Getting started with Modbus

Contents

Project history

Date	Author	Modification
30.04.2009	TCS / sr	V1 Preparation of the documentation (Version 1) and project for PG5 1.4.300
23.06.2009	TCS	V2 Updated project and documentation for PG5 2.0
03.12.2009	TCS / sdu	V3 Improved the explanation of UID and Media Mapping

1 Introduction

This document offers a simple introduction for the use of Saia Modbus. With the associated PG5 project, it is meant as a guide for the implementation of Modbus communication.
The information contained in this document is an abstract of the corresponding manual and online help and will make it easy for you to get started. For more information, please consult the appropriate document (see section "References").

2 Required hard- and software

Hardware
This project has been configured for the following hardware configuration.

- PCD3.M5540 as client IP and/or RS
  Firmware 1.10.16 or higher
- PCD3.M3330 as server IP
  Firmware 1.10.16 or higher
- PCD2.M5540 as server RS
  Firmware 1.10.16 or higher
- Ethernet cable (CAT5) to connect PCD3 client (IP) and PCD3 server (IP)
- Cable to connect the RS interfaces RS 485
- A USB cable (max. 1.8m) for programming the PCD

Software
For programming the PCD, the following softwares with valid licences are required.

- PG5 2.0.90 or higher
  For programming the PCD. (HLK library was used for the time function, but is not necessary for the operation of the Modbus communication.)
- Modbus Saia library

This project can also be certainly operated with other hardwares. Specific adaptations of the configuration have to be undertaken according to the hardware for that purpose (hardware configuration in PG5, software settings in PG5, available memory in Fupla and corresponding settings for the communication between the PCDs).
The Modbus Saia library is supported only by the new systems like PCD3 and PCD2.M5xxx.

3 The Modbus basics

This section will give you a brief overview of Modbus – the typical applications, historical background, special features of Modbus and the comparison with S-Bus. The mapping of the PCD resources is briefly explained. Also, the differences between the Saia Modbus library and the existing Engiby library are pointed out.

3.1 Typical applications

3.1.1 Serial

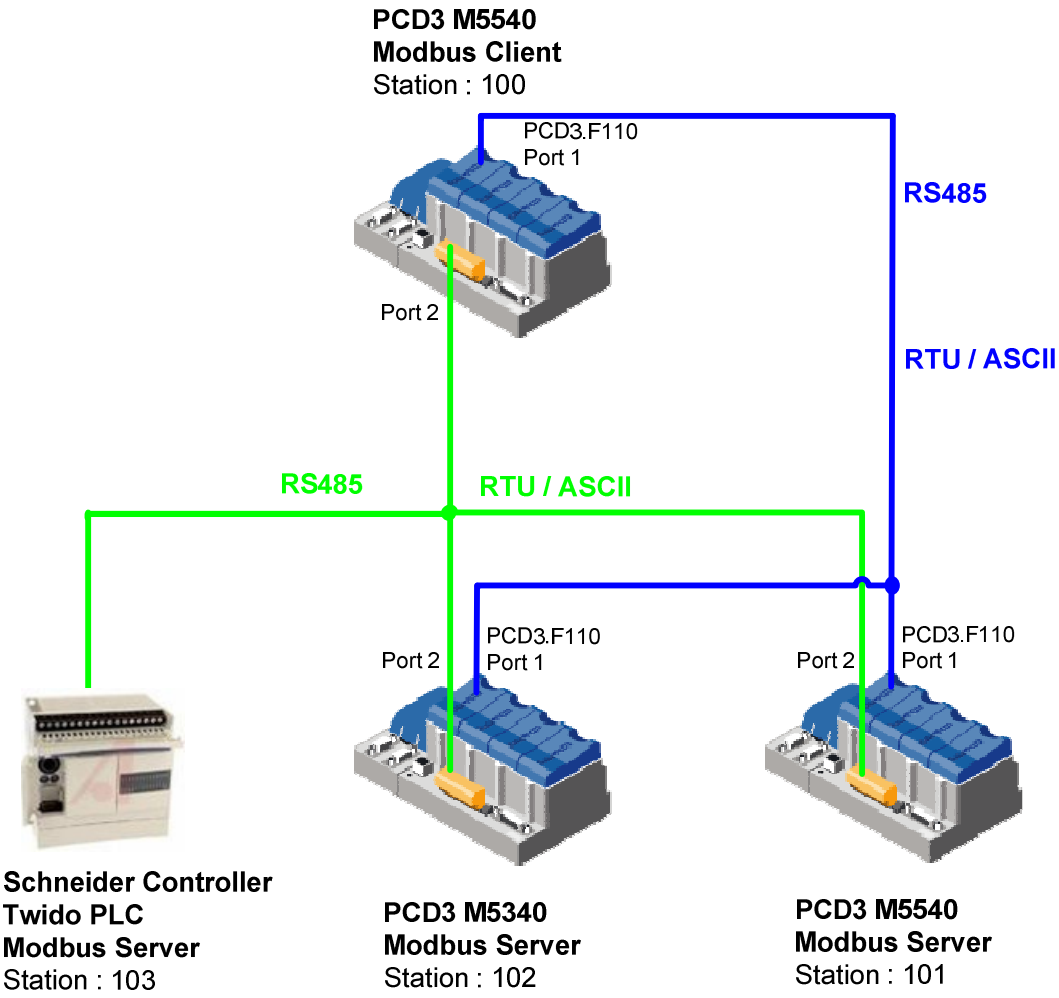


Fig. 3.1.1.1 Serial structure

3.1.2 Ethernet

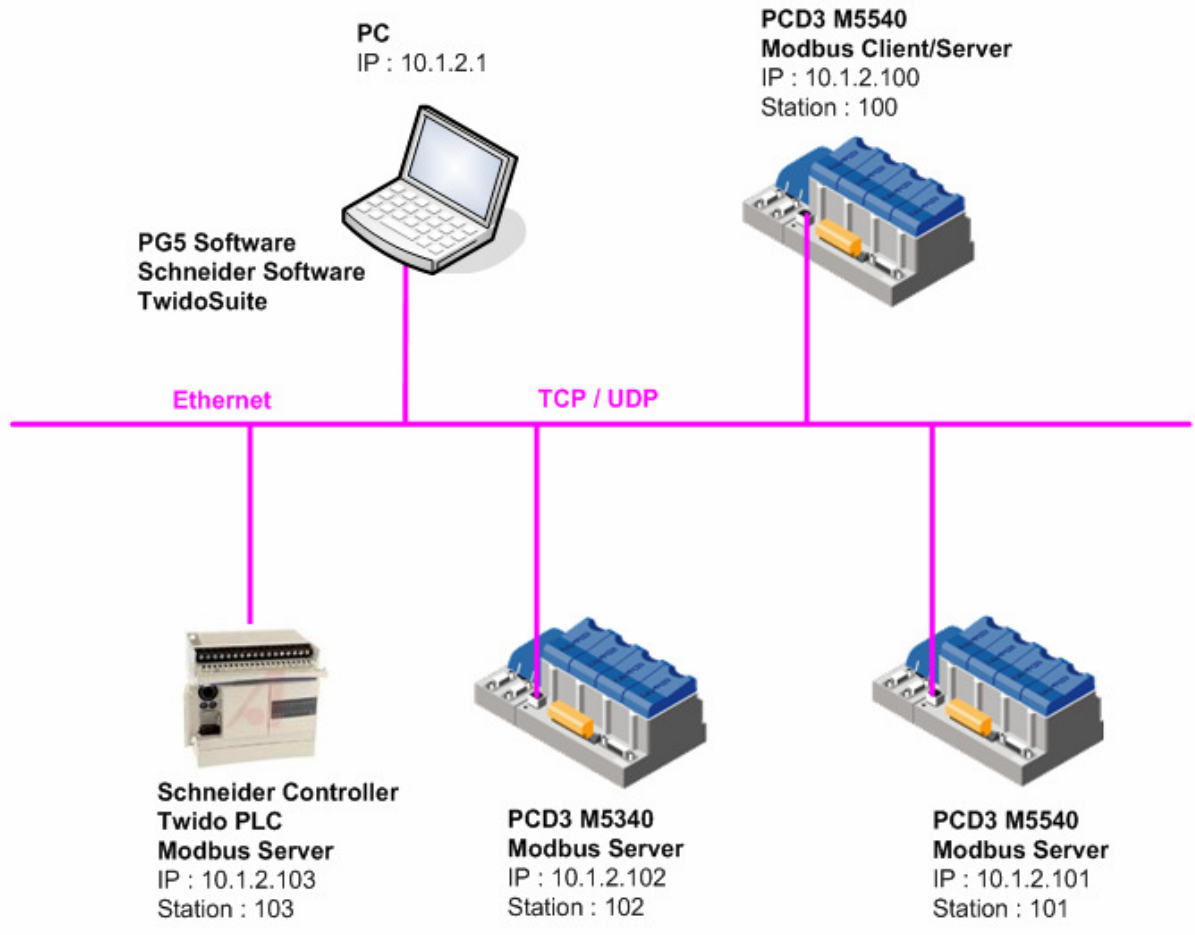


Fig. 3.1.2.1 Ethernet structure

3.2 Historical background

Modbus was introduced in 1979 by the company Modicon (today Schneider). It is a single master bus. In the course of time the Modbus protocol became a de facto standard supported by very many manufacturers.
Modbus communication protocols:

- Modbus RTU: Binary coded with breaks between telegrams. This type of communication is efficient, but sensitive to delays in character transmission.
- Modbus ASCII: Allows easy interpretation/reading by humans. The Start and Stop characters are used. The telegrams are approximately 2 times longer than with Modbus RTU.
- Modbus TCP/UDP: Operated as client/server network. Implementation of Modbus on Ethernet (TCP/IP and UDP/IP). This allows the implementation of a multi-master network topology.

