

Modbus User Interface Document

0	Content	
0.1	Document History	0-3
0.2	About this manual	0-3
0.3	Brands and trademarks	0-3
1	The Modbus Protocol	
1.1	ISO / OSI Model.....	1-1
1.2	Modbus Communication	1-1
1.3	Modbus Media	1-3
2	SBC Modbus	
2.1	SBC Modbus Features.....	2-1
2.2	Diagnostic	2-3
2.2.1	Client / Server Diagnostic Flags.....	2-3
2.2.2	Diagnostic Register.....	2-4
3	SBC Modbus Server	
3.1	Server Instance.....	3-1
3.2	Server UID	3-2
3.3	Media Mapping	3-2
3.4	Default Mapping.....	3-5
4	SBC Modbus Client	
4.1	Client Channel	4-1
4.2	Modbus Requests	4-2
5	Modbus FBox Library	
6	Modbus System Functions Library	
6.1	Introduction	6-1
6.2	Parameters	6-2
6.3	Error codes	6-6
6.3.1	CSF error codes.....	6-6
6.3.2	ModbusShell error codes	6-7
6.3.3	ModbusDriver error codes	6-7
6.4	Function Codes.....	6-8
6.5	Protocols.....	6-8
6.6	Modbus Data Types	6-8
6.7	Modbus Area Types (Server)	6-8
6.8	Modbus Area Access Types (Server).....	6-9
6.9	Processing Types (Client).....	6-9
6.10	Saia PCD® Media Types	6-9
6.11	Exception Codes.....	6-10



7 Modbus CSF Specification

7.1	S.SF.Modbus.InitSerialPort (Client / Server).....	7-1
7.2	S.SF.Modbus.OpenChannel (Client).....	7-2
7.3	S.SF.Modbus.SendReadRequest / S.SF.Modbus.SendWriteRequest (Client)	7-4
7.4	S.SF.Modbus.InitServer (Server)	7-6
7.5	S.SF.Modbus.InitUID (Server).....	7-7
7.6	S.SF.Modbus.InitMap (Server).....	7-8

A Appendix

A.1	Icons	A-1
A.2	List of abbreviations	A-2
A.3	Contact.....	A-3

0.1 Document History

Version	Changes	Published	Remarks
EN01	-	2009-06-18	New edition
EN01	-	2009-06-26	Text in drawing "Figure 7"
EN01	2009-08-12	2009-09-21	fixed in chapter 7
EN01		2010-02-11	Fixed table in chapter 7.2
EN02		2011-05-24	Modifications in some chaptres. Chapter 7.2, added text in CloseTimeout; S.Modbus changed to S.SF.Modbus.
EN03	-	2013-10-25	new Logo and new company name
ENG04	-	2019-01-04	Chapter 6.4 "Function codes": values changed

0.2 About this manual

See the section in the appendix in relation to some of the terms, abbreviations and the references used in this manual.

0.3 Brands and trademarks

Saia PCD® and Saia PG5® are registered trademarks of Saia-Burgess Controls AG.

Technical modifications are based on the current state-of-the-art technology.

Saia-Burgess Controls AG, 2019. © All rights reserved.

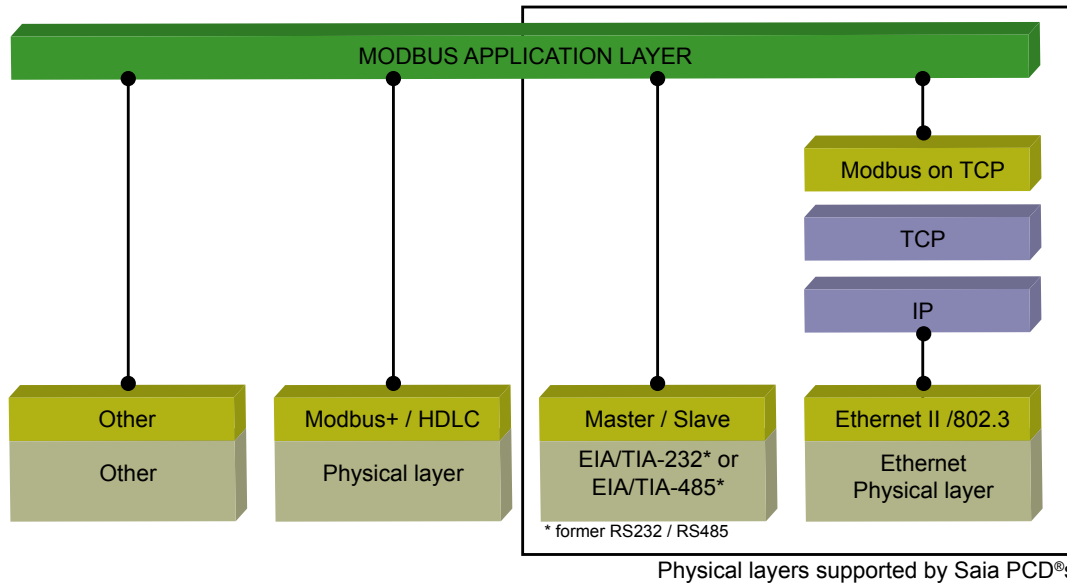
Published in Switzerland

1 The Modbus Protocol

1.1 ISO / OSI Model

1

Modbus is an application layer messaging protocol that enables the communication between one master and many slaves. It can be implemented on serial lines (RS-485, RS-232, RS-422) in RTU and ASCII modes or on Ethernet via TCP and UDP. In case of TCP / UDP, the public port 502 is reserved for Modbus but other ports can be defined.



1.2 Modbus Communication

Modbus is a request / reply protocol, which offers services specified by function codes. 127 public and user defined function codes are available, which can be used for data access (Read, Write), diagnostic, and other services.

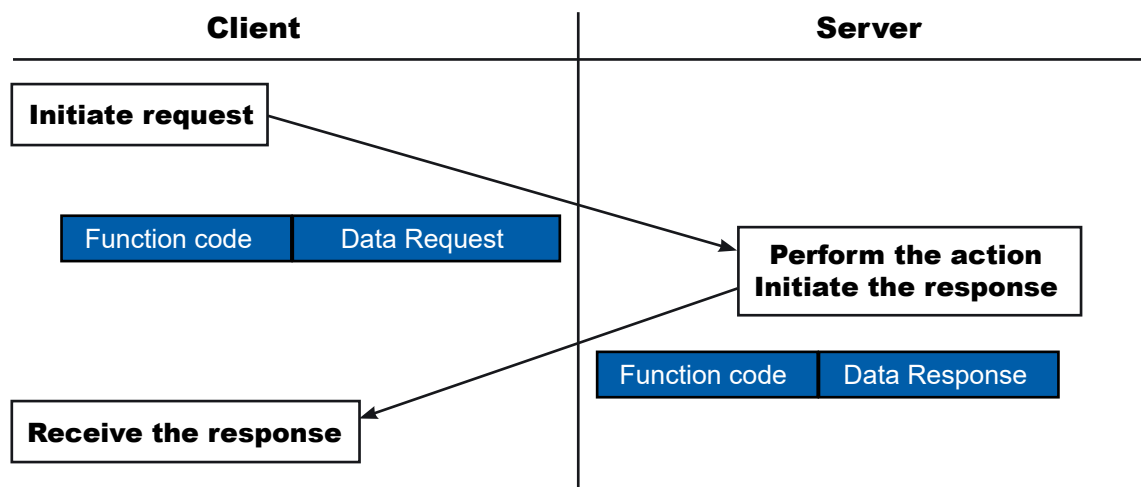


Figure 2: Modbus transaction (Error free)

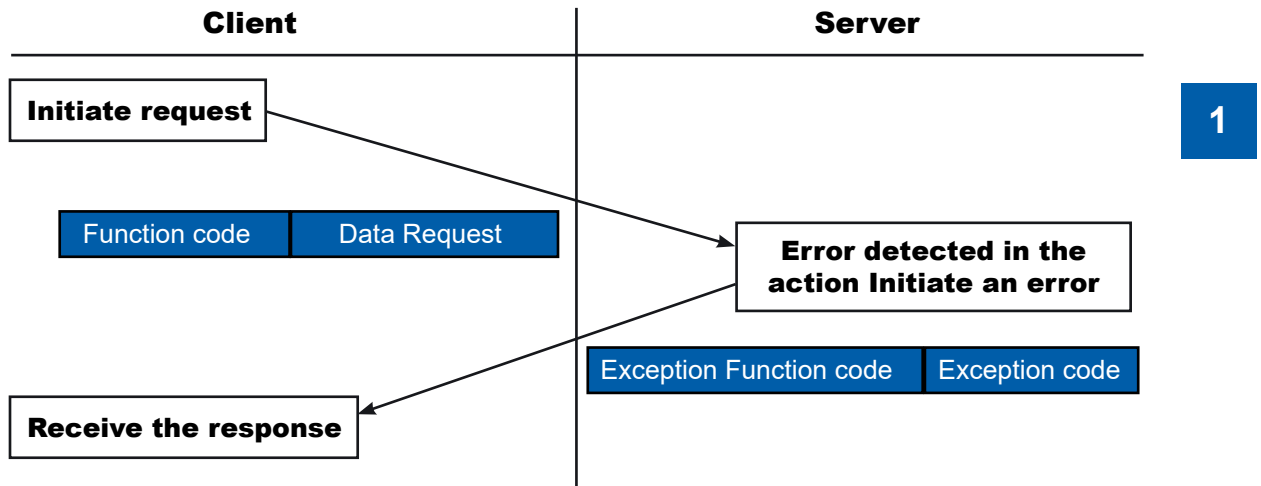


Figure 3: Modbus transaction (Exception response)

In serial mode, at the most 247 slaves can be present on the bus, each having at least one unique address, called Unit Identifier (UID).

RTU is the default mode for Modbus over serial lines. This mode is time controlled: intercharacter and interframe times must be respected. In case of a too long inter-character time the telegram is rejected.

ASCII mode can also be used for Modbus over serial lines. In this case, the communication is controlled by specific start and stop characters. But as one data byte is sent using 2 characters, twice more data than RTU are transmitted.

The Modbus communication is based on specific frames that encapsulate at least one function code and the useful data. For Modbus over serial line, a CRC is appended as well as an address. The address can also be used for Modbus over IP.

