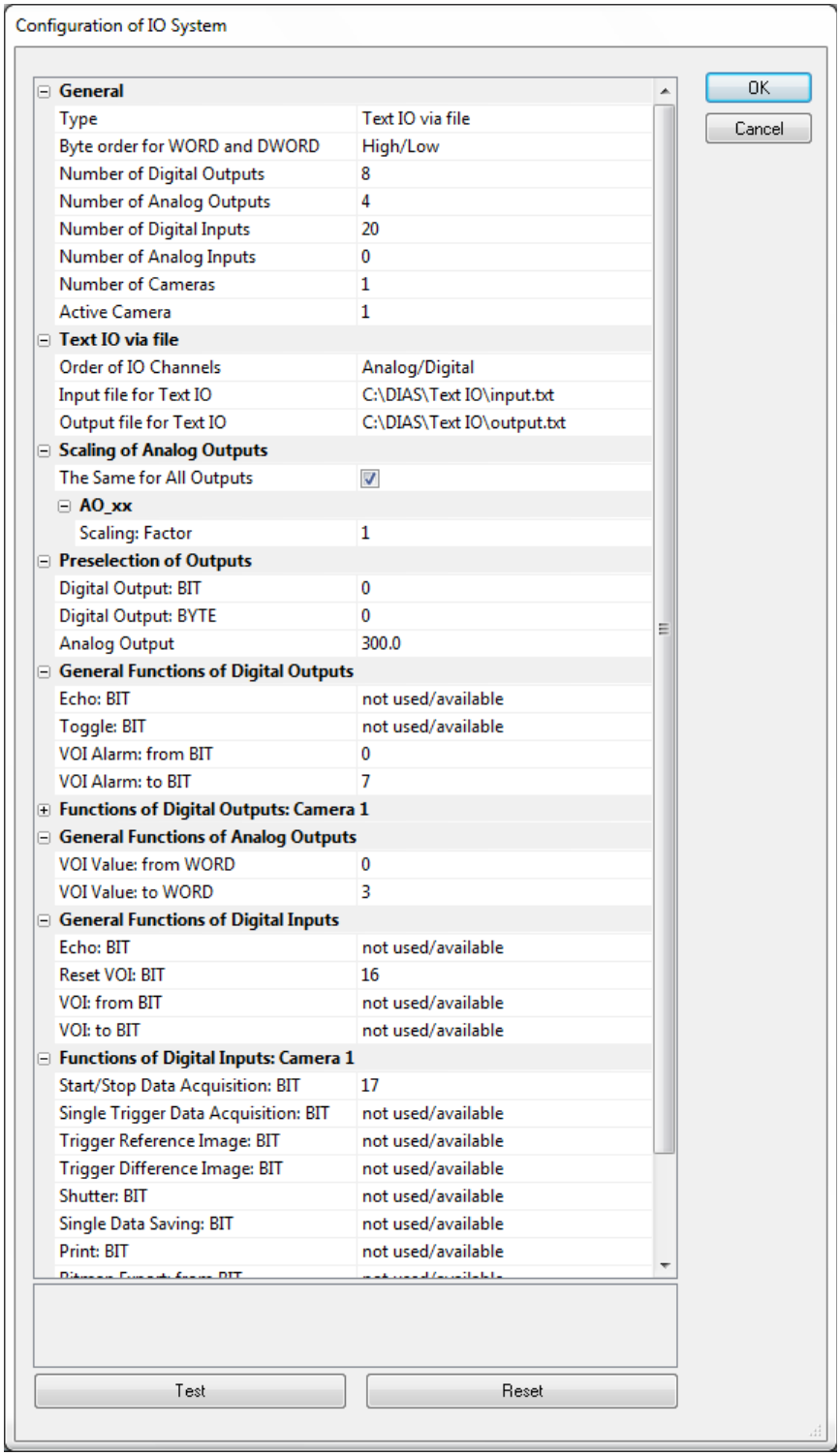
PYROSOFT automatically updates the status of the output channels by re-writing the corresponding text file. The input channels are set by changing and storing the associated text file.

Settings

Configuration of PYROSOFT

Start **IOConfig.exe** (see **IOConfig.exe** on page 7), activate **[Use IO System]** and click on the **[Configuration and Test of the IO System**] button.

The following dialog appears:



In **[General > Type]** choose the option "Text IO via file". Specify the required number of inputs and outputs.

The paths for the text files for input and output channels have to be specified in category **[Text IO via file]**.

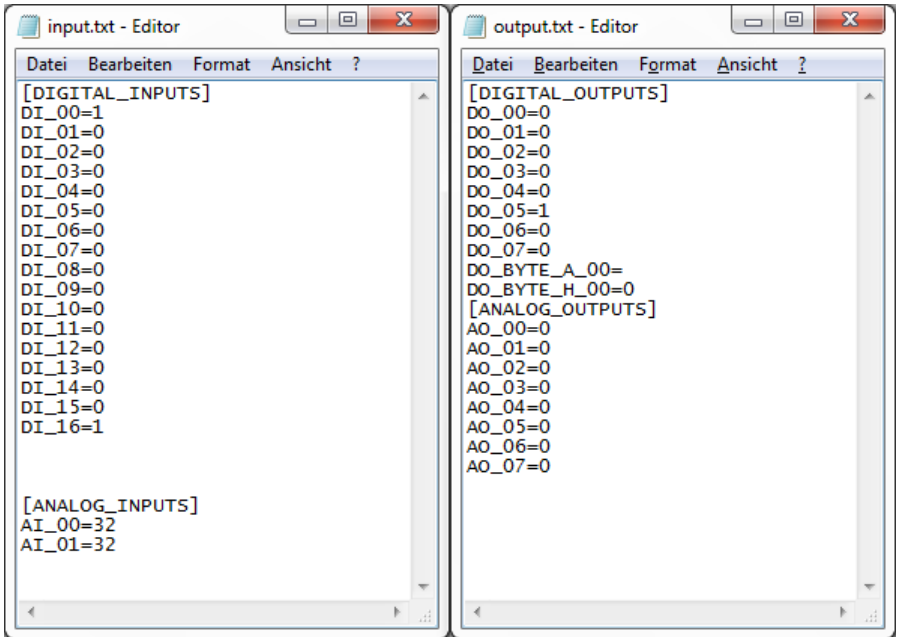
If required, you can also change the order for the IO channels and the byte order.

Finally, define the functions of the individual inputs and outputs. For an overview of the available functions for inputs and outputs see **IO System in PYROSOFT** on page **5**.

Data Exchange

PYROSOFT writes the required output values into the specified "Output file for Text IO" and reads the input values from the "Input file for Text IO".

Example:



The abbreviations stand for:

DI_xx:	Digital input (bit)
AI_xx:	Analog input
DO_xx:	Digital output (bit)
DO_BYTE_A_xx:	Digital output (byte, ASCII)
DO_BYTE_H_xx:	Digital output (byte, hexadecimal)
AO_xx:	Analog output