3.3 Comparison S-Bus ⇔ Modbus

Even though serial S-Bus and Modbus RTU / ASCI are available on the same physical layers, there are some differences. The protocols are contrasted with each other in the following table.

- An important point is that an S-Bus server (slave) always has one address, whereas a Modbus server (slave) can have several UIDs.
- In the S-Bus, all media (registers, inputs, flags, etc.) have fixed addresses. However, in Modbus a specific mapping can be defined for each UID (Unique Identifier).
- Modbus holding registers are 16 bits long. PCD registers are 32 bits long.

3.3.1 Comparison Serial S-Bus ⇔ Modbus RTU/ASCII

Criterion	Serial S-Bus	Modbus RTU/ASCII
Physical interface	RS 485, RS 232, RS 422, current loop ...	Ditto, serial modem not supported by our implementation
Max. number of masters	1	Ditto
Max. number of slaves	252	247
Address per slave	1 S-Bus address	Up to 9 unique Ids (UID)*
Broadcast address	255	0
Checksum	CRC 16	Ditto
1 bit data	Input / output, flag	Discrete input (DI, read only) Coil (readable and writable)
16/32 bits data	Register (32 bits) DB (32 bits)	Input register (IR, 16 bits read only) Holding register (HR, 16 bits, readable and writable)
First data address	0	1
Supported baud rates	300-115000	1200-115000**
Number of ports one can operate with the protocol	As many as the serial interfaces it has.	Maximum 10
Can the protocols be mixed on a PCD?	Yes	Ditto
Maximum telegram length (data)	128 bytes	253 bytes
PCD3.F2xx module supported	Yes	Not yet
Influence on the performance of the control.	The performance decreases with every configured port according to baud rate.	Ditto

*In our implementation, plus default UID 0

** Given by the supported CPU's

3.3.2 Comparison Ether-S-Bus ⇔ Modbus TCP/UDP

Criterion	Ether S-Bus	Modbus TCP/UDP
Physical interface	Ethernet	Ditto
Connection	Only UDP	TCP or UDP
Standard port	5050	502
Max. number of servers per network	Many	Ditto
Max. number of clients per network	Many	Ditto
Max. number of client connections per network	No restriction, as they are sequentially processed	10
Max. number of server connections per network	No restriction, as they are sequentially processed	26 minus number of client connections
Addresses per slave / server	1 S-Bus address	Up to 9 unique Ids (UID)*

*In our implementation, plus default UID 0

Important difference: In S-Bus, a communication channel is always assigned to a physical port. A Modbus channel consists of port and protocol. That means several channels with different protocols can be defined in one physical Ethernet port.

E.g. TCP on 502, UDP on 502, TCP on 503 and ASCII on the serial port 2.

3.4 Modbus Media Mapping and characteristics

3.4.1 Master/Slave ⇔ Client/Server

The serial Modbus is based on master/slave. However, Modbus TCP/UDP is based on client/server. To simplify, the designation client/server is used in all FBoxes. In which, Client=Master and Server=Slave correspond.

3.4.2 16 bits ⇔ 32 bits

Modbus is based on 16 bits (word). The transmission of 32-bit values is normal, but not standardised. Neither for integer nor for floating-point values. So, one also finds several ways for the mapping of 32-bit values in holding register and vice versa.

In order to cover all these cases, we support all required mapping methods and processes like "word swapping" and conversion. If 32-bit data are replaced with an external product, the following mapping settings should be specified:

- Signed / unsigned?
- Lower word first or last?
- In floating point: IEEE or FFP

3.4.3 Offset

Modbus resources are counted from 1 upwards, whereas PCD register begins with 0. This offset can lead to problems, as there is a chance to read or write the wrong value because of it. Therefore a checking of the offset settings is required on both sides. The FBoxes permit the configuration of an offset.

3.4.4 What is a UID

A UID (Unique Identifier) is like an S-Bus address, otherwise each station can have up to 9 UIDs. In a serial network, each UID can exist only once. Each UID has his particular mapping. The UID can be accessed over every Modbus communication port (TCP, UDP, Serial).

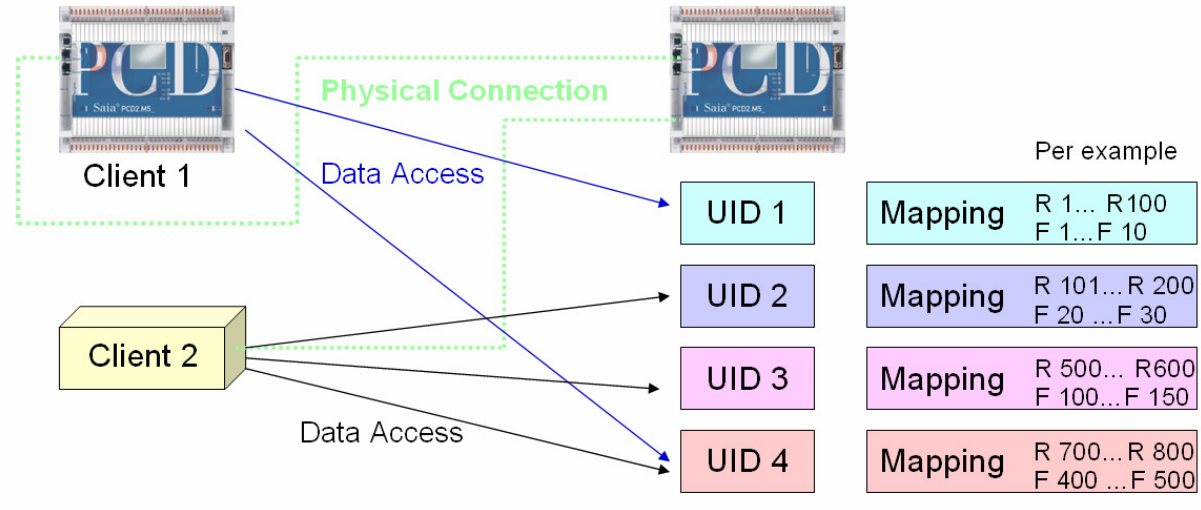


Fig. 3.4.4.1 UID

3.4.5 Why do we need a Mapping?

With the Mapping the UID allows access to certain resources. As server the following FBoxes are available for the Mapping:

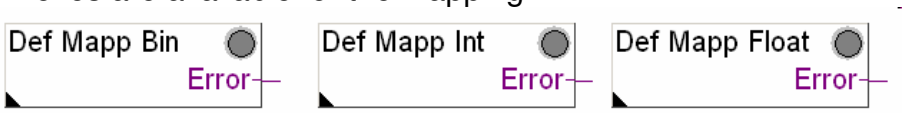


Fig. 3.4.5.1 Mapping FBoxes

As client we are able to access the resources and distribute them to the local resources on the PCD with the following FBoxes:

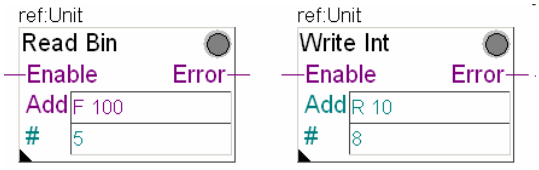


Fig. 3.4.5.2 Read/Write FBoxes

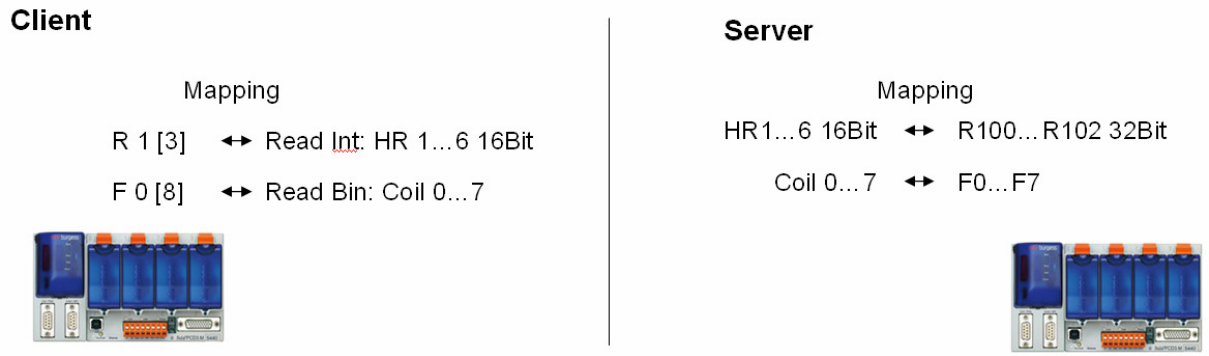


Fig. 3.4.5.3 Read/Write FBoxes

3.4.6 Mapping and UIDs

Each server PCD can have up to 9 Modbus UIDs. All these can be configured by the user. A separate mapping can be defined for each UID. Up to 10 mapping areas can be created per UID. The default mapping can always be used instead of the mapping that can be configured by the user. In this case, all resources are available. If other mappings are available, it should be ensured that no collisions are caused.

The UIDs are always valid for all Modbus ports (serial and TCP/UDP). Each UID can exist only once in a serial network.
UIDs can be reconfigured in run-time.