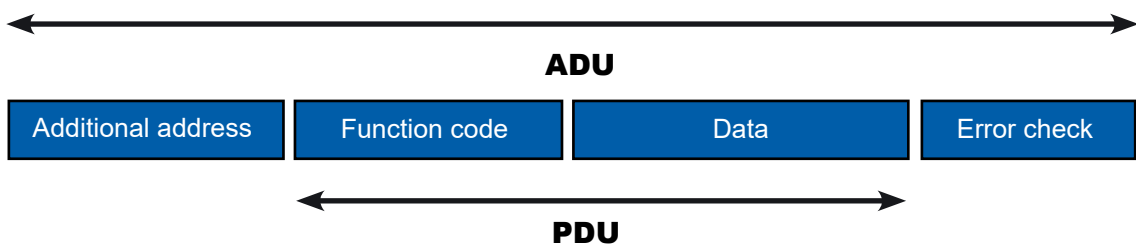


Figure 4: General Modbus frame

2 types of requests can be sent:

- Read requests: Data = Modbus address + number of elements to read
→ Response: Data = number of data bytes + data
- Write requests: Data = Modbus address + number of elements to write + data to write
→ Response: Data = Request data
- If an error related to the requested Modbus function occurs, an Exception Response is sent from the server, with an exception function code:
→ Exception Function code = Request Function Code + 0x80, Data = Exception Code

1.3 Modbus Media

Modbus defines 4 media types :

- Holding Registers (HR): 16 bits - read / write
- Input Registers (IR): 16 bits - read only
- Coils: 1 bit - read / write
- Discrete Inputs (DI): 1 bit - read only

1

In each device, these Modbus media correspond to device specific media. Each device can have its own data organization; therefore it is necessary for the user to look at the device documentation before using Modbus. Two examples of data organization are shown below.

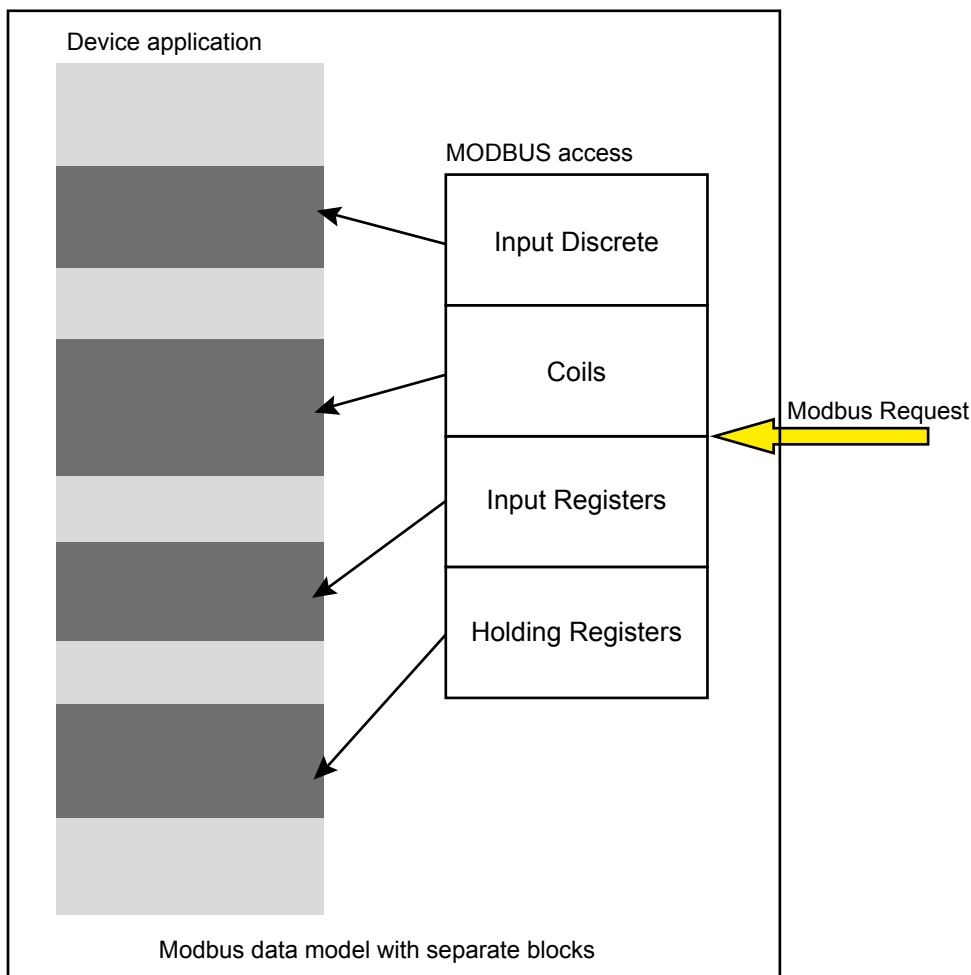


Figure 5: Modbus data model with separate blocks

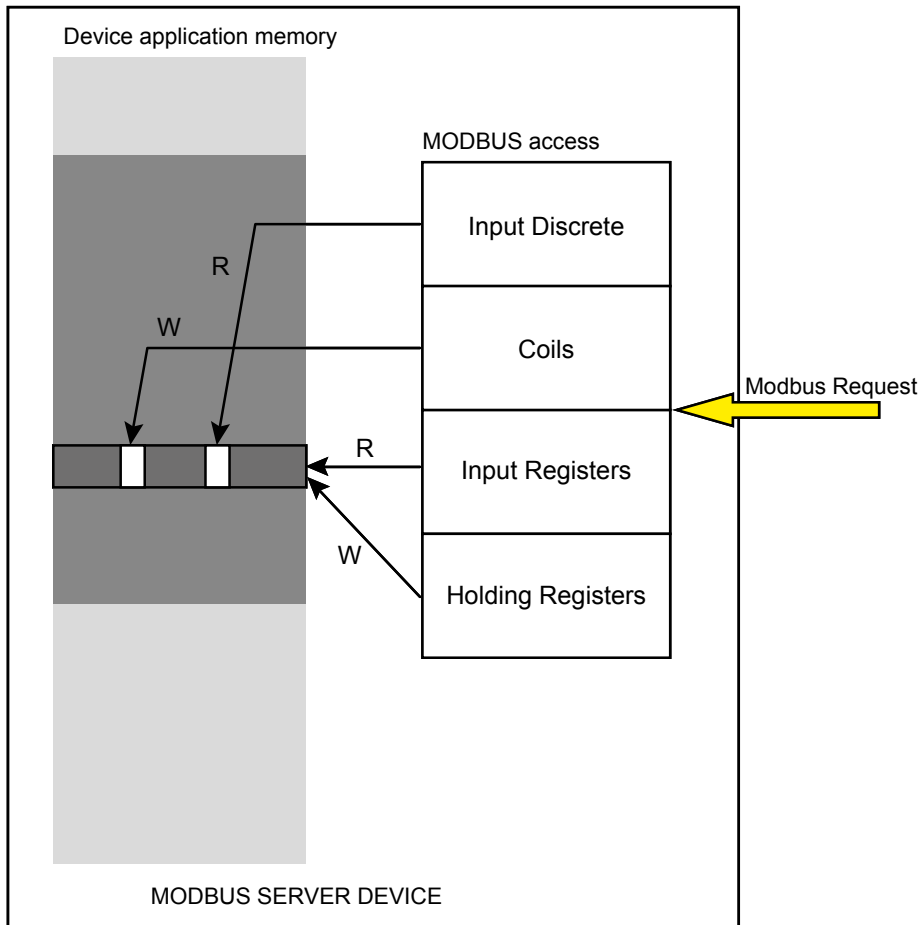


Figure 6: Modbus data model with only 1 block

In a Modbus frame, each data can be addressed from 0 to 65535, whereas the Modbus data model addresses each element within a data block from 1 to n. Then the Modbus data model has to be bound to the device application. The mapping between the Modbus data model and the device application is vendor specific.

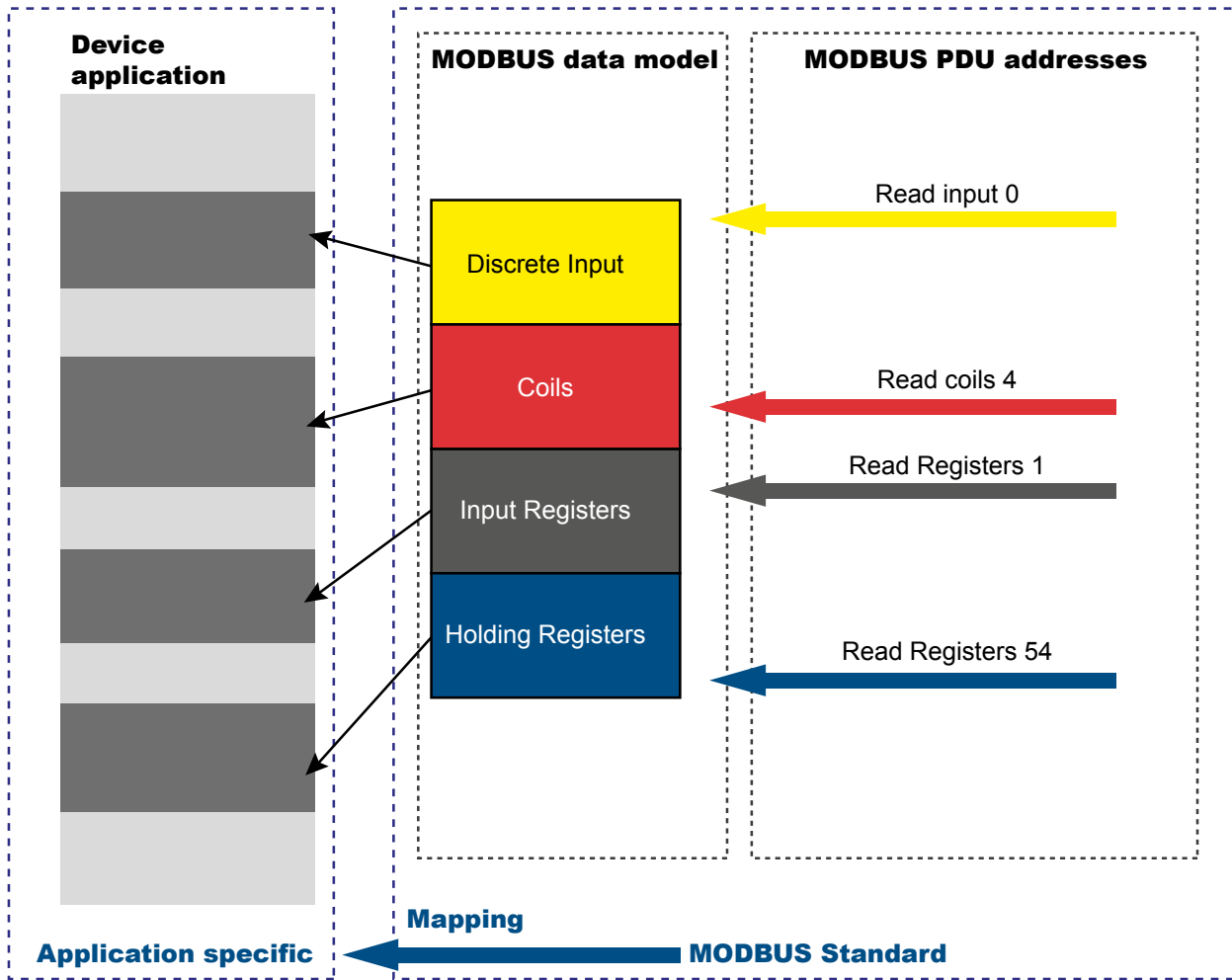


Figure 7: Modbus addressing model

More information about Modbus can be found at <http://www.modbus.org>.

