Technical specification

Studer Modbus RTU Protocol for Xcom-485i

Date	:	02.03.2021
Version	:	V1.1.2

409T

CONTENTS

1 INTRODUCTION

This technical specification describes the "Studer Modbus RTU Protocol" for the Xcom-485i communication module (Refer to Xcom-485i User manual). This protocol enables the control of a Studer system from a third party device (PLC, SCADA, etc.) using a RS-485 based communication interface. Using this protocol, it is possible to:

- Read user info
- Read parameters
- Write parameters
- Read pending messages

1.1 CONVENTIONS USED IN THIS DOCUMENT

Numbers starting with a "0x" prefix are hexadecimal numbers, otherwise, there are decimal numbers.

1.2 LIST OF ACRONYMS

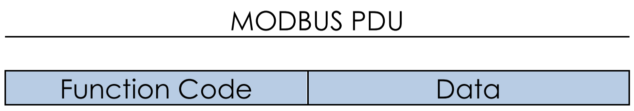
RCC	The Studer Innotec remote control used to configure the Xtender system.
PDU	Protocol Data Unit (see Modbus specification for more information)

2 MODBUS RTU IMPLEMENTATION

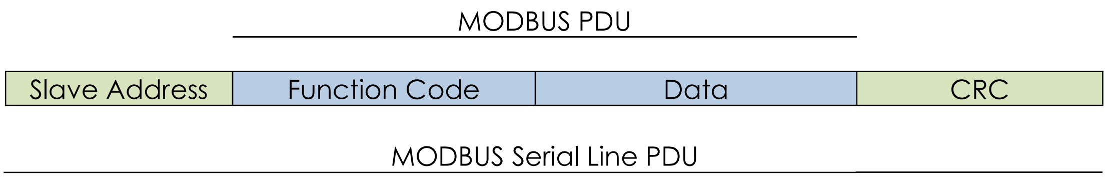
The Xcom-485i device offers a way for installers/integrators/developers to access and control an Xtender system using the Modbus RTU protocol. The Modbus protocol is standard and well known and used in the industry field. For more information regarding Modbus, please see the Modbus official web page: www.modbus.org.

2.1 PROTOCOL OVERVIEW

The Modbus application protocol defines the Modbus Protocol Data Unit (PDU) which is independent on the communication layer. This PDU has the following format:



On RS-485, the standard Modbus PDU is encapsulated and fields are added in order to make communication possible over a serial line.



The Slave Address field (8 bits) is used by the Modbus Master to access a slave device on the bus. The CRC field (16 bits – CRC16 Modbus as described into the 6.2.2 chapter of *Modbus_over_serial_line_v1_02.pdf* document) with the low byte first and high byte second is used to check if the frame transmission is done successfully.-For more information regarding Modbus and related documents, please see the Modbus official web page: www.modbus.org.

2.2 DEFAULT CONFIGURATION

By default, the Xcom-485i has the following Modbus configuration

Modbus	Value	Remarks
Addressing	1 to 63 33 to 95 65 to 127 129 to 192	Configurable by dip switches, see chapter 3.1 for more information.
Baud Rate	9'600 bps 19'200 bps 38'400 bps 115'200 bps	Configurable by dip switches, see chapter 2.3 for more information.
Parity	Even	
Protocol	Modbus RTU only	
Byte transmission	1 start bit 8 data bits, LSB first 1 parity bit (Even) 1 stop bit	
Electrical Interface	RS-485 on 2 wires	
Connector type	RJ-45	

Frames endianness are defined by the Modbus standard and are big endian, so MSB (Most Significant Byte) is sent first on the medium. For example, a 16 bits value of 0x1234 will be send 0x12 then 0x34 on the medium.

2.3 BAUD RATE SELECTION

As explained above, the RS-485 baud rate can be selected using the dip switches 7 and 8 of the Xcom-485i. The following tables shows how to select it.

Position		Baud rate
7	8	
OFF	OFF	9'600 bps
OFF	ON	19'200 bps
ON	OFF	38'400 bps
ON	ON	115'200 bps

3 MODBUS RTU FOR XTENDER SYSTEM

The Xcom-485i Studer Modbus RTU Protocol offers the following functionalities in the “Xtender World”:

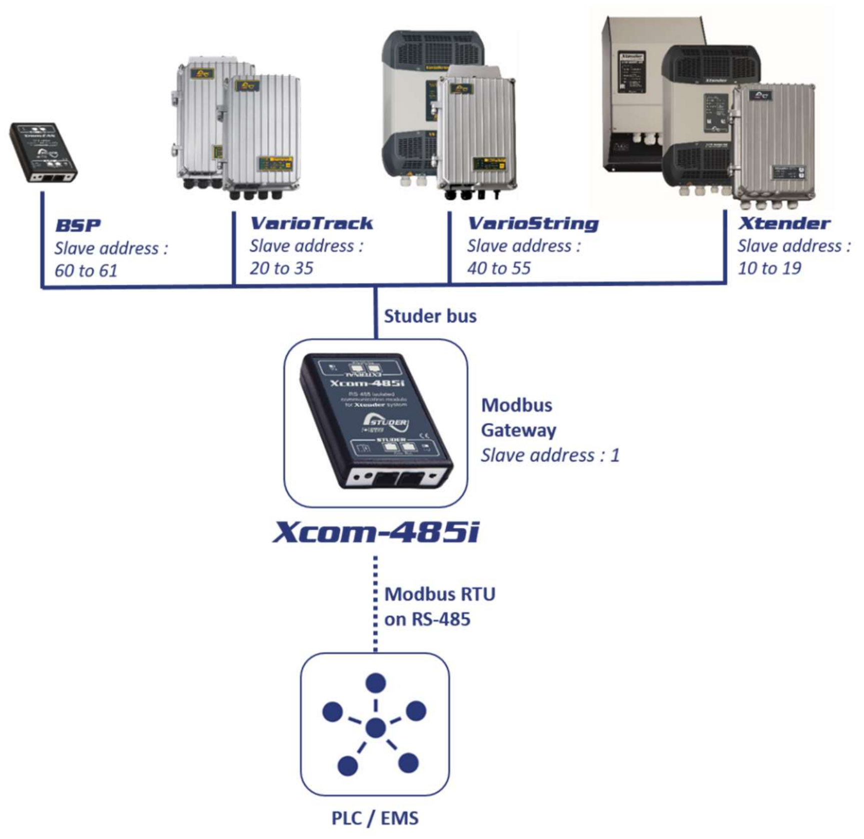
- Read parameters from Flash (value, min possible value, max possible value)
- Write parameters in Flash (and RAM)
- Write parameters in RAM only
- Read User Infos
- Read messages