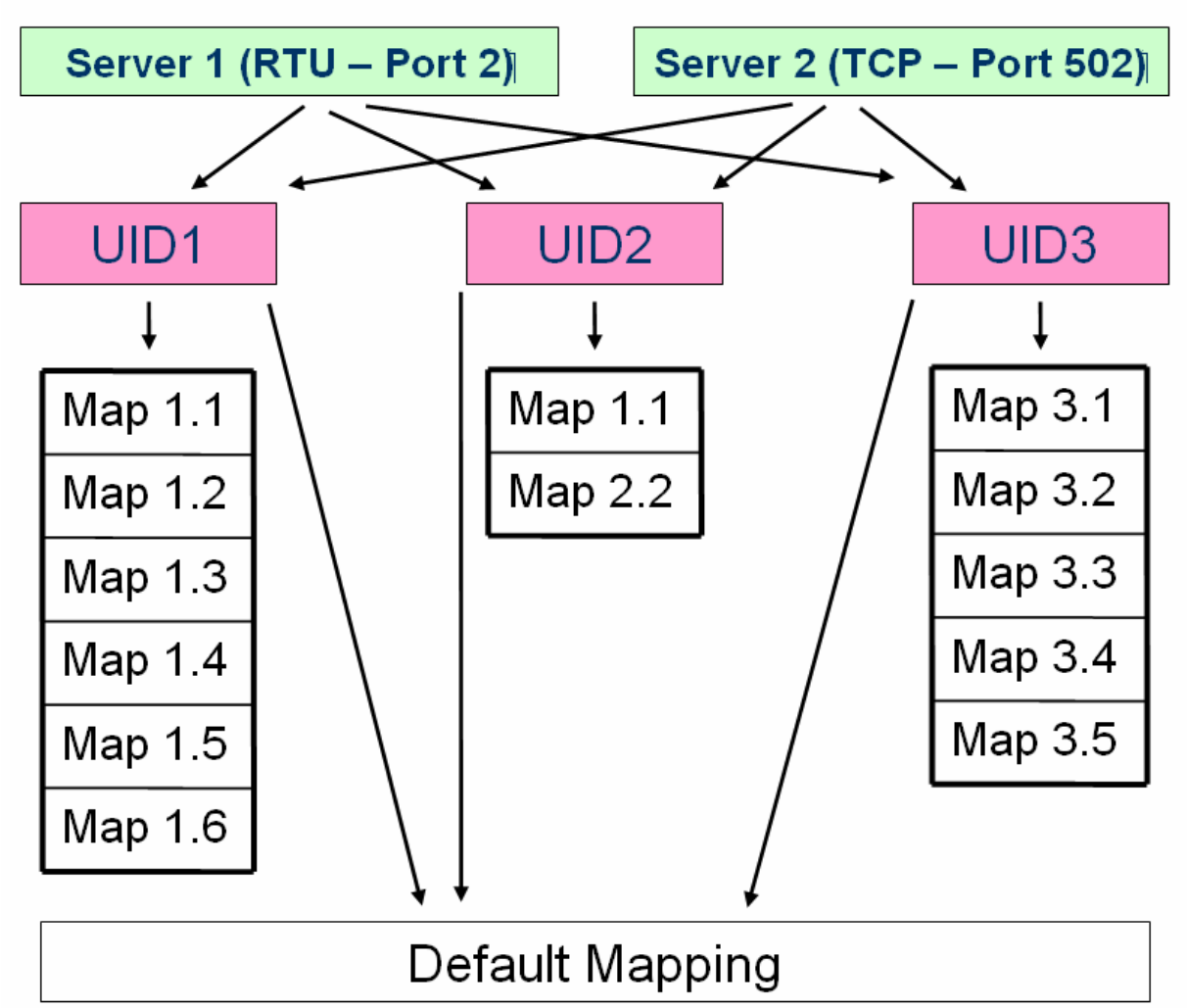


Fig. 3.4.4.1 Mapping

If no mapping is created, the default mapping is active. However, as soon as a mapping is created for a UID, it has priority.

UID 0 is the default UID for the serial Modbus. This UID is always available and can be used to send broadcast telegrams.

In the TCP/UDP communication, the default UID is always active likewise, but no default mapping is defined. Requests to a nonexistent UID is always answered by the default UID. As soon as a mapping is defined for the default UID, this is used for the default UID and all non-configured UIDs.

3.4.7 Channel Definition

A Modbus Channel is always a pair of a port and a protocol. Per example: TCP on Port 502 or ASCII on serial Port 2.

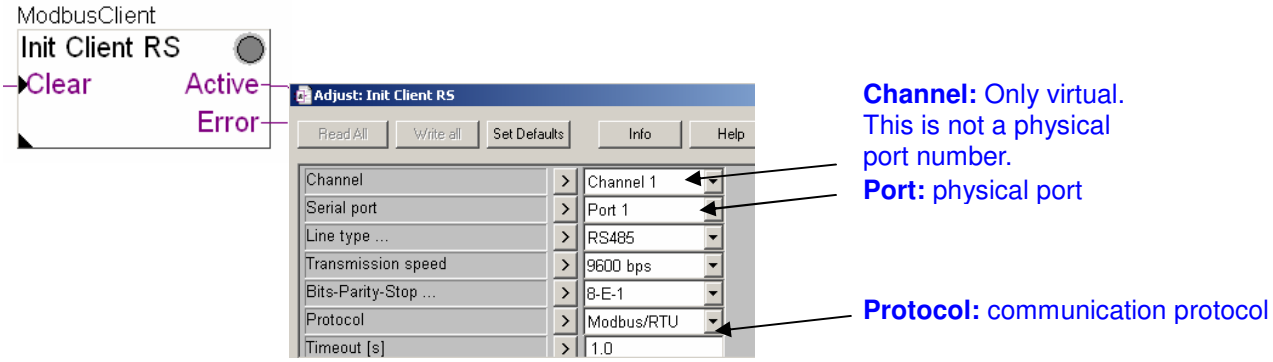


Bild 3.4.7.3 Read/Write FBoxen

Each channel can only be defined once. It is possible to define up to 4 Server-Channels on a PCD.

3.4.8 Connections

If a client requests data from a server, a connection is automatically opened. The user need not tend to the opening and closing of the connections.
All the same, it is important to know when a connection is opened or closed, as the number of connections is limited. The Modbus Manual 26/866 can be consulted for details.

3.4.9 Holes

If 32-bit values are mapped on the register, one can choose between two options. One can work with holes. Then the data are easy to interpret, because register addresses in the PCD agree with the holding register addresses in the Modbus. However in this case only the even registers are used, which keeps the odd registers unused for this purpose. Without holes it is more compact and saves more resources, but the allocation proves to be somewhat more complex.

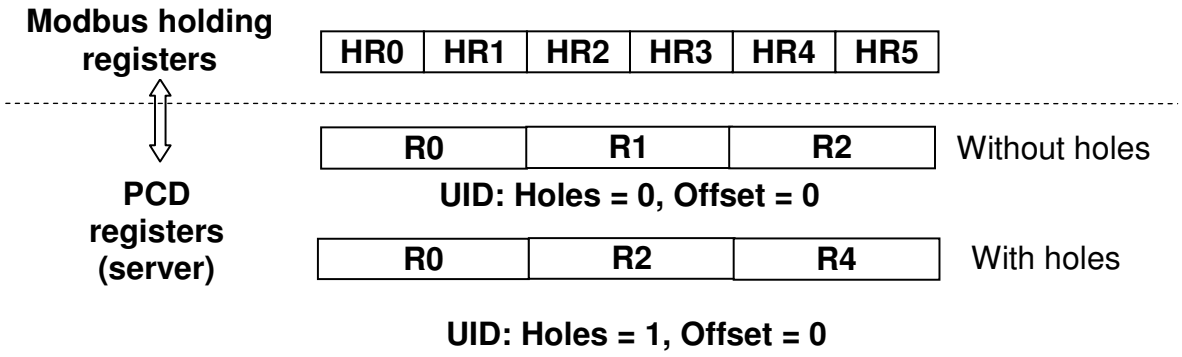


Fig. 3.4.9.1 Holes

3.4.10 Programming client

As client we need to know which UID to access and where the needed resources are mapped.
Furthermore it is necessary to verify if there are binary, 16bit or 32bit values to transfer and if an offset or holes are in use.

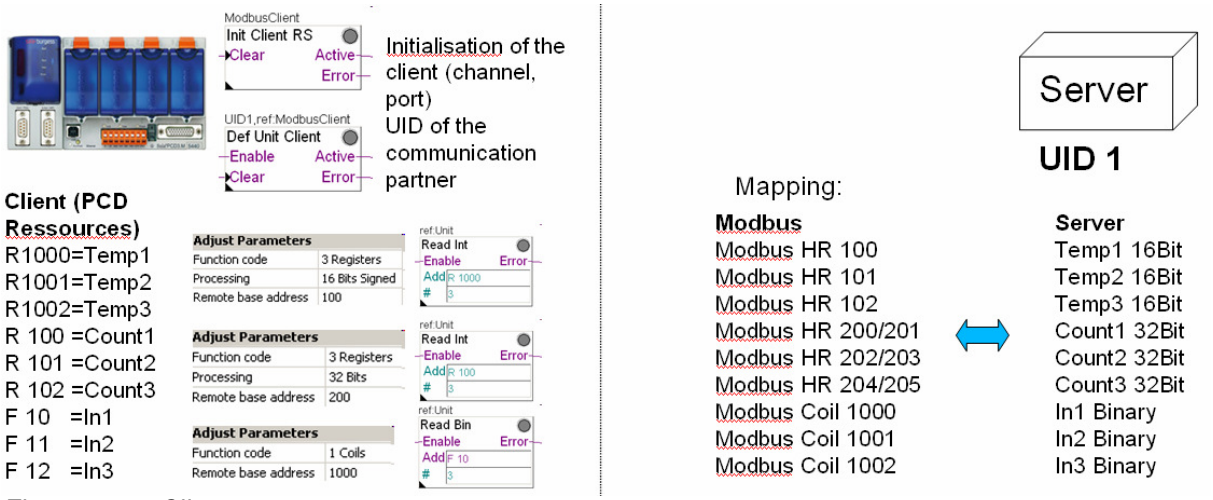


Fig. 3.4.10.1 Client

It is important to reserve the proper range for the received resources. This means that if the base address is register 1000 and we read 3 elements, we have to reserve register 1000, 1001 and 1002. PG5 will not recognize during compile that more than one register is in use. Therefore it is recommended to reserve an array of registers, per example R 1000 [3] or to make sure in another way that these registers are not used for another purpose or overwritten.