2 SBC Modbus

This part describes the implementation of the Modbus Protocol on Saia PCD®. In this part, the terms Client and Server will be used, either for TCP / UDP or for RTU / ASCII in place of Master and Slave.

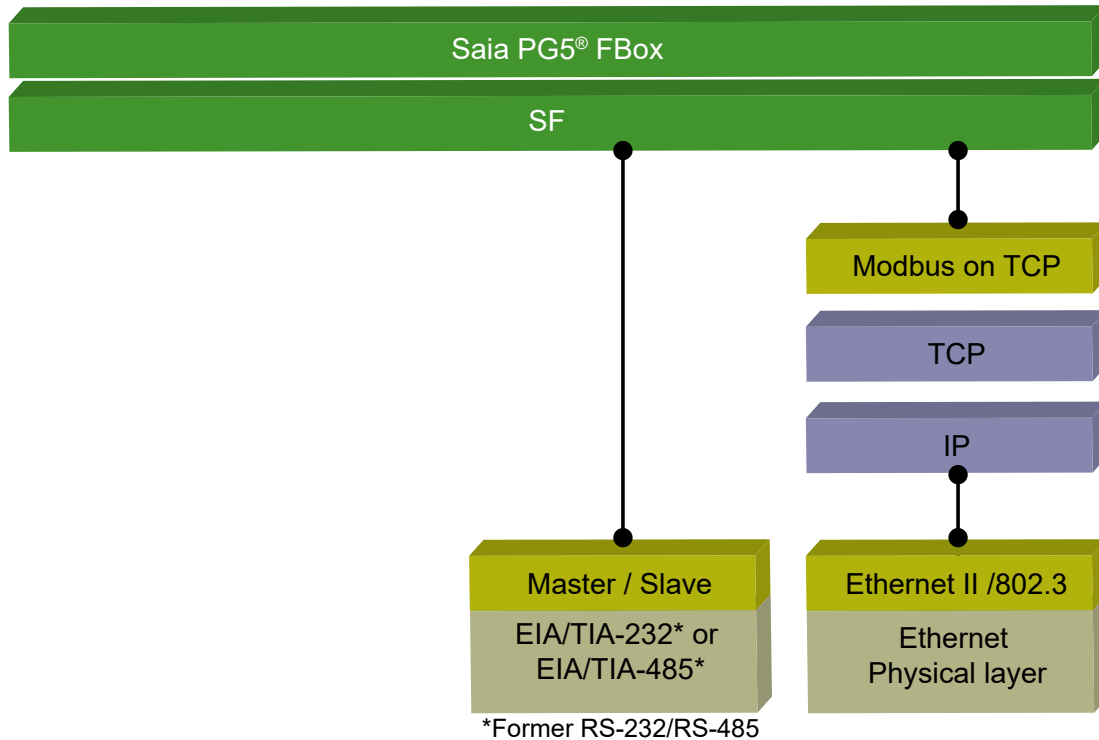


Figure 8: SBC Modbus communication stack

The SBC implementation of Modbus enables the user to configure and use Modbus clients and servers. This can be done using a dedicated interface which consists of Saia PG5® FBoxes and CSFs.

2.1 SBC Modbus Features

- Client and server functionalities
- Modes: IP (TCP and UDP) and serial (RTU and ASCII)
- Serial Modbus: available on all the serial ports, from 1200 Bd to 115,2 kBd (For using Modbus on a PCD2.F2xxx or a PCD3.F2xx module the minimal Saia PCD® Firmware is 1.14.23, and the F2xx module must have Hardwareversion D or more recent.)

- 8 Function codes (see ch. 6.4):
 - Read Coils
 - Read Discrete Inputs
 - Read Holding Registers
 - Read Input Registers
 - Write Single Coil
 - Write Multiple Coils
 - Write Single Holding Register
 - Write Multiple Holding Registers
- Media mapping: user definable
- Data handlings: many available to ensure compatibility with non-SBC devices, to enable 32 bits access on registers, floating point format, ...
- Servers: maximum 4 on a Saia PCD® system (a server is defined by a unique {port / protocol} pair) – For example: 1 server TCP port 502, 1 server TCP port 503, 1 server UDP port 502 and 1 server RTU on serial port 2.
- Unit IDs (UID): maximum 10 on a Saia PCD® system: 10 = 9 + UID 255 (UID from 0 to 254 are authorized; UID 255 exists by default on server but it must be configured before being accessed).
- Mapping areas: maximum 10 per UID – If more than 10 mapping areas are necessary, an additional UID can be defined (example: define a UID for Coils/ Discrete Inputs and another for Holding/Input Registers).
- Channels: maximum 10 on a Saia PCD® system (a channel is defined by a unique {port / protocol} pair) – For example, 1 channel TCP to server port 502, 1 channel TCP to server port 503, 1 channel UDP to server port 502 and 1 channel RTU using port 1.
- Connections: 26 connections can be open simultaneously. From these 26, maximum 10 connections can be open on client side. The remaining connections can be used by server. Remark: a connection is defined by a unique triplet {Port –Protocol – IP Address} in TCP / UDP and only pair Port / Protocol) in serial mode). The connections are open at the first Modbus request. In TCP/UDP, the connections are closed automatically after a user defined time.

2.2 Diagnostic

Diagnostic flags and registers are used to report the state of the UIDs (UID Diagnostic) and the Channels (Channel Diagnostic). They are also used in serial mode to report errors that can occur on the serial line (Server Diagnostic and Channel Diagnostic).

In the diagnostic register, all the bits but “Retry count” and “Response timeout” are cumulative (that is the user has to clear them).

2.2.1 Client / Server Diagnostic Flags

Address	Name	Description
Fxxx	RBSY	Channel: not used UID: UID busy Server: not used
Fxxx + 1	RFUL	Channel: not used UID: request with valid UID has been received and processed Server: not used
Fxxx + 2	RDIA	Reception diagnostic (see Diagnostic register)
Fxxx + 3	TBSY	Channel: Channel busy (if channel is busy, no request can be sent on it) UID: not used Server: not used
Fxxx + 4	Reserved	-
Fxxx + 5	TDIA	Transmission diagnostic (see diagnostic register)
Fxxx+ 6	Reserved	-
Fxxx + 7	NEXE	Client: request has not been processed successfully UID: not used Server: not used

2.2.2 Diagnostic Register

	Bit	Designation	Description	Protocol	Used for
R E C E I V E D	0	Overrun Error	Overrun of the internal reception buffer	Serial	Channel/Server
	1	Parity Error	Error in the parity of the received character	Serial	Channel/Server
	2	Framing Error	Incorrect baudrate	Serial	Channel/Server
	3	Break Error	Break on the serial line	Serial	Channel/Server
	4	CRC Error	CRC incorrect	Serial	Channel/Server
	5	-	-		
	6	-	-		
	7	-	-		
	8	Length Error	Telegram with invalid length received	Serial	Channel/Server
	9	Media Access Error	Error while accessing Saia PCD® Media	All	Channel/UID
	10	-	-		
	11	Server Start Error	Server could not start	All	Server
	12	Range Error	Error in the address or number of Modbus elements to access (see below).	All	UID
	13	Value Error	A value in the telegram is invalid (Media type, ...)	All	UID
	14	Area access error	Area not accessible (UID busy, area access not allowed)	All	UID
T R A N S M I S S I O N	15	-	-		
	16	Retry count	Number of repetitions done	All	Channel
	17				
	18				
	19				
	20	Response exception	Response exception received	All	Channel
	21	Response Timeout	No response received after a time-out	All	Channel
	22	Response Error	Invalid response received	All	Channel
	23	-	-		
	24	Send error	Request could not be sent	All	Channel
	25	-	-		
	26	Client Start Error	Client could not be started	All	Channel
	27	-	-		
	28	Media Error	Error in Saia PCD® media parameter	All	Channel
	29	-	-		
	30	Status Error	Error in client status (internal error)	All	Channel
	31	Program Error	Error in user program (for ex.: calling a CSF while TBSY or RBSY set)	All	Channel, UID

A “Range Error” may have different causes:

- No corresponding mapping found: no mapping area corresponding to data start address and number of Modbus elements could be found in User defined mapping areas (and also in default mapping areas if DEFMAP = 1).
- Number of Saia PCD® media to access exceeds Saia PCD® range of the area: for example when trying to access 8 Saia PCD® registers at offset 2 in an area containing 8 Saia PCD® registers. Attention must be paid on the number of Saia PCD® registers needed when using holes.
- Odd number or offset of HR while accessing 32 bits registers: in a 32 bits area 1 Saia PCD® register (32 bits) corresponds to 2 modbus holding registers. It is therefore not allowed to access Saia PCD® registers at an odd offset or with an odd number of holding registers.

(see Figure 12: Mapping Holding Registers (16 bits) on Saia PCD® Registers (32 bits) in a 32 bits area)

A “Client Start Error” may have different causes:

- TCP/UDP: cannot establish connection with server
- RTU/ASCII: cannot start listening on the serial line (Port number invalid → internal error)

A “Media Error” can occur if number of Saia PCD® media to access exceeds (Parameter “Count”) or if an error occurs while accessing Saia PCD® media (Saia PCD® media invalid, address of Saia PCD® media invalid...).