Also, it is possible to access multiple devices of the same kind (e.g. multicast access on all Xtender devices) or to access a single device (e.g. unicat access to Xtender 1).

The Slave Address field is used by the Xcom-485i gateway to access Studer Innotec devices. Please see chapter 3.1 for more information.

3.1 ADDRESSING STUDER DEVICES

As the Xtender system can be made of multiple devices, we decided to implement the addressing of the Studer devices using the Modbus slave address encoded on 8 bits. So, the Xcom-485i will do the gateway and will respond to a certain address range. This address range can be mapped using the dip switches of the Xcom-485i device. So, an offset can be programmed with dip switches 1 and 2. The following figure shows how the Slave address are distributed in an Xtender system with dipswitches 1 and 2 set to off.



The following tables shows the Address range and the corresponding Studer devices.

Address Offset	**Devices**	**Remarks**
1	Xcom-485i Modbus gateway	The gateway itself for configuration and status
2 to 6	Reserved	
7 to 9	Virtual address to access all XTH, XTM and XTS present on the same phase 7 for L1 8 for L2 9 for L3	See section "multicast accesses"
10	Virtual address to access all XTH, XTM and XTS	See section "multicast accesses"
11 to 19	A single XTH, XTM or XTS inverter/charger	Ordered by the index displayed on the RCC
20	Virtual address to access all VarioTrack	See section "multicast accesses"
21 to 35	A single VarioTrack	Ordered by the index displayed on the RCC
36 to 39	Reserved	
40	Virtual address to access all VarioString	See section "multicast accesses"
41 to 55	A single VarioString	Ordered by the index displayed on the RCC
56 to 59	Reserved	
60	Virtual address to access all BSP or Xcom-CAN BMS (only one per installation)	See section "multicast accesses"
61	A single BSP or Xcom-CAN BMS	Only one BSP per installation
62 to 63	Reserved	

The following tables shows the Address offset that can be programmed using the dip switches 1 and 2.

Position		Address Offset	Address range
1	2		
OFF	OFF	0	(0) 1 to 63
	ON	32	(0) 33 to 95
ON	OFF	64	(0) 65 to 127
	ON	128	(0) 129 to 192

3.2 MULTICAST ACCESSES

Multicast accesses enable the possibility to access a group of devices of the same kind. So, it is possible to access all Xtenders (XTH, XTM, XTS or a mix), all VarioTracks or all VarioStrings. In order to do it, you need to use Multicast address as described in the previous table. For reasons of uniformity, the BSP can also be accessed with a multicast address.

A write parameter operation to a multicast address will have the effect to change the parameter value on all devices of the same kind. A read parameter/user info operation will return the value of the first device displayed in the RCC list in the group you are targetted.

As an exemple, if you write a parameter at address “10”, all Xtenders will have the corresponding value changed according to your write operation. If you read a parameter or a user information at address “10”, the system will return the value of the Xtender from address “11”, which is the first Xtender.

Note that the same behavior is guaranteed with phase group accesses, a read parameter/user info operation will return the value of the first Xtender on this phase and a write parameter operation acts on all the Xtenders of the same phase (as defined physically by the jumper on the PCB inside the Xtender).

As an example, if you write a parameter at address “7”, all Xtenders present on phase L1 will have the corresponding value changed according to your write operation. If you read a parameter or a user information at address “7”, the system will return the value of the first Xtender present on phase L1.

3.3 Unicast Accesses

Unicast accesses enable the possibility to access a single device in the installation. So any single Xtender, VarioTrack or VarioString can be access using Unicast address.

As an exemple, assume that you have 3 Xtenders in your system. Imagine that you want to access the second one displayed in the “System Info” of your RCC, you will need to use Unicast address “12”.

3.4 Response delay

The response delay of the Xcom-485i could be up to 1 second. This is a good value for a timeout in the master implementation.

The response delay depends on the internal Studer bus load (number of devices, number of RCC, values displayed on the RCC, etc.). The use of the data logger on other RCCs could cause a periodic increase of the response delay every 60 seconds as it needs to do more transactions on the Studer CAN bus.

It is strongly recommended not to spam the Xcom-485i with multiple requests. The correct way to communicate with the Xcom-485i is to send a request and to wait for the response before sending the next request. If no response comes from Xcom-485i after a delay of 1 second, we can consider that the timeout is over, and another request can be send. It is also how Modbus RTU is intended to work.

4 STUDER INNOTEC MODBUS FUNCTION CODE

Studer