3.4.11 Programming server

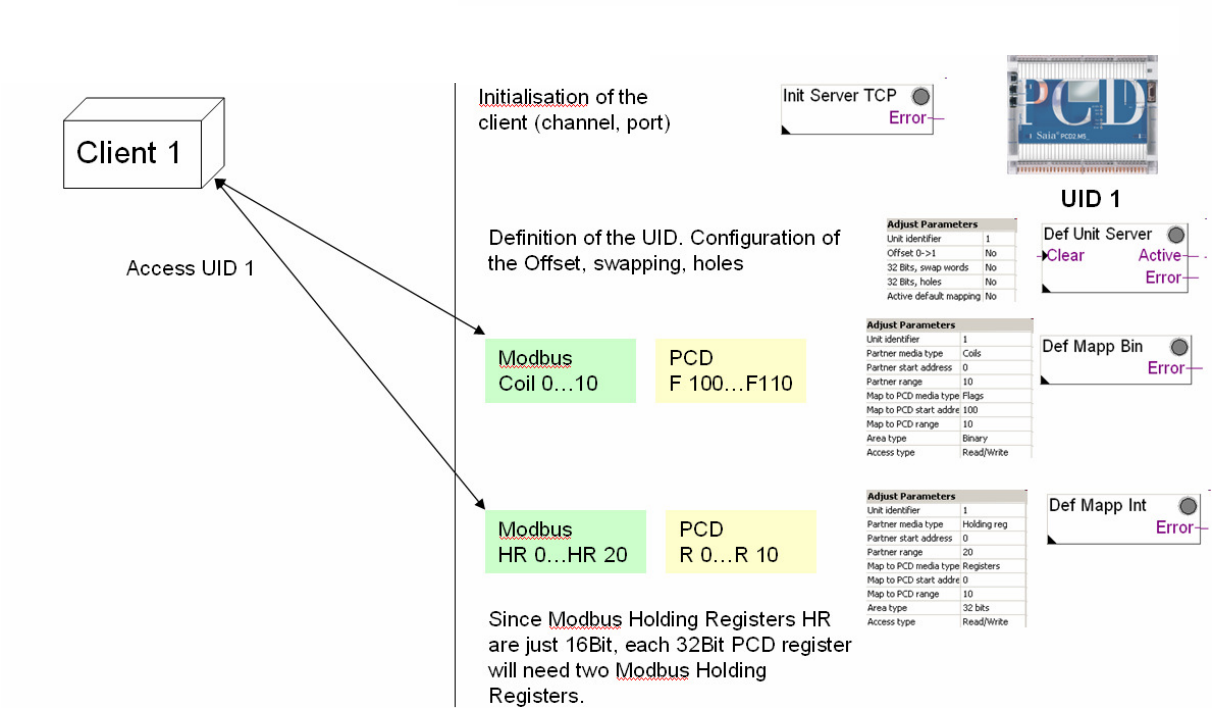


Fig. 3.4.11.1 Server

As server we provide the resources for each UID with the datamapping.

3.5 Comparison Saia solution ⇔ Engiby solution

If the Engiby library was used earlier, pay attention that the mapping is correctly adopted. The designations are partially different, which can lead to confusion. Therefore an overview of the different designations in the Engiby and Saia FBoxes is given here:

Saia FBoxes	**Engiby FBoxes**
32-bit swap words NO	LITTLE Endian
32-bit swap words YES	BIG Endian
Holding register	R
Input register	InR

3.6 Limits to consider

Summary of limitations to consider in the Modbus network:

- Maximum 247 servers (slaves) per bus for serial Modbus
- Maximum 10 channels in toto, thereof maximum 4 server channels (port+protocol)
- Maximum 9 user-specific UIDs (+default ID)
- Maximum 10 mapping areas per UID
- Serial modem connection is not supported
- Supported Modbus functions:
    - READ_COILS
    - READ-DISC_INPUT
    - READ_HOLD_REG
    - READ_INPUT_REG
    - WRITE_SINGLE_COIL
    - WRITE_SINGLE_REG (16-bit only)
    - WRITE_MULTIPLE_COILS
    - WRITE_MULTIPLE_REGS

4 Description of the project example

There are 3 Saia PCD's in this project example. A communication takes place over IP (Modbus TCP) between the "Client" and the "Modbus_Server" and a serial communication RS 485 (Modbus ASCII) takes place between the "Client" and the "Server_RS".

4.1 Datenaustausch über Modbus

The PCD Resources (R, F, T, C, I, O…) can not be directly transferred to the Modbus. On Modbus there are Holding Registers and Coils (read/write) or Input Register and Discret Inputs (read only) available. Because of that a mapping for each UID is necessary or the Default Mapping can be used. The mapping is the link between the PCD resources and the Modbus resources. Details about the mapping can be found in the Modbus manual 26/866 and in chapter 3.4.4. above.

4.1.1 Communication IP

The following data exchange takes place between the PCD „Client“ and the PCD „Modbus_Server“:
On the „Modbus_Server“ the clock is transferred to the registers 100, 101 und 102 (32 Bit registers). It will be transferred over Modbus to the „Client“ and there stored to the registers 100, 101 und 102. The clock on the “Client” can now be updated with the clock from the “Modbus_Server”
The flags 1000 bis 1007 of the „Client“ are transferred over Modbus to the „Modbus_Server“ to the flags 1000 bis 1007. The flags are counting up on the “Client” and if the communication is working correctly this up-counting can be observed on the “Modbus_Server as well.
The following data-mapping is configured:

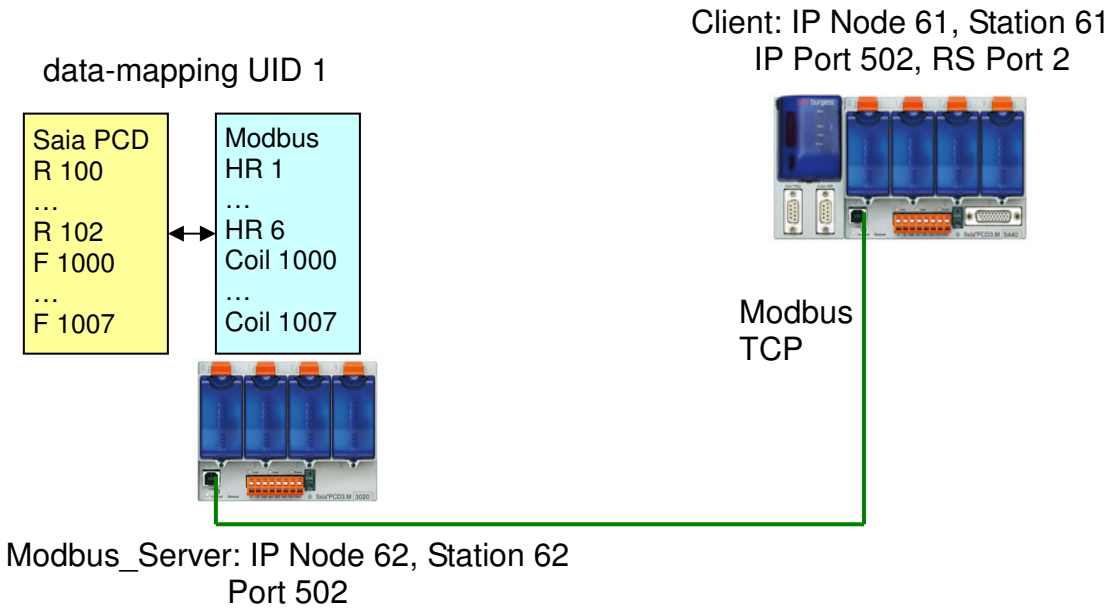


Bild 4.1.1.1 Modbus network-topology

Client IP Saia PCD Medias (R u. F)	**Modbus Holding Register / Coils**	**Server IP Saia PCD Medias (R u. F)**
R 100…R 102 (32 Bit)	<= HR 1…6 (16 Bit) <=	R 100…R 102 (32 Bit)
F 1000…F 1007	=> Coil 1000…1007=>	F 1000…F 1007

4.1.2 Communication RS 485

Data exchange between the PCD „Client“ and the PCD „Server_RS“:
The „Client“ is reading 8 registers and 8 flags from the “Server RS”. The registers 0 to 7 and the flags 0 to 7 are read from the „Server_RS“. The registers are configured as 16 Bit signed values. In this case there is no specific mapping defined, means that the default mapping is in use. For 16 Bit signed values this means that all PCD registers are available on the Modbus in Holding Register 1…10000. The flags are available in the Coils 1…10000.
The following data-mapping is in use:

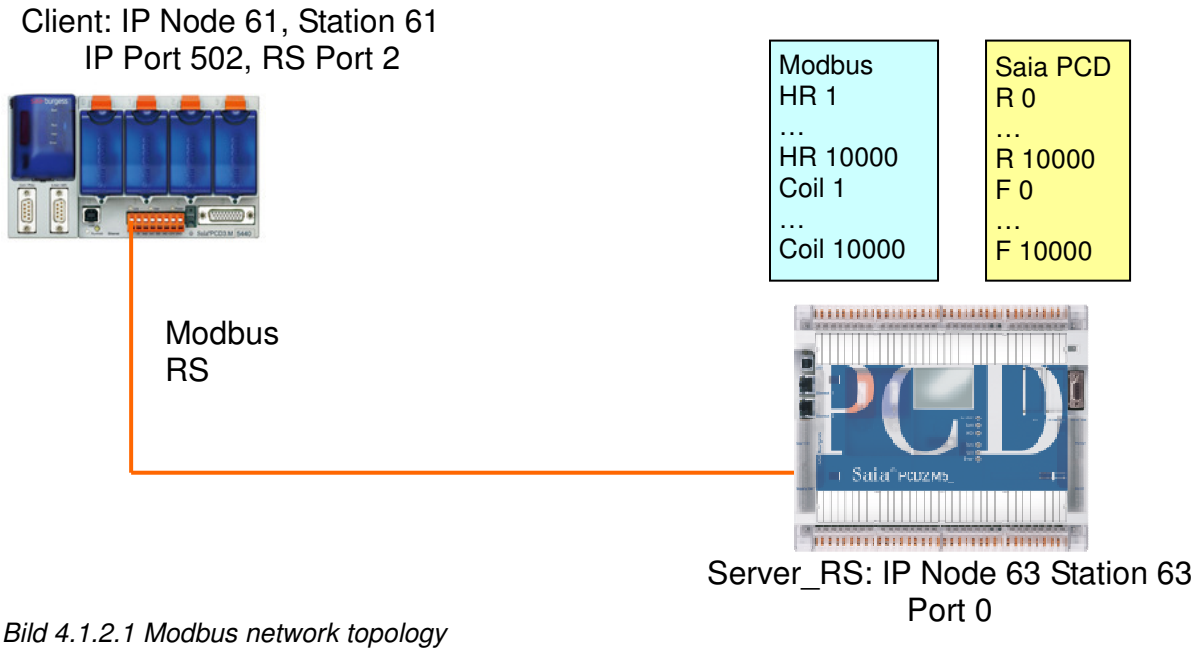


Bild 4.1.2.1 Modbus network topology

Client RS Saia PCD Medias (R u. F)	Modbus Holding Register / Coils	Server RS Saia PCD Medias (R u. F)
R 0…R 7 (16 Bit signed)	<= HR 1…8 (16 Bit) <=	R 0…R 7 (16 Bit signed)
F 100…F 107	=> Coil 1…8=>	F 0…F 7

5 Preparation of the sample project

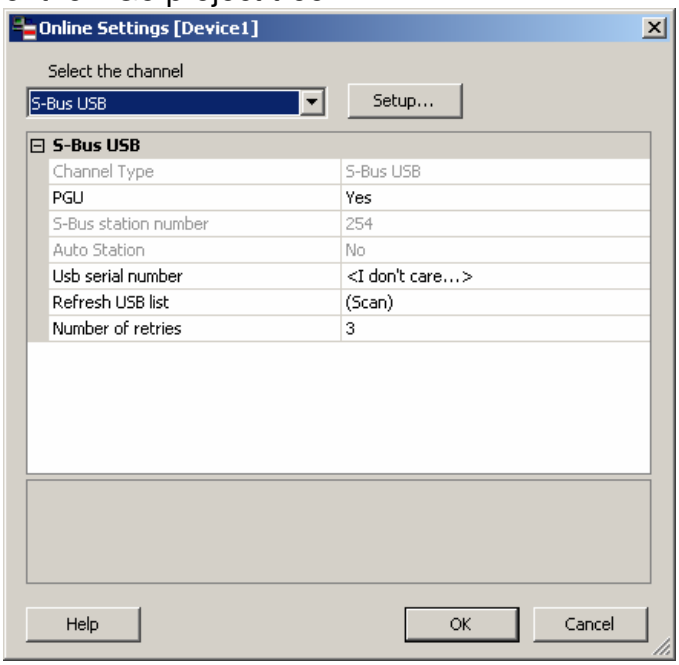
To import the project into the PG5, use the function "Restore" from the PG5 Project Manager menu "Project".

In order to make a Modbus communication possible, at least two stations should be configured and programmed.

5.1 Configuring PCD

3 steps are required to prepare the PCD:

Establish online connection to the PCD
Before a connection can be established, PG5 should "know" over which medium/cable the PCD should be accessed. This is defined in the "Online settings" of the PG5 project tree:



Here "S-Bus USB" is selected as "Channel", as the IP configuration is not loaded at the moment. The option PGU should be activated.

After these settings, it can be checked with the "Online Configurator" whether the communication functions.

Fig. 5.1.1 Online Settings

Configuring hardware
Settings such as the IP address, usage of memory and activation of the "Run/Stop" switches of the PCD are configured using Device Configurator. Device configurator can be opened from PG5 project tree directly under the "Online settings".

Before you proceed further, please configure a not yet assigned IP address and subnet mask compatible for your network in Device Configuration.

To load the configuration in the controller select ‘Download configuration’ from device configurator or execute a Right Click on the CPU in the project tree. In the Context Menu there is as well an option “Download Configuration” available. When asked what should be loaded in the controller, the option "Memory allocation" should also be selected at the first download, so that the memory is correctly configured.

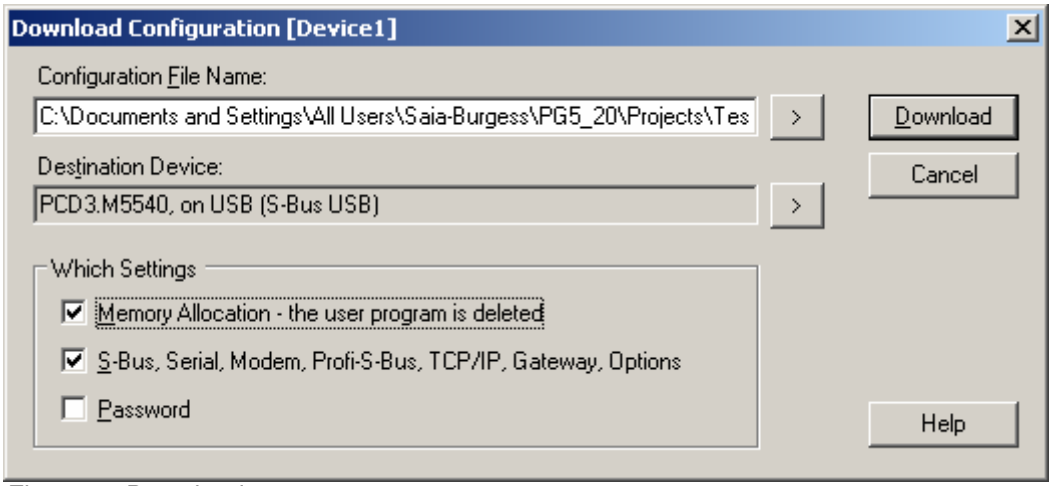


Fig. 5.1.2 Download

If the exact type of the PCD is unknown, or if the existing configuration of the hardware should not be changed, the button "Upload" in the window "Hardware settings" can also be used. Consequently, the current configuration of the PCD is transferred to the PG5 project.

The hardware settings are to be accordingly adapted in all PCDs that will be used:
Client, S-Bus address 61, IP node 61: This is the client (master) IP and RS.
Modbus server, SBus address 62, IP node 62: This is the server (slave) IP.
Server_RS, SBus address 63, IP node 63: This is the server (slave) RS.

5.2 Further configuration

The example is construed in such a way that there is a client (master) on which one Fupla file is located for each of the communication possibilities IP and RS.

If only one Modbus communication should be implemented on IP, the other file is to be deactivated. If only one RS communication is desired, the IP file is to be accordingly deactivated and the RS file is to be activated.

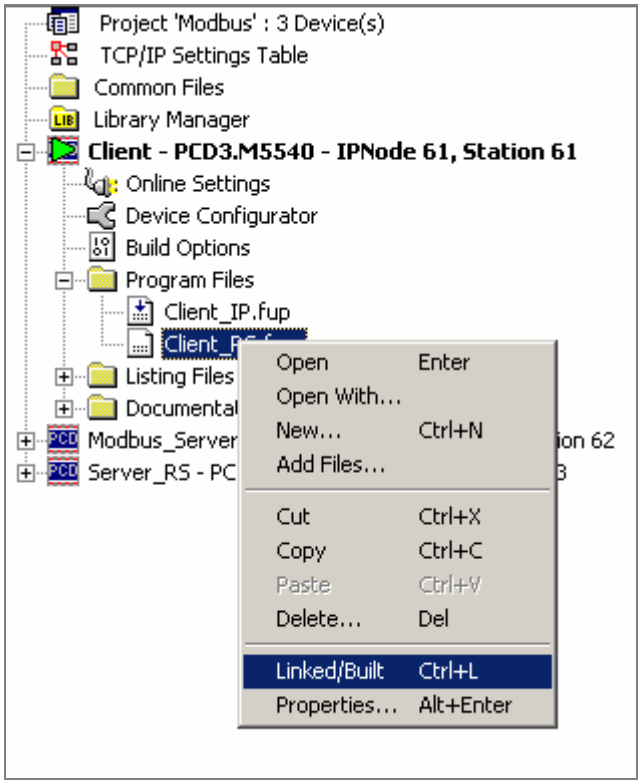
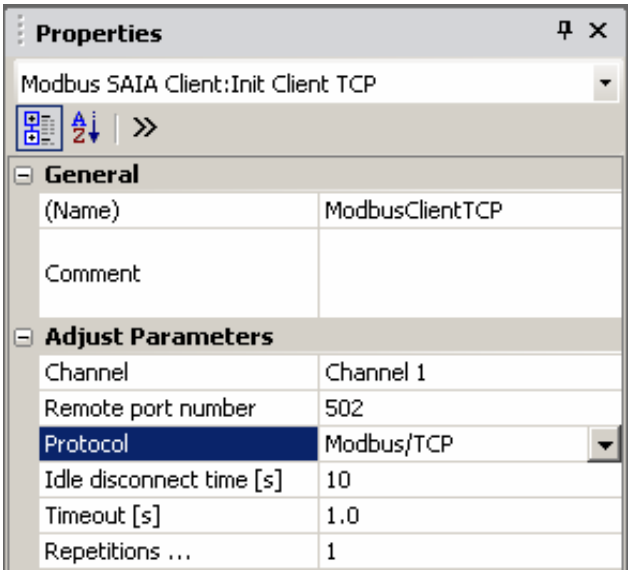


Fig. 5.2.1. Links

5.2.1 Communication IP



In the Fupla Client_IP.fup, some settings are to be checked and adapted according to their infrastructure A Modbus TCP communication is configured. If Modbus UDP is needed, it can be set in the FBOX Init Client TCP. The important thing in such a case is that the communication partner also gets the same settings.