3 SBC Modbus Server

On the Server side, the essential communication steps are:

1. Creating server instance
2. Processing received request
3. Sending response

3

The SBC Modbus Server can be accessed over TCP/UDP or over serial line (RTU/ASCII).

SaiaPCD®

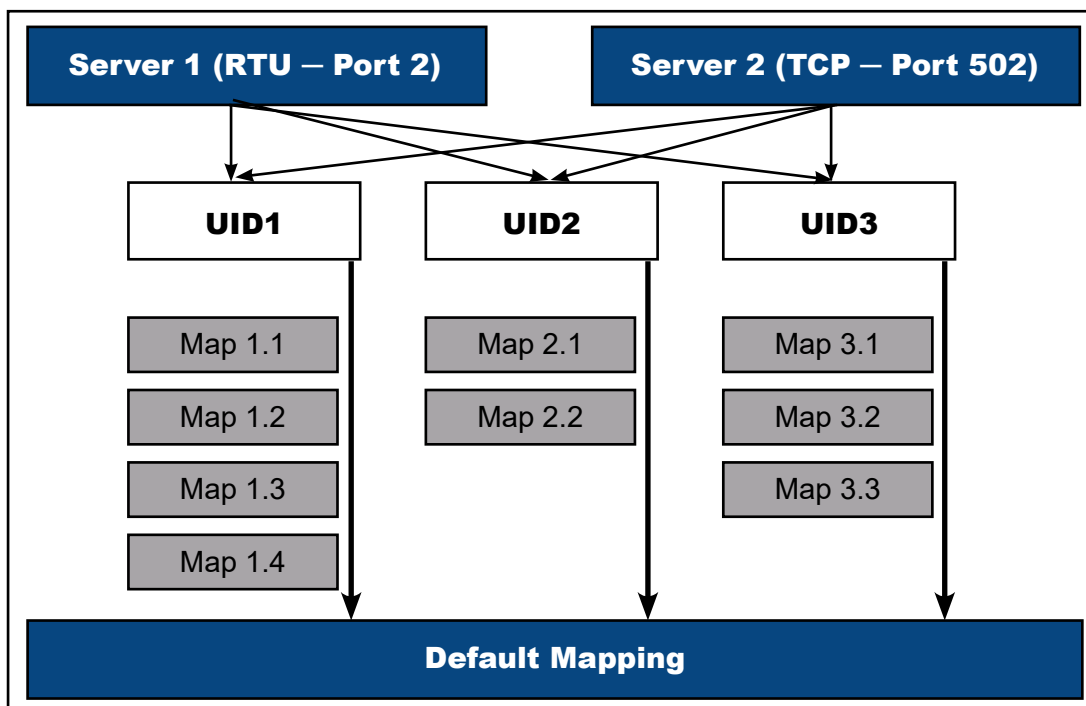


Figure 9: General Modbus architecture of a Saia PCD® server station

3.1 Server Instance

Many server instances can be defined on a SaiaPCD®, allowing multiple accesses over serial or IP.

A server is defined as a pair {Port – Protocol}. In serial mode it is not possible to use more than one server on one hardware interface (eg. RTU on port 1, ASCII on port 2, ...). For TCP / UDP, the port is logical port; several servers can be defined on the same port or the same protocol (TCP on port 502, UDP on port 502, TCP on port 503, ...). Moreover, it is not possible to define 2 identical server instances.

The servers are accessed via connections. In Serial and UDP, one connection per server instance is used. In TCP, one connection per connected client is needed.

Diagnostic flags and registers are available for each server (only used in serial mode).

3.2 Server UID

The SBC Modbus server is also characterized by one or many Unit Identifiers (UID). A UID is referred by all the server instances of the station. A UID is also associated with a specific media mapping and a specific data processing. It is possible to configure different UIDs on a Saia PCD®. A UID can be seen as a station number. The Modbus server processes all the requests addressed to one of its UIDs.

In serial mode, the server answers only to its defined UIDs. In case of broadcast UID (UID 0), the server processes the request for all defined UIDs. It is not allowed to define 2 times the same UID on a serial bus (except 0).

In TCP/UDP, the server answers to all the requests, either with an exception code or with a successful response. There is no broadcast UID.

Through the multiple UIDs, it is possible to use different media mapping configurations with different processing types, or to extend the existing default mapping.

Diagnostic flags and registers are available for each UID.

3.3 Media Mapping

Modbus uses 4 media types: Coils, Discrete Inputs, Holding Registers, Input Registers.

In order to make Saia PCD® media accessible on a Modbus server, these Modbus media have to be mapped on Saia PCD® Media; the mapping areas and the UIDs enable the user to configure the mapping of Modbus Media on Saia PCD® Media.

At least one UID has to be defined.

Coils and Discrete Inputs can be mapped only on Flags or I/O without any processing.

Holding Registers and Input Registers can only be mapped on Timers/Counters/Registers/DB but attention must be paid to data processing. In addition, on Client side, registers can be read from server and stored into a text or a text can be read and written into server registers. This is done using the function codes concerning holding or input registers.

Different processing types can be used on client or server side to be compatible with non-SBC devices: 16bits/32bits, 16bits signed / 16 bits unsigned, 32 bits FFP / 32 bits IEEE-754, offset and, for 32 bits registers, swap and holes.

	Modbus Media			Saia PCD® Media			
Access Type	Media Type	Start Address	Range	Media Type	Start Address	Range	Area Type

Figure 10: Definition of a mapping area

The **area type** defines the processing that will be performed on the received data:

- Coils, no special processing
- 16 bits signed (only write requests): if HR / IR value < 0, set MSW of Saia PCD® register to 0xFFFF
Example: HR = 0x8001 pro Saia PCD® register = 0xFFFF 8001
- 16 bits unsigned → MSW of Saia PCD® register set to 0x0000
Example: HR = 0x8001 pro Saia PCD® register = 0x0000 8001
- 32 bits: 2 HR / IR = 1 Saia PCD® register, no processing
- 32 bits IEEE: FFP to IEEE conversion performed before sending data on the bus (only used if user has FFP values, else use 32 bits), IEEE to FFP conversion performed before writing data into Saia PCD® media (only used if user wants the value in FFP, else use 32 bits).

3

The following schema shows how the data are processed between Saia PCD® and Modbus Frame.

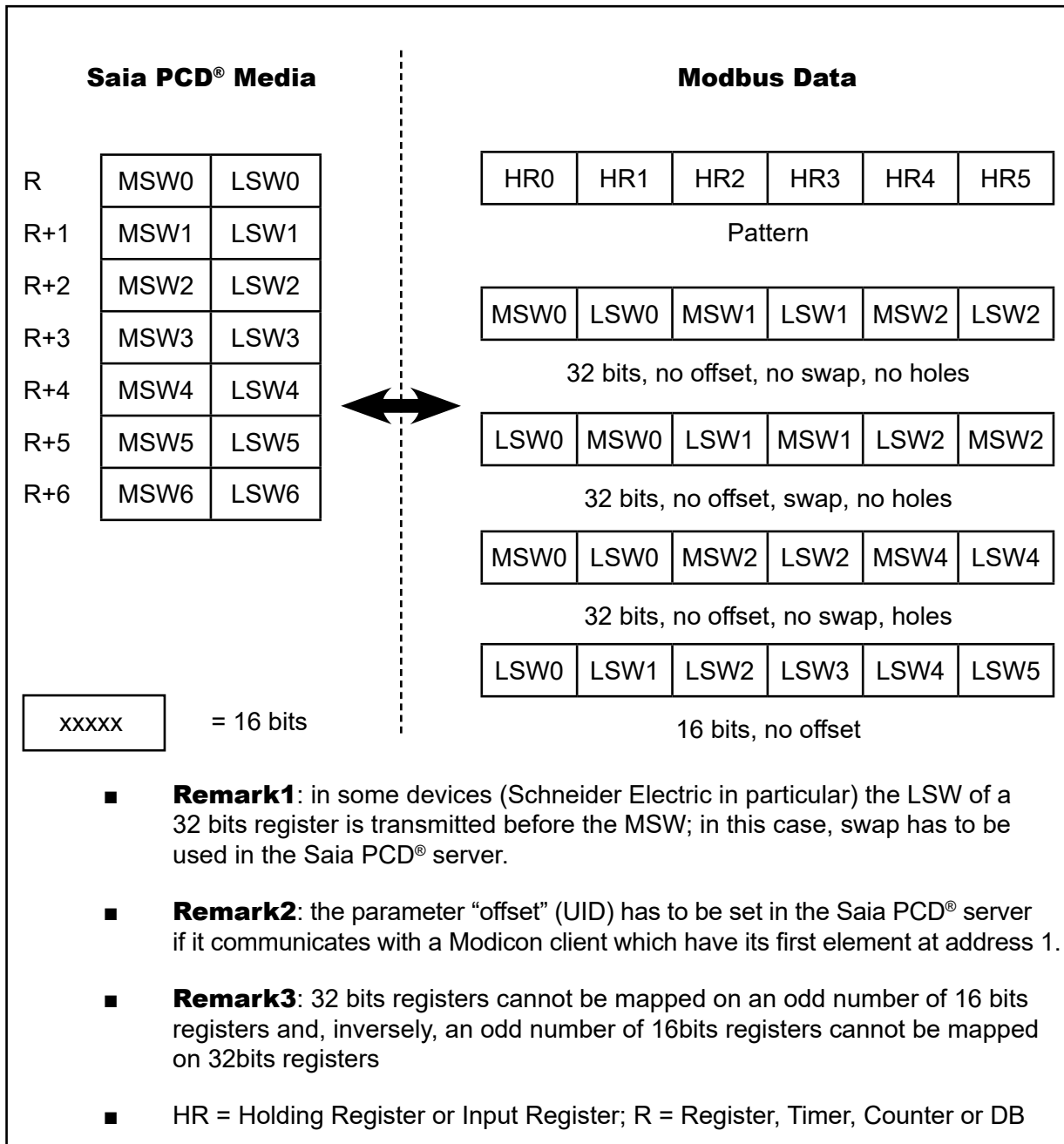


Figure 11: different processing types to map Modbus Holding Registers on Saia PCD® Registers (assuming that Modbus HR at address 0 is mapped on Saia PCD® register 0)

3.4 Default Mapping

Access Type	Modbus Request			Saia PCD®			AreaType
	Media Type	Start Addr.	Range	Media Type	Start Addr.	Range	
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

3

4 SBC Modbus Client

The Modbus communication is initialized by the Modbus Client. On Saia PCD®, this is done using Modbus Channels.