Fig. 5.2.1.1 Client TCP

In the FBox Def Unit Client, the IP address of the communication partner needs to be adapted.

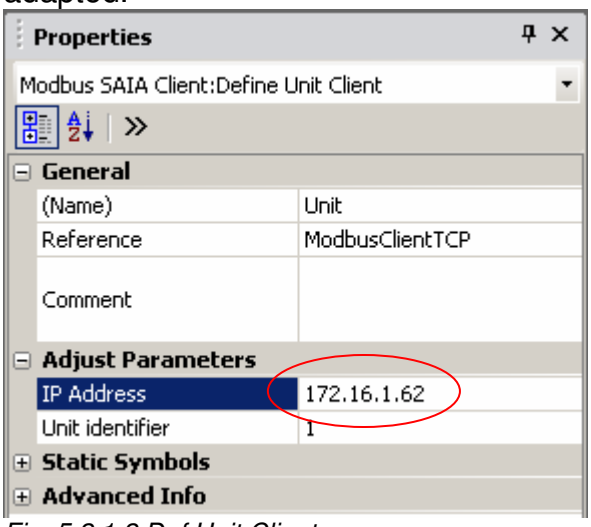


Fig. 5.2.1.2 Def Unit Client

In the FBox Def Unit Client, the IP address of the communication partner needs to be adapted.

5.2.2 Communication RS

For a serial communication, the settings in the Fupla Client_RS.fup are to be checked and if necessary adapted. The serial communication takes place via the onboard RS 485 port 2 of the PCD3. If another port is used, it is to be adapted in the FBox Init Client RS. If the baud rate or the protocol is changed, it is to be adapted in the communication partner as well.

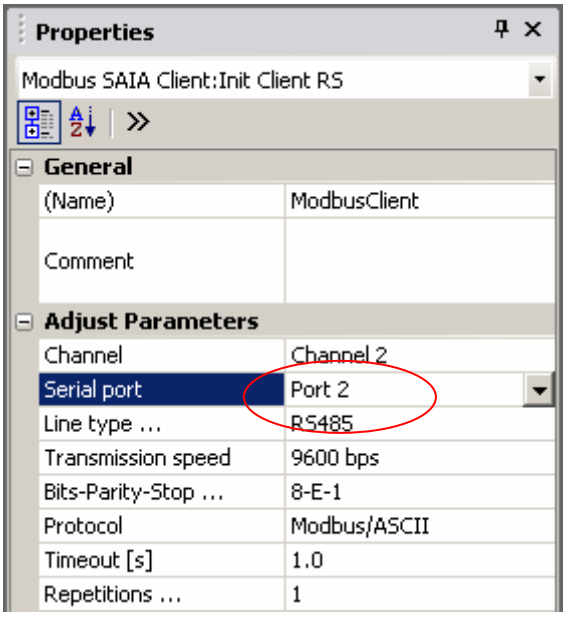


Fig. 5.2.2.1 Init Client RS

The communication port is also to be adapted in the Server_RS, unless it is communicated via the onboard port 2 of the PCD3.

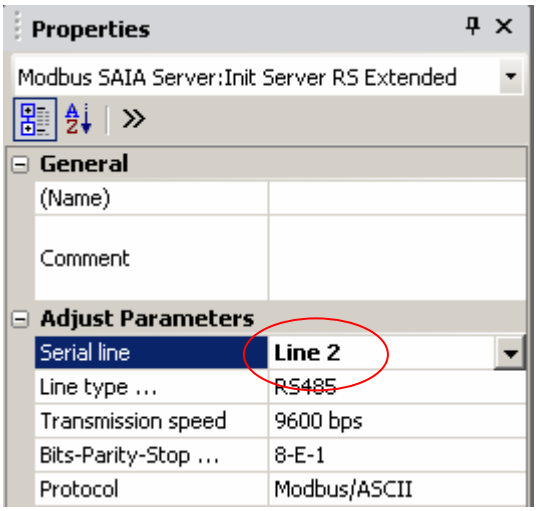


Fig. 5.2.2.1 Init Server RS Extended

5.3 Programming PCD

Loading programme on the control

Now only the programming of the PCD is missing. For that, the programme should be translated ("built") first. For this purpose you can use the "Rebuild all" button.

After the "build" of the programme is correctly carried out, you can load the programme on the PCD with the button "Download programme". The PCD is thus prepared. According to the setting of your PG5, the control automatically goes into RUN after the download. If this is not the case, set the controls in RUN.

6 Programming of the PCD

This section contains a brief description of the application.

6.1 Client

This PCD acts with the Fupla Client_IP as TCP Client and with the Fupla Client_RS as RS Client (RS 485).

6.1.1 Client_IP

Page 1
The Modbus communication is programmed on the first page of this Fupla programme.

The FBox Init Client TCP defines the communication channel, the remote port number, the protocol and the timing behaviour.

The FBox Def Unit Client defines the IP address of the server and the unit identifier. The flag Enable Def should be activated.

Integer values are read with the FBox Read Int. The field Add indicates the address where the read elements are copied (Register: ReadIP, R 100). # defines the number of registers to be read (10 = R 100 to R 109).

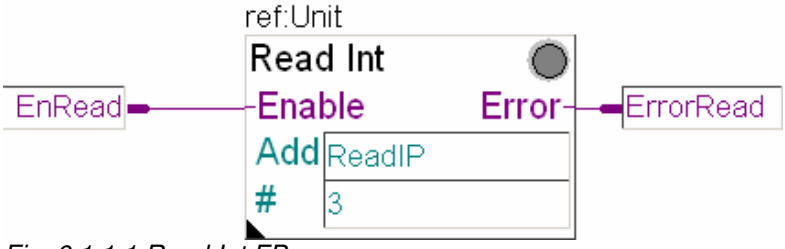


Fig. 6.1.1.1 Read Int FBox

Registers (32 Bit) are read. The base address of the Modbus Holding Registers is in this case the address 1. On the Modbus Server the data to transfer (PCD Register 100-102) is mapped with the according DataMapping to the Modbus Holding Registers 1 to 6. On Modbus there are only 16 Bit Holding Registers available. The Saia PCD Registers to transmit contain 32 Bit values. This means that we will need 2 Modbus Holding Registers for each Saia PCD Register.

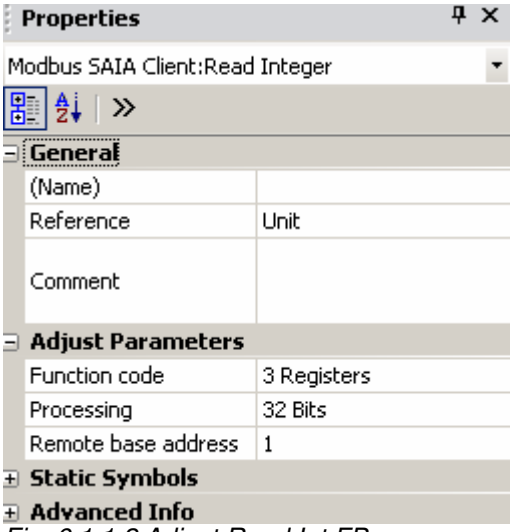


Fig. 6.1.1.2 Adjust Read Int FBox

The EnRead flag should be put up, so that data are read.
Important: If more than one Register is overwritten by the data from the Modbus (3 Registers in this case), it is necessary to reserve the correct range of registers for the received data (Array of 3 PCD Registers).

Binary values are written with the FBox Write Bin. 8 elements are written. These flags that are to be transmitted are flag WriteBin = F 1000 and the subsequent flags (F 1000 to F 1007).
The EnWrite flag should be put up to start the write operation. On the Modbus, these Flags are mapped to the Coils 1000 to 1007.

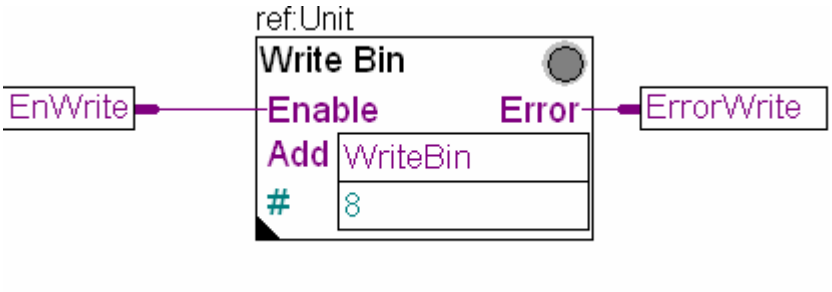


Fig. 6.1.1.3 Write Bin FBox

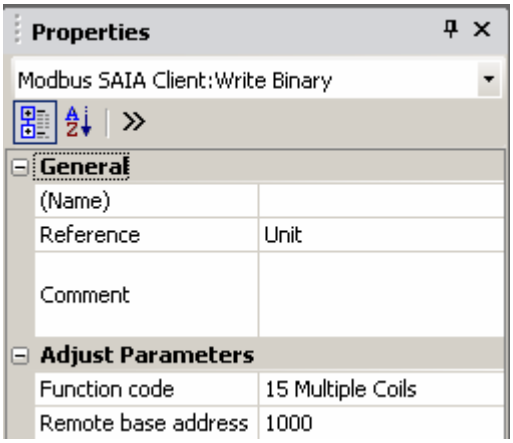


Fig. 6.1.1.3 Adjust Write Bin FBox

Page 2
We obtain the time and date of the other control with the first 3 registers received via

the Modbus. On this page, the time is then transmitted to the hardware clock of the control. As we obtain the time in hours, minutes and seconds in the first register, a division by 100 is necessary to obtain only hours and minutes, as the FBox Write Clock anticipates.

Page 3
Here it is taken care that the flags that are to be transmitted always change their status, so that it can be immediately recognised whether the communication runs at the opposite side.

6.1.2 Client_RS

Page 1

The initialisation of the client RS takes place on this page. A channel (communication channel = virtual), a port (physical port on the PCD), the physical interface, communications parameters, protocol and timing are defined with the FBox Init Client RS.

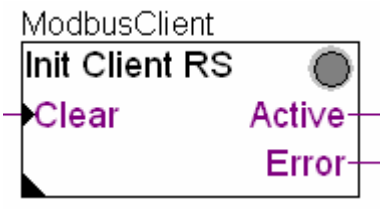


Fig. 6.1.2.1 Init Client RS FBox

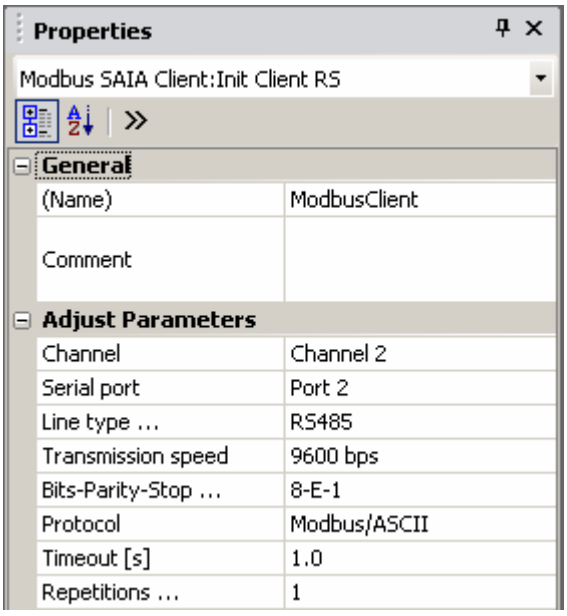


Fig. 6.1.2.2 Adjust Init Client RS FBox

The FBox Def Unit Client defines the unit identifier UID of the communication partner. There is no need to enter the IP address in this case. The field IP address can contain any one address.

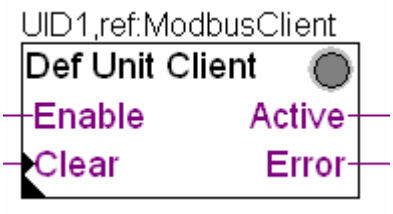


Fig. 6.1.2.3 Def Unit Client FBox

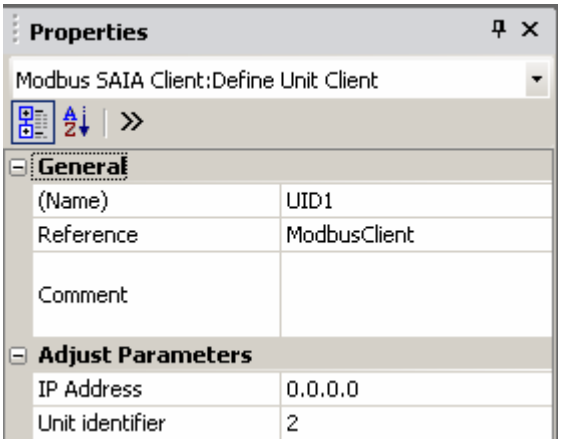


Fig. 6.1.2.4 Adjust Unit Client FBox

The EnDefRS flag should be put up.

Fig. 6.1.2.4 Read Int FBox

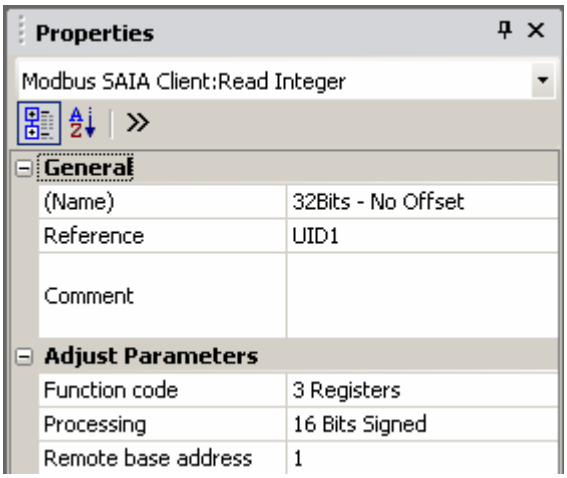


Fig. 6.1.2.5 Adjust Read Int FBox

16-bit signed register is read from the Modbus base address 1. These are filed in the address ReadInt R 0 to R 9. Since the Default Mapping is activated on the Server, the PCD Registers 0 to 9 (16 Bit signed) are mapped to the Modbus Holdingregisters 1 to 10.

Default Mapping

	Modbus Request			PCD			
Access Type	**MediaType**	**Start Addr.**	**Range**	**MediaType**	**Start Addr.**	**Range**	**AreaType**
ReadWrite	MB_HOLDING_REG_MEDIA	1	10000	PCD_REG_MEDIA	0	10000	MB_AREA_16BITS_SIGNED
ReadWrite	MB_HOLDING_REG_MEDIA	10001	10000	PCD_REG_MEDIA	0	10000	MB_AREA_32BITS
ReadWrite	MB_HOLDING_REG_MEDIA	20001	10000	PCD_REG_MEDIA	0	10000	MB_AREA_32BITS_IEEE
ReadOnly	MB_INPUT_REG_MEDIA	1	10000	PCD_TC_MEDIA	0	10000	MB_AREA_16BITS_SIGNED
ReadOnly	MB_INPUT_REG_MEDIA	10001	10000	PCD_TC_MEDIA	0	10000	MB_AREA_32BITS
ReadWrite	MB_COILS_MEDIA	1	10000	PCD_FLAG_MEDIA	0	10000	MB_AREA_COILS
ReadOnly	MB_DISCR_INPUT_MEDIA	1	10000	PCD_IO_MEDIA	0	10000	MB_AREA_COILS

Bild 6.1.2.6 Default Mapping

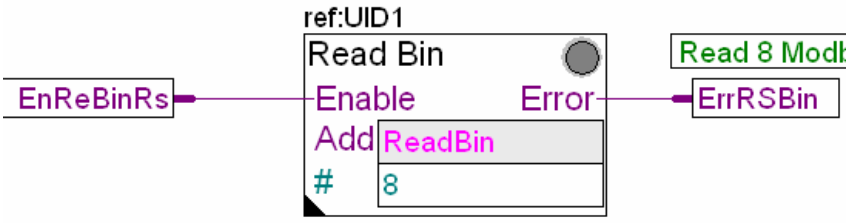


Fig. 6.1.2.7 Read Bin FBox

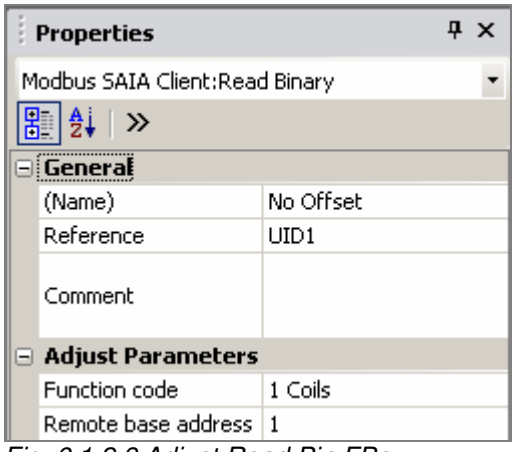


Fig. 6.1.2.8 Adjust Read Bin FBox

8 binary values are read from Modbus coil 1 onwards. They are filed in the flags ReadBin F 100 to F 107.

Page 2

The flags 100 to 107 are changed from Bin to Int. If the transmission is successful, it is incremented here.
Read registers are added with the Additions FBox. Here too the value should be continuously changed in active transmission.

6.1.3 Modbus server (IP)

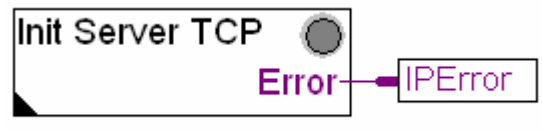


Fig. 6.1.3.1 Init Server TCP FBox

The port and protocol for the IP connection are defined with this FBox.

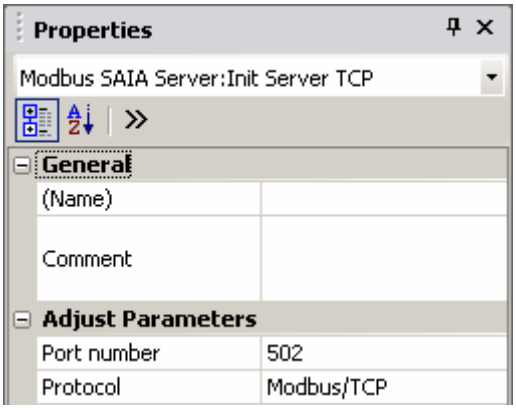


Fig. 6.1.3.2 Adjust Init Server TCP FBox

The FBox Def Unit server defines the unit identifier UID. In addition an offset can be defined, so that it starts with 1 and not 0. If required, word swapping can be activated, and the holes can be switched on if need be.

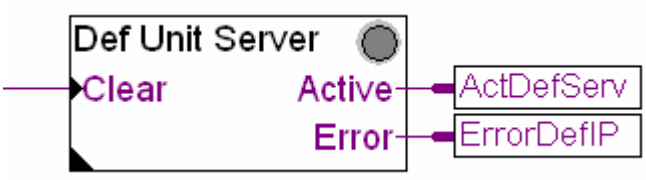


Fig. 6.1.3.3 Def Unit Server TCP FBox

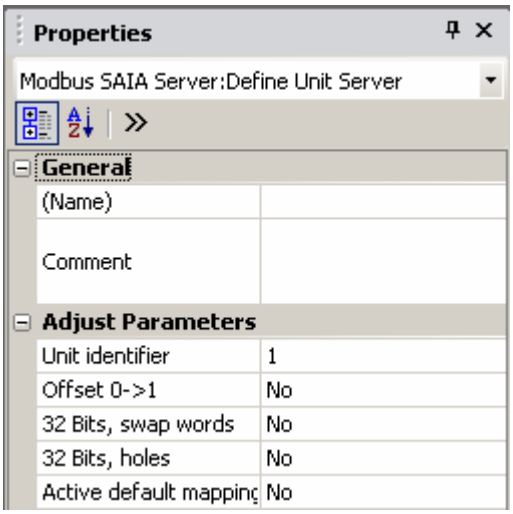


Fig. 5.1.3.4 Adjust Def Unit Server TCP FBox

A mapping for the binary area is created with the FBox Def Mapp Bin.