On Client side, the essential communication steps are:

1. Creating client instance
2. Establishing connection with server (TCP)
3. Sending request
4. Processing response

4

The SBC Modbus Client enables the user to send Modbus Read and Write requests over TCP/UDP or serial line (RTU/ASCII).



It is not allowed to use 2 Modbus clients on the same serial network. The correct behaviour of the network in case 2 clients communicate simultaneously on the bus cannot be guaranteed.

4.1 Client Channel

Many Modbus channels can be defined on a Saia PCD®, allowing communication over serial or IP to different Modbus servers.

A channel is defined as a pair {Port – Protocol}. The port is either the local port number in Serial mode or the remote server port in TCP / UDP. For example: {RTU – Port 2}, {TCP – Port 502}.

Diagnostic flags and registers are available for each channel. Moreover, it is possible to define 2 timeouts: a response timeout which represents the time within a response from the server is expected, and a close timeout, which represents the time after which a connection has to be closed if no activity occurs.

The channel is busy from request sending to response reception (or response timeout). In this time slice, it is not possible to use the channel for another request.

Diagnostic flags and registers are available for each channel.

4.2 Modbus Requests

A Modbus request is sent over a specified channel and is addressed to a specific UID (and an IP address for TCP/UDP). It is also possible to set the data processing that has to be applied to sent or received data.

A response from the server is expected within a specified time slice. If no answer comes within this time slice, the server does not exist (or the UID is not defined in serial mode). In case of broadcast UID (UID 0) over serial line, no answer is expected from the client. It has just to be waited for the defined time before sending the next request.

When sending the request, it has to be defined which processing will be performed on data (before sending them or after receiving them):

- Coils_DI_Processing, no special processing
- 16 bits signed (only read requests): if HR / IR value < 0, set MSW of Saia PCD® register to 0xFFFF

Example: HR = 0x8001 pro Saia PCD® register = 0xFFFF 8000

- 16 bits unsigned pro MSW of Saia PCD® register set to 0x0000

Ex: HR = 0x8001 pro Saia PCD® register = 0x0000 8001

- 32 bits binary pro 2 HR / IR = 1 Saia PCD® register, no processing
- 32 bits swap pro 2 HR / IR wordwise swapped = 1 Saia PCD® register

Ex: HR0 = 0x1234, HR1 = 0x5678

Saia PCD® register = 0x1234 5678 (32 bits binary)

Saia PCD® register = 0x5678 1234 (32 bits swap)

- 32 bits IEEE2FFP - 32 bits FFP2IEEE

FFP to IEEE conversion performed before sending data on the bus
(only used if user has FFP values, else use 32 bits)

IEEE to FFP conversion performed before writing data into Saia PCD® media
(only used if user wants the value in FFP, else use 32 bits)

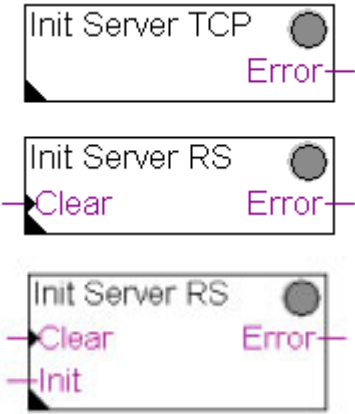
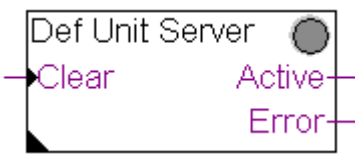

- Swap + 32 bits IEEE2FFP / 32 bits FFP2IEEE : wordwise swap performed just before sending data / after receiving data

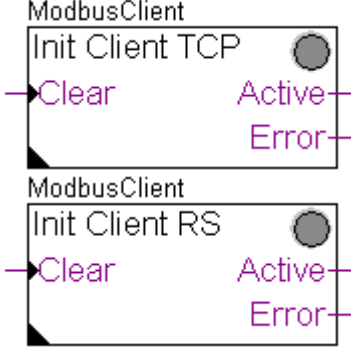
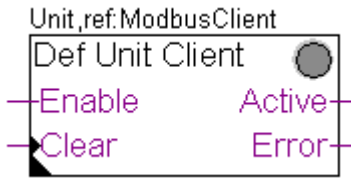
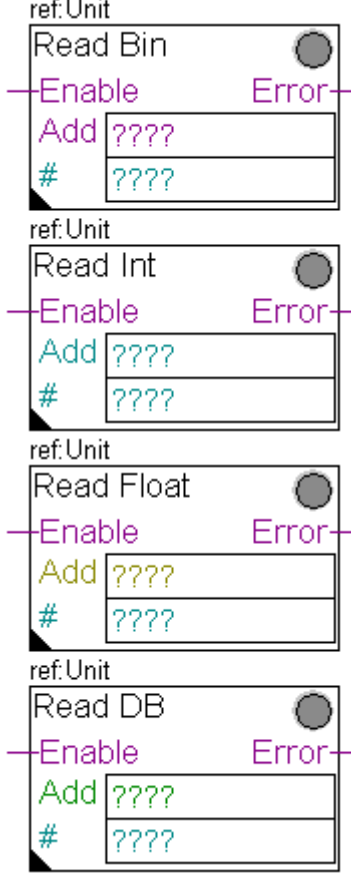
5 Modbus FBox Library

The Modbus FBox Library allows using the Modbus protocol on a Saia PCD® by specific Saia PG5® FBoxes.

The library includes the following functionalities:

- Server initialization
- Defining UIDs (Server)
- Defining mappings (Server)
- Defining Channels (Client)
- Sending Read and Write Requests from a Client

	<p>Initializes the server instance:</p> <p>Port Number (+ port parameters for serial) Mode: TCP, UDP, RTU, ASCII</p> <p>Another Extended InitServerRS FBox is available to allow destroying a serial server instance and closing the related serial port.</p>
	<p>UID definition with specific parameters:</p> <p>Offset, Holes, Mapping, Swap</p>
	<p>Defines mapping areas:</p> <p>UID concerned by the mapping</p> <p>Modbus media and addresses mapped</p> <p>Saia PCD® media and addresses corresponding to Modbus data</p> <p>Area type</p>

	<p>Initializes the channel instance:</p> <ul style="list-style-type: none"> Channel number Port Number (+ port parameters for serial) Mode: TCP, UDP, RTU, ASCII Close timeout Response timeout Retries <p>Another Extended InitClientRS FBox is available to allow destroying a serial channel and closing the related serial port.</p>
	<p>Define the server, the client communicates with:</p> <ul style="list-style-type: none"> IP address for TCP / UDP UID
	<p>Sends a read request (Read Coils, Discrete Inputs, Holding Registers or Input Registers), process the data received and store them in specified Saia PCD® media. There are also additional FBoxes for Indirect Addressing of the Saia PCD® media.</p>

<p>ref.Unit Write Bin <input type="radio"/> <input type="radio"/></p> <p>Enable Error</p> <p>Add [????]</p> <p># [????]</p> <p>ref.Unit Write Int <input type="radio"/> <input type="radio"/></p> <p>Enable Error</p> <p>Add [????]</p> <p># [????]</p> <p>ref.Unit Write Float <input type="radio"/> <input type="radio"/></p> <p>Enable Error</p> <p>Add [????]</p> <p># [????]</p> <p>ref.Unit Write DB <input type="radio"/> <input type="radio"/></p> <p>Enable Error</p> <p>Add [????]</p> <p># [????]</p>	<p>Read the specified Saia PCD® media, process the data and send them as a write request (write Holding registers or Coils). There are also additional FBoxes for Indirect Addressing of the Saia PCD® media.</p>
--	---

For details refer to the online help of the Saia PG5® FBoxes.

If the CSFs are dragged from the Function Selector there is no need to include a file manually with the \$INCLUDE; this is done automatically in the background.

Please note that the names of the keywords have been changed to S.SF.Modbus in PG5 2.0.

6 Modbus System Functions Library

6.1 Introduction

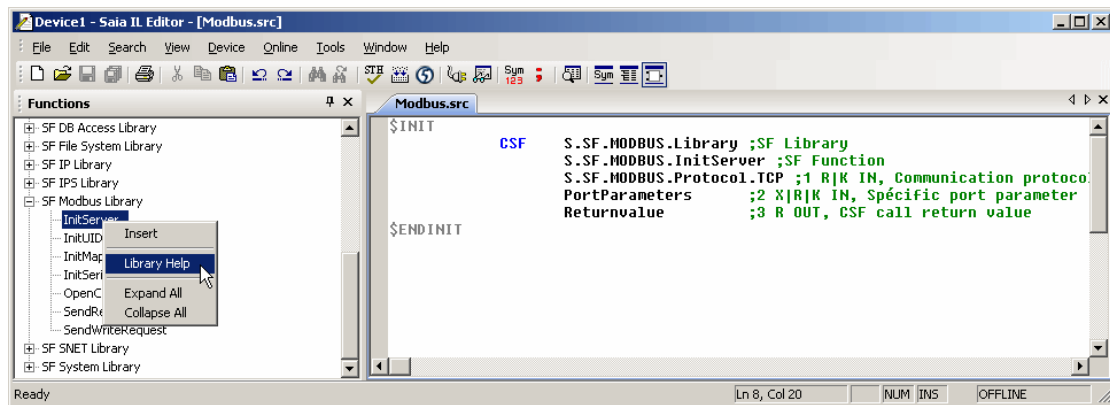
The system functions in a user program are accessed by the CSF (Call System Function) command. The general form of the CSF call is:

```
CSF [cc]   lib_number ;library number 0-4095/4096-8191
           func_number ;function number to call
           [parms...] ;optional parameters
```

PG5 2.0

In PG5 2.0 the CFSs can be added to the IL File by drag-and-dropping them from the Function Selector.

6



PG5. 1.4

The “lib_number” and “function_number” are required to execute CSF commands. Keywords can be used by including the “Modbus.inc” file in PG5. 1.4.

```
$INCLUDE "Modbus.inc"
```

```
COB 0
```

```
0 ...
```

Modbus library calls are accessed through the library keyword:

```
S.Modbus.Library
```

The functions available are explained in the following chapters, but please use “S.Modbus” instead of S.SF.Modbus in PG5 1.4..

6.2 Parameters

Name	Used	Description
AccessType	InitMap	Defines the way the mapping area can be accessed (Read/Write).
AreaType	InitMap	Defines the default processing to be performed on data of this area before Saia PCD® media access (Write) or after Saia PCD® media access (Read).
ChannelID	OpenChannel SendReadRequest SendWriteRequest	A client station can implement one or many logical channels. The ChannelID is used for creating or modifying a channel, or for sending a request to a remote partner. Range authorized: 1-10
CloseTimeout	OpenChannel	Timeout to close a connection after an idle time (in seconds) – Only used for Modbus TCP.
Count	SendReadRequest SendWriteRequest	Number of elements to read/write (NOT modbus elements: if request to read one 32 bits register on a Saia PCD®, Count = 1) Max number: - Read Coils / DI: 2000 - Write coils: 1968 - Read HR / IR: 125 - Write HR: 123 in case of 16 bits request (Respectively 62 and 61 in case of 32 bits access)
Diag	OpenChannel InitUID	This parameter defines the location of the diagnostic Flags and Register. Format: "DIAG:Fxxx,Ryyy" Fxxx = base address of 8 consecutive flags Ryyy = address of diagnostic register It is recommended to clear all the diagnostic media used at the start of the program (XOB16).
ExceptCode	SendReadRequest SendWriteRequest	Location where to store the exception code returned in the partner response.
Mode	InitUID	Defines operating mode: 0 = auto: the requests are processed and answered automatically 1 = transparent: the requests are processed and answered by the user application (not implemented yet)
PartnerRange	InitMap	Number of Modbus elements belonging to the mapping area.
PartnerStart	SendReadRequest SendWriteRequest InitMap	First Modbus address of the mapping area (InitMap) or first modbus address of the data to read/write (SendRequest). Pay attention to the parameter holes (UID) to access the right register.
PartnerType	InitMap	This parameter defines the type of Modbus data accessed by the client.

Saia PCD® Media	SendReadRequest SendWriteRequest	Read Request: defines at which which location on the Saia PCD® the requested data will be copied. Write Request: Defines from which location in the Saia PCD® the data to send will be taken
Saia PCD® MediaType	InitMap	Saia PCD® media type on which the Modbus data with type "PartnerType" will be mapped.
Saia PCD® Range	InitMap	Number of Saia PCD® elements in the mapping area. In case of DB, number of elements of this DB in the area; address of 1st element is 0.
Saia PCD® Start	InitMap	Saia PCD® Start Address corresponding to Partner Start. In case of DB, number of the DB concerned by the area.
Port	OpenChannel	TCP/UDP: Logical port number the request will be sent to RTU/ASCII: number of the serial port used for sending
PortParam	InitServer InitSerialPort	TCP/UDP: number of the logical port used by the server RTU/ASCII: text containing the serial port parameters. Format: « PORT :portnumber; UART :baudrate,data,parity,stopbits; LINE :linetype» - PortNumber is the number of the serial port - Baudrate is the baudrate used on this serial port - Data, Parity, Stopbits ; Data must be 8 to comply with the Modbus norm in RTU and 7 in ASCII. Parity and stop bits must also be set so that the total number of bits in a character is 11 in RTU and 10 in ASCII (that is N,2 or E,1 or O,1) - LineType can be: - SL0: RS-232 - SL1: RS-422 - SL2: RS-232, no handshake - SL3: RS-422, no handshake - SL4: RS-485, auto RTS For InitServer: this text has to be enhanced with " DIAG :Fxxx,Ryyy" to indicate where to store the diagnostics related to the server instance. It is recommended to clear the diagnostic flags and registers at the start of the program. To close a Port: Format: « PORT :portnumber; OFF »

Process	InitUID	<p>Defines the processing to be performed on data</p> <p>Format: "OFFSET=0, SWAP=1, DEFMAP=1, HOLES=0"</p> <p>OFFSET =1 for communication with a Modicon client (first element at address 1); OFFSET =0 for other devices with first register at address 0 (Saia PCD®, ...).</p> <p>SWAP=1 if a wordwise swap has to be performed on 32bits data (cf. Figure 11)</p> <p>DEFMAP: defines in which mapping table the modbus addresses will be searched</p> <p>DEFMAP=0: only user defined mapping has to be used.</p> <p>DEFMAP=1: first search in user defined mapping. If not found, search in default mapping (see DefaultMapping).</p> <p>HOLES: defines how the Saia PCD® registers are mapped to HR/IR.</p> <p>HOLES=0: for 32 bits access, HR0 request will update HR0 and HR1 with content of Reg0, HR2 request will update HR2 and HR3 with content of Reg1.</p> <p>HOLES=1: for 32 bits access, HR0 request will update HR0 and HR1 with content of Reg0, HR2 request will update HR2 and HR3 with content of Reg2.</p> <p>(cf. Figure 11)</p>
Processing	SendReadRequest SendWriteRequest	<p>Read Request: defines the processing to be performed on received data before being written in Saia PCD® media</p> <p>Write Request: defines the processing to be performed on data before being copied into the request</p>
Protocol	InitServer OpenChannel	Protocol used for modbus communication.
RemoteIPAddr	SendReadRequest SendWriteRequest	IP address of the server to send a request to (TCP/UDP)
ReqType	SendReadRequest SendWriteRequest	Modbus function code. This parameter defines the type of the request (Read/Write, Coils/Holding Registers, ...).
RespTimeout	OpenChannel	Timeout within which a response shall be received (in milliseconds)
Retries	OpenChannel	Number of retries in case of response timeout (1st try included)

<p>Status</p>	<p>InitMap InitServer InitUID OpenChannel SendReadRequest SendWriteRequest</p>	<p>State after the CSF has been executed.</p> <p>If smaller than 0 an error occurred during the call. Refer to the error code list</p> <p>When the function has been successfully executed, status is ≥ 0</p>
<p>UID</p>	<p>SendReadRequest SendWriteRequest InitUID InitMap</p>	<p>The Unit Identifier (UID) is inserted into the request. When a request is received, the UID is used to find out which Mode has to be used and which processing will be performed on data. More than one UID can be defined on a server station. The UIDs are referred by all server instances on the station. It is recommended to use only one UID on a station in serial mode.</p> <p>In Serial mode, if a request with UID 0 (broadcast UID) is received, all the defined UIDs react to this request with their own processing and mapping.</p> <p>In TCP/UDP, the UID 0 and the UID 255 correspond to the same UID. To access this UID on a TCP/UDP server, the user has to configure its parameters. It can be done using the InitUID CSF.</p> <p>It is recommended to avoid accessing a TCP/UDP server with UID 0.</p>

6.3 Error codes

In case of successful processing, all CSF return either 0 (zero) or a positive value. A negative value indicates an error. Below is a list of the error codes:

6.3.1 CSF error codes

Most of these errors can appear when one or more CSF parameters are not correct.

Code	Designation	Description
0	NO_ERROR	No error
-4200	INVALID_MEDIATYPE	PCD media type is invalid (Send Request)
-4199	DIAGTEXT_ERROR	Cannot read the diagnostic text (Channel / UID) (internal)
-4198	INVALID_DIAG_SYNTAX	Syntax error in diagnostic text (Channel / UID)
-4197	INVALID_PROCESSING_PARAM	At least 1 parameter in the processing text is invalid (UID)
-4196	INVALID_PROCESSING_SYNTAX	Syntax error in processing text (UID)
-4195	PROCESSINGTEXT_ERROR	Cannot read the processing text (UID) (internal)
-4194	CHANNEL_NOT_DEFINED	Channel is not defined (Send Request)
-4193	CHANNEL_BUSY	Channel is already used (Send Request)
-4192	SERVER_NOT_CREATED	Server cannot be created (Init Server): max number of servers already reached or server already defined.
-4191	SERVER_NOT_STARTED	Server cannot start (Init Server)
-4190	INVALID_FUNCTION_CODE	Function code does not match request type (Read / Write)
-4189	INVALID_PROCESSING	Processing does not match function code or request type (Read / Write)
-4188	INVALID_PROTOCOL	Protocol is invalid (Server / Channel)
-4187	INVALID_PARTNER_TYPE	Modbus Media is not a valid type(Init Map)
-4186	INVALID_PCDMEDIA_TYPE	PCD Media is not a valid type(Init Map)
-4185	INVALID_AREA_TYPE	Area type is not a valid type (Init Map)
-4184	INVALID_PCD_MEDIA_ADDR	PCD Media address is invalid (Init Map)
-4183	INVALID_RANGE	Invalid media range (Init Map): Partner range < PCD range or range = 0.
-4182	INVALID_AREA_ACCESS_TYPE	Access type is invalid (Init Map): does not exist or does not match Modbus media.
-4181	INVALID_NB_OF_MEDIA	Number of media does not match function code (Send Request): > 1 for write single coil or single register.
-4180	INVALID_UID	UID out of the range 0-0xFF (Init UID, Send Request) or UID = 0 for a read request in serial mode
-4179	INVALID_CHANNEL	Channel out of the range 1-10 (Open Channel, Send Request)
-4178	INCOMPATIBLE_PCDMEDIA	PCD media is incompatible with Modbus media (Init Map) or Function code (Send Request)
-4177	INVALID_PORT_CONF	Port configuration (Text or number) does not match protocol (Init Server)
-4176	PORTCONFIGTEXT_ERROR	Cannot read port configuration text in serial mode (Init Server, Init SR Port) (internal)
-4175	PORTCONFIGPARAM_ERROR	Error in port configuration text in serial mode (Init Server, Init SR Port)
-4174	SRPORT_START_ERROR	SR port cannot be open (Init Server, Init SR Port). In serial, the port may be already used or port parameters are invalid.
-4173	INVALID_PORT_NUMBER	SR Port Number is invalid (Open Channel)
-4172	INVALID_RETRY_NB	Number of retries invalid (must be < 16)
-4171	NO_PORT_TO_CLOSE	The serial port to close has not been opened

6.3.2 ModbusShell error codes

These errors appear when processing CSF.

Code	Designation	Description
-4100	UID_TABLE_FULL	Maximum number of UIDs is already reached (Init UID)
-4099	UID_NOT_DEFINED	UID is not defined (Init Map)
-4098	UID_BUSY	UID is being used (Init UID), cannot modify it
-4097	CHANNEL_TABLE_FULL	Table of channels is full; cannot add the channel
-4096	CHANNEL_ALREADY_DEFINED	A channel with same pair Port / Protocol is already defined (Open Channel)
-4095	CHANNEL_BUSY	This channel is being used, cannot modify it
-4094	MAP_TABLE_FULL	Maximum number of mappings reached for the UID (Init Map)
-4093	MAP_AREA_OVERLAP	At least 2 map areas overlap (Init Map)
-4092	CONNECTION_TABLE_FULL	Maximum number of "connections" is reached (Send Request). TCP/UDP: connection = {Port-Protocol-IPAddress} Serial: connection = {Port-Protocol}
-4091	CONNECTION_NOT_READY	Connection is not ready for sending (Send Request): IP address not set, ...

6.3.3 ModbusDriver error codes

Most of the errors between -3980 and -4000 are internal. If an error not listed below occurs, please contact Saia Burgess Controls.

Code	Designation	Description
-3990	CLII_DRIVER_NOT_READY	Driver is not ready (Send Request)
-3987	QUANTITY_ERROR	Error in the quantity of media to read / write (Send Request): exceeds limit or < 0
-3983	CLIENT_NO_RESSOURCE	No resource available (Send Request)
-3982	PORT_ID_ERROR	Port already initialized or used
-3981	INVALID_PORT_STATE	Port not ready (Serial) (Send Request)
-3980	UNKNOWN_ERROR	Unknown error

6.4 Function Codes

PG5 2.0	PG5 1.4	EQU
S.SF.Modbus.FunctionCode.	S.Modbus.FunctionCode.	[hex]
ReadCoils	ReadCoils	0x01
ReadDiscreteInput	ReadDiscreteInput	0x02
ReadHoldingReg	ReadHoldingReg	0x03
ReadInputReg	ReadInputReg	0x04
WriteSingleCoil	WriteSingleCoil	0x05
WriteSingleReg	WriteSingleReg	0x06
WriteMultipleCoils	WriteMultipleCoils	0x0F
WriteMultipleRegs	WriteMultipleRegs	0x10

6.5 Protocols

PG5 2.0	PG5 1.4	EQU
S.SF.Modbus.Protocol.	S.Modbus.Protocol.	
TCP	TCP	0
UDP	UDP	1
RTU	RTU	2
ASCII	ASCII	3

6.6 Modbus Data Types

PG5 2.0	PG5 1.4	EQU
S.SF.Modbus.MBMedia.	S.Modbus.MBMedia.	
NoMedia	NoMedia	0
Coils	Coils	1
DiscrInput	DiscrInput	2
HoldingReg	HoldingReg	3
InputReg	InputReg	4

6.7 Modbus Area Types (Server)

PG5 2.0	PG5 1.4	EQU
S.SF.Modbus.AreaType.	S.Modbus.AreaType.	
Coils	Coils	0
Reg_16bits_signed	Reg_16bits_signed	1
Reg_16bits_unsigned	Reg_16bits_unsigned	2
Reg_32bits	Reg_32bits	3
Reg_32bits_IEEE	Reg_32bits_IEEE	4

6.8 Modbus Area Access Types (Server)

PG5 2.0	PG5 1.4	EQU
S.SF.Modbus.AreaAccess.	S.Modbus.AreaAccess.	
ReadWrite	ReadWrite	0
ReadOnly	ReadOnly	1
WriteOnly	WriteOnly	2
NoAccess	NoAccess	3

6.9 Processing Types (Client)

PG5 2.0	PG5 1.4	EQU
S.SF.Modbus.ClientProcessing.	S.Modbus.ClientProcessing.	
Coils_DI_Processing	Coils_DI_Processing	0
R_16bits_signed	R_16bits_signed	1
RW_16bits_unsigned	RW_16bits_unsigned	2
RW_32bits_binary	RW_32bits_binary	3
RW_32bits_swap	RW_32bits_swap	4
R_32bits_IEEE2FFP	R_32bits_IEEE2FFP	5
R_32bits_swap_IEEE2FFP	R_32bits_swap_IEEE2FFP	6
W_32bits_FFP2IEEE	W_32bits_FFP2IEEE	7
W_32bits_swap_FFP2IEEE	W_32bits_swap_FFP2IEEE	8

6.10 Saia PCD® Media Types

PG5 2.0	PG5 1.4	EQU
S.SF.Modbus.PCDMedia.	S.Modbus.PCDMedia.	
NoMedia	NoMedia	0
IO	IO	1
Input	Input	2
Output	Output	3
Flag	Flag	4
TC	TC	5
Reg	Reg	6
DB_Media	DB_Media	9

6.11 Exception Codes

Codes replied by a Server in case the request cannot be executed.

ILLEGAL_FUNCTION_CODE	1	Function code not valid or does not correspond to Modbus media type
ILLEGAL_DATA_ADDRESS	2	Modbus media at received address cannot be accessed (check UID mapping/ see diagnostic)
ILLEGAL_DATA_VALUE	3	Number of data in the request is invalid
SERVER_FAILURE	4	PCD Media could not be accessed (see diagnostic) or UID not configured (in particular UID 255 in TCP/UDP)
ACKNOWLEDGE	5	
SERVER_BUSY	6	Server is already busy when trying to access it
GATEWAY_PATH_NOT_FOUND	10	UID not defined.
GATEWAY_TARGET_PROBLEM	11	

7 Modbus CSF Specification


7.1 S.SF.Modbus.InitSerialPort (Client / Server)

S.SF.Modbus.InitSerialPort		<p>This function opens a serial port on client side. On server side, it is not needed since the port initialization is done when creating the server.</p> <p>For Client and Server, this function can also be used to close a serial port and all the related instances (Channels and Servers). For sending / receiving on the serial port that has been closed, the appropriate initialization functions have to be called (InitServer, InitChannel, InitSerialPort) again first.</p> <p>Multiple use of the same serial port is not allowed.</p>		
Parameters		8		
1	PortParam	X	IN	Parameters of the serial port to open
2	Status	R	OUT	Returned Value (see below)
Returned Value				
	Success	= 0	Channel successfully created or modified	
	Error	< 0	see chapter 6.3	

7

Example	Initialization of a serial port
CSF	<pre>S.SF.Modbus.Library S.SF.Modbus.InitSerialPort ConfigPort2 ; Text: "PORT:2;ART:9600,8,N,2;LINE:SL4" R_StatusOpenPort ; R 10101</pre>
Example	Closing of a serial port
CSF	<pre>S.SF.Modbus.Library S.SF.Modbus.InitSerialPort ConfigPort2 ;Text: "PORT:2; OFF" R_StatusOpenPort ; R 10101</pre>

7.2 S.SF.Modbus.OpenChannel (Client)

S.SF.Modbus.OpenChannel		<p>This function creates a new channel or modifies an existing channel (if not currently busy). A channel is used to send request to partner and to process the received data. A channel can be used over any given interface (Serial / Ethernet).</p> <p>It is not allowed to open 2 channels with same pair Port/Protocol. When trying to open a channel with an already defined Channel ID, the “old” channel is overwritten (if not currently in use, see TBSY).</p> <p>It is not allowed to use 2 Modbus clients on the same serial network. The correct behaviour of the network in case 2 clients communicate simultaneously on the bus cannot be guaranteed.</p>		
				
Parameters		8		
1	ChannelID	R K	IN	Channel Identifier of the channel to create or modify. Range authorized: 1-10
2	Port	R K	IN	TCP/UDP: Remote Port the request will be sent to RTU/ASCII: number of the serial port used for sending
3	Protocol	R K	IN	Protocol used for communication on this channel
4	Diag	X	IN	Diagnostic Definition. “DIAG:Fxxxx,Ryyyy”
5	CloseTimeout	R K	IN	Timeout to close a TCP-connection after an idle time. Parameter to specify the time in seconds after which the TCP-connection is closed if no data has been transmitted. If the value is 0, the connection is closed after each request immediately (requires firmware 1.16.xx or later). This could be used if more than 10 Modbus TCP servers are to be connected, but it does increase the amount of generated Ethernet frames due to the connection establishments.
6	RespTimeout	R K	IN	Timeout within which a response shall be received – This value is also used as “broadcast timeout”
7	Retries	R K	IN	Number of retries allowed in case of response timeout on this channel
8	Status	R	OUT	Returned Value (see below)
Returned Value				
	Success	= 0	Channel successfully created or modified	
	Error	< 0	see chapter 6.3	

Example		
CSF	S.SF.Modbus.Library	
	S.SF.Modbus.OpenChannel	
	2	; Channel ID
	502	; Remote Port
	S.SF.Modbus.Protocol.TCP	; Protocol
	Diagnostic	; «DIAG:F1000,R1000»
	2000	; CloseTimeout
	1000	; ResponseTimeout
	0	; Number of retries
	R_StatusOpenCh	; R 10105

7.3 S.SF.Modbus.SendReadRequest / S.SF.Modbus.SendWriteRequest (Client)

S.SF.Modbus.SendRead-Request		This function sends a Modbus Read request over a given channel.		
S.SF.Modbus.SendWriteRequest		This function sends a Modbus Write request over a given channel.		
		For Send Requests, the TBSY flag of the channel must be low to send the request; otherwise, the CSF returns an error.		
Parameters		12		
1	ChannelID	R K	IN	Channel Identifier (one of the already open channels): indicates on which channel the request will be sent
2	RemotelIPAddr	R	IN	IP address of server the request will be sent to (only used in TCP/UDP)
3	UID	R K	IN	Unit identifier: used to define which processing and which mode shall be used by server
4	ReqType	R K	IN	Function code
5	ExceptCode	R	OUT	Location where the exception code of the partner response will be copied
6	Saia PCD® Media	R TC IO F X DB	IN	Read Request: defines on which Saia PCD® media and at which location the requested data will be copied Write request: defines from which Saia PCD® media and at which location the requests will be taken
7	PartnerStart	R K	IN	Defines at/from which server media the data will be written/read (depends on server mapping)
8	Count	R K	IN	Number of elements to be read/written from/to partner (NOT Modbus elements: if 32 bits registers are requested, number of 32 bits registers)
9	Processing	R K	IN	Read Request: defines which processing will be performed on the received data before being written in Saia PCD® media. Write Request: defines the processing to perform on data before copying them into the request.
10	Status	R	OUT	Returned Value (see below)
Returned Value				
	Success	= 0		Request successfully sent
	Error	< 0		see chapter 6.3

Example		
CSF	S.SF.Modbus.Library	

S.SF.Modbus.SendReadRequest/S.SF.Modbus.SendWriteRequest(Client)

```
S.SF.Modbus.SendReadRequest      ; Function = Send Read Request
2                                ; ChannelID
RegisterContainingIP             ; IP address = 172.23.2.129 = 0AC170281H
1                                ; Remote UnitID
S.SF.Modbus.FunctionCode.ReadCoils
R 100                            ; Location to store Exception Code
F 0                               ; PCD Local Media
10000                            ; PartnerStartAddress
10                               ; Number of Media
S.SF.Modbus.ClientProcessing.Coils_DI_Processing
R 101                            ; Location to store CSF Status
```

7.4 S.SF.Modbus.InitServer (Server)

S.SF.Modbus.InitServer		This function allows creating a server instance. Multiple use of server with same pair Port/Protocol is not allowed in TCP/UDP (only same Port in Serial)		
Parameters		3		
1	Protocol	R K	IN	Communication protocol used by the server instance
2	PortParam	R K X	IN	TCP/UDP: number of the logical communication port used by the server instance RTU/ASCII: Text containing the serial port parameters
3	Status	R	OUT	Returned Value (see below)
Returned Value				
	Success	= 0		Server successfully started
	Error	< 0		see chapter 6.3

7

Example (TCP/UDP)		
CSF	<pre>S.SF.Modbus.Library S.SF.Modbus.InitServer S.SF.Modbus.Protocol.UDP 502 ;Port R 102 ;Location to store CSF status</pre>	

Example (SERIAL)		
CSF	<pre>S.SF.Modbus.Library S.SF.Modbus.InitServer S.SF.Modbus.Protocol.RTU ConfigPort1 ;Port configuration: ;PORT:1; UART:9600,8,N,2;LINE:SL4 ;DIAG:F800,R800" R 102 ;Location to store CSF status</pre>	

7.5 S.SF.Modbus.InitUID (Server)

S.SF.Modbus.InitUID		<p>This function creates or modifies a unit identifier which will be referred by all the server instances. When a request is received, the UID is used to find out which Mode has to be used and which processing will be performed on data. More than one UID can be defined on a server station.</p> <p>When trying to create a UID with an already defined UID number, the “old” one is overwritten (if not currently in use, see RBSY).</p>		
Parameters		5		
1	UID	R K	IN	Unit identifier
2	Mode	R K	IN	Server mode: 0 = auto, 1 = transparent (not implemented yet)
3	Diag	X	IN	Diagnostic Definition. Format: “DIAG:Fxxx,Ryyy”
4	Process	X	IN	Data Processing definition. Format: “OFFSET=0, SWAP=1, DEFMAP=1, HOLES=0”
5	Status	R	OUT	Returned value (see below)
Returned Value				
	Success	= 0		UID successfully added
	Error	< 0		see chapter 6.3



Example		
CSF	S.SF.Modbus.Library	
	S.SF.Modbus.InitUID	
	1	; UID
	S.SF.Modbus.ServerMode.	
	Auto	
	DiagnosticUID	; “DIAG:F2000,R2000”
	Processing	; “OFFSET=0, SWAP=0, DEFMAP=1, HOLES=0”
	R 103	; Location to store CSF status

7.6 S.SF.Modbus.InitMap (Server)

S.SF.Modbus.InitMap		This function allows defining a mapping area; that is a set of Modbus media at specified address are mapped on a set of Saia PCD® media at a specified address It is not allowed to create an area which overlaps another one (same Modbus media type and overlapping Modbus address ranges)		
Parameters		10		
1	PartnerType	R K	IN	Type of Modbus media which will be mapped on Saia PCD® media
2	PartnerStart	R K	IN	Partner Start Address of the mapping area
3	PartnerRange	R K	IN	Number of elements of the mapping area (cannot be 0)
4	Saia PCD® Media Type	R K	IN	Type of Saia PCD® media on which the Modbus media will be mapped
5	Saia PCD® Start	R K	IN	Saia PCD® media start address corresponding to Partner Start Address. In case of mapping on DB, number of the DB the data will be mapped on.
6	Saia PCD® Range	R K	IN	Number of Saia PCD® elements in the mapping area (should correspond to number of partner elements, take care of AreaType used). In case of DB, number of element of this DB belonging to the area; 1st element of the area is element 0 of the DB. Saia PCD® Range cannot be 0. To avoid errors when accessing area, remember that, in a 32 bits area, 1 Saia PCD® registers corresponds to 2 holding registers (see Figure 12 below).
7	AreaType	R K	IN	Type of mapping area
8	AccessType	R K	IN	Type of mapping area access
9	UID	R K	IN	UID concerned by this mapping area
10	Status	R	OUT	Returned value (see below)
Returned Value				
	Success	= 0		Mapping area added successfully
	Error	< 0		see chapter 6.3

Example		
CSF	S.SF.Modbus.Library	
	S.SF.Modbus.InitMap	
	S.SF.Modbus.ModbusType.Coils	
	1000	; Partner Start Address
	2000	; Partner Range
	S.SF.Modbus.PCDMedia.Flag	
	100	; PCD Start Address
	2000	; PCD Range
	S.SF.Modbus.AreaType.Coils	
	S.SF.Modbus.AreaAccess.ReadWrite	
	1	; UID concerned by this mapping
	R 104	; Location to store CSF status

In this example, the coils with Modbus address in range 2000 to 3999 are respectively mapped onto Saia PCD® flags 100 to 2099 (for UID 1).

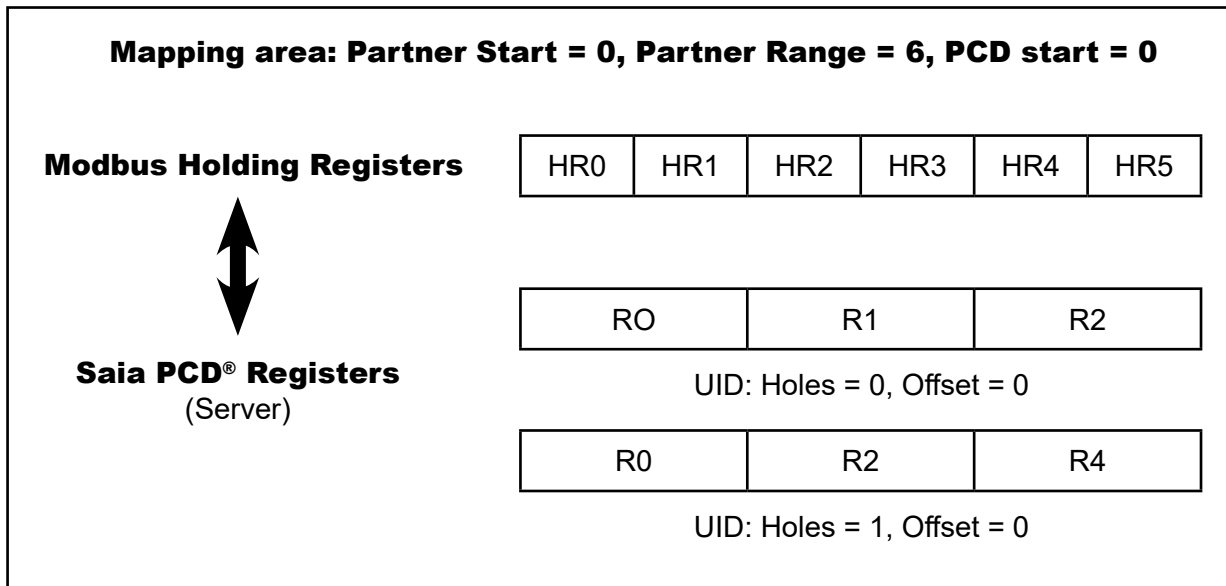


Figure 12: Mapping Holding Registers (16 bits) on Saia PCD® Registers (32 bits) in a 32 bits area






In the 32 bit area described above, Modbus element 1, 3 and 5 cannot be accessed: an error will be returned. Accessing HR2 with 3 Saia PCD® registers (that is 6 HR) will also return an error.

To access the HR 0 with 3 Saia PCD® registers (= 6 HR), the mapping area must be defined with at least 6 HR beginning at address 0 corresponding with:

- at least 3 Saia PCD® registers beginning at address 0 (1st case)
- at least 5 Saia PCD® registers beginning at address 0 (2nd case)

A Appendix

A.1 Icons

	<p>In manuals, this symbol refers the reader to further information in this manual or other manuals or technical information documents. As a rule there is no direct link to such documents.</p>
	<p>This symbol warns the reader of the risk to components from electrostatic discharges caused by touch. Recommendation: Before coming into contact with electrical components, you should at least touch the Minus of the system (cabinet of PGU connector). It is better to use a grounding wrist strap with its cable permanently attached to the Minus of the system.</p>
	<p>This sign accompanies instructions that must always be followed.</p>
	<p>Explanations beside this sign are valid only for the Saia PCD® Classic series.</p>
	<p>Explanations beside this sign are valid only for the Saia PCD® xx7 series.</p>



A.2 List of abbreviations

UID	Unit Identifier
HR	Holding Registers
IR	Input Registers
DI	Digital Inputs
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
FFP	Motorola (Freescale); Format, used by the PCD for floating point values.
IEEE	Institute of Electrical and Electronics Engineers
IEEE 754	32 Bit floating point format, used by a number of non-Saia controllers/system
LSW	Least Significant Word
MSW	Most Significant Word
DB	Data Block
I/O	Inputs/Outputs
SR	Serial

A.3 Contact**Contact****Saia-Burgess Controls AG**

Bahnhofstrasse 18
3280 Murten, Switzerland

Telephone switchboard..... +41 26 580 30 00
Telephone SBC Support..... +41 26 580 31 00
Fax..... +41 26 580 34 99

Support

E-mail Support: support@saia-pcd.com
Support site: www.sbc-support.com
SBC site: www.saia-pcd.com

International representations &
SBC sales companies: www.saia-pcd.com/contact

Repair**A****Postal address for customers to return products in Switzerland:****Saia-Burgess Controls AG**

After sales service
Bahnhofstrasse 18
3280 Murten, Switzerland