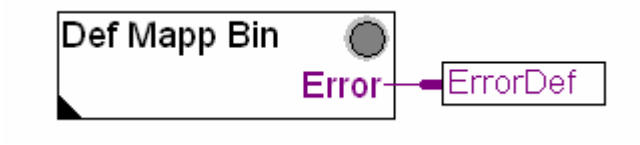


Fig. 6.1.3.5 Def Mapp Bin FBox

In this case the Modbus Coils 1000 to 1007 are mapped to the PCD Flags 1000 to 1007.

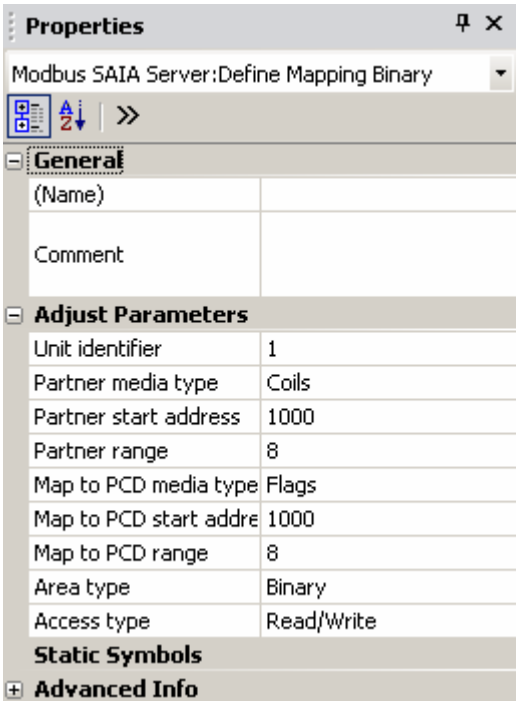


Fig. 6.1.3.6 Adjust Def Mapp Bin FBox

A mapping for the integer area is created with the FBox Def Mapp Int.

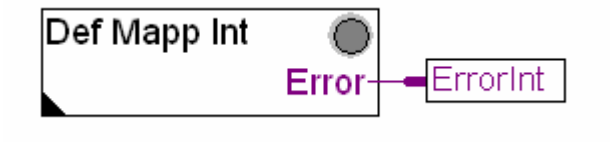


Fig. 6.1.3.7 Def Mapp Int FBox

The Modbus Holding Registers 1 to 6 are mapped to the PCD Registers 100 to 102. Since the PCD Registers are 32 Bit and the Modbus Holding Registers only 16 Bit, two Holding Registers are needed for each PCD Register.

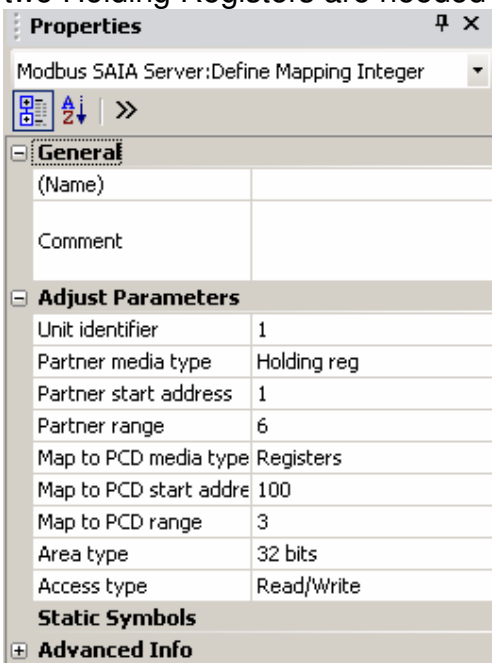


Fig. 6.1.3.8 Adjust Def Mapp Int FBox

Page 2
The time and date of the PCD is read here. Then this is transmitted to the client. Further the received flags are converted from binary to integer. An increment is seen in continuous communication.

6.1.4 Server_RS

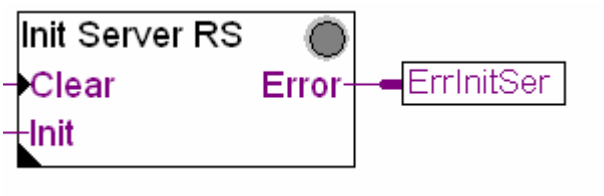


Fig. 6.1.4.1 Init Server RS FBox

This FBox initialises the appropriate interface for Modbus.

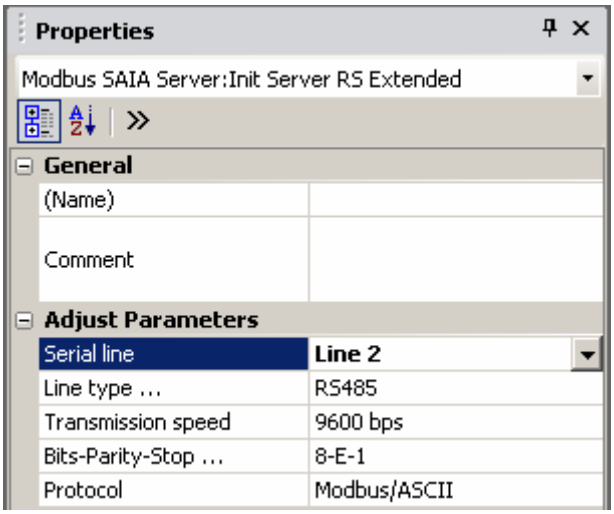


Fig. 6.1.4.2 Adjust Init Server RS FBox

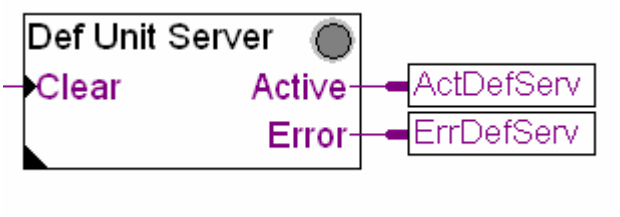


Fig. 6.1.4.3 Def Unit Server FBox

The unit identifier UID is set with this FBox and defined whether an offset should be made. Word swapping and holes can be activated. The default mapping can be moreover activated.

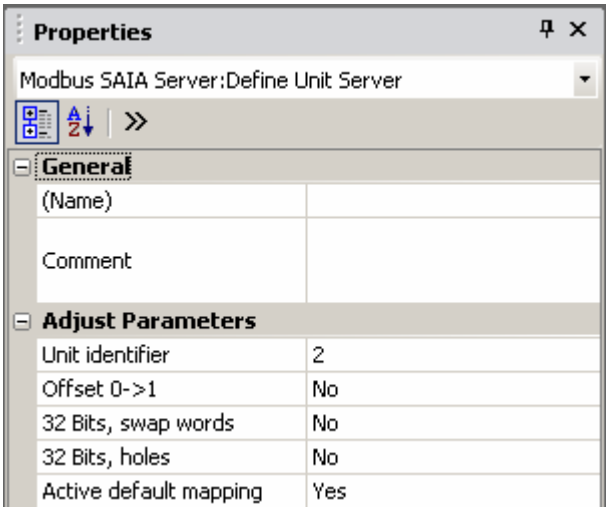


Fig. 6.1.4.4 Adjust Def Unit Server FBox

The following FBoxes provide for a flashing of the flags that are to be transmitted and for a conversion of the values in the registers that are to be transmitted.

6.2 Transmitted data

6.2.1 IP Ethernet

The Registers 100 to 102 of the Server IP are transmitted to the Modbus Holding Registers 1 to 6. On the Client IP these Holding Registers are transmitted to the Registers 100 to 102.
The Flags 1000 to 1007 of the Client are transmitted to the Coils 1000 to 100. On the Server these Coils 1000 to 1007 are mapped to the Flags 1000 to 100.

6.2.2 Serial RS 485

The flags 0 to 7 are transmitted from Server_RS to the Modbus coils 1 to 8. These are then recorded in the Client_RS on the flags 100 to 107.

The registers 0 to 9 (16 bits) are transmitted to the Modbus holding registers 1 to 10 and recorded in the client on the registers 0 to 9.

7 Troubleshooting

Symptom	Possible cause	Solution
Default mapping is not read, even though the FBox "Default Mapping" is set = "Yes".	As soon as a mapping is configured for a UID, the default mapping is deactivated.	Delete all mappings for this UID on the PCD.
Data do not come in, are not transmitted, but the FBox is green.	The data possibly appear in another address.	Check mapping
FBox is red, communication doesn't function / only partially	Communication parameters don't agree with the communication partner. Port is configured otherwise (HW settings). Incorrect wiring	Check communication parameters, configuration and wiring
A project, which was previously implemented with the Engiby library, does not function anymore with the Saia Modbus library.	The mapping possibly doesn't agree. The different designations can lead to confusion.	Check whether the mapping has been correctly adopted according to section 3.6.1.
The "Define unit server" FBox has a "Range error".	The range is incorrectly defined (e.g. wrong length) or only a "half" of the PCD register is read by a 32-bit mapping (e.g. only HR 0 instead of 0 and 1).	Check range and examine whether 32-bit registers are correctly read.

8 References

Subject	Document	No.
Modbus	Modbus Manual	26/866
Miscellaneous	Saia® FAQ Manager www.sbc-support.ch/faq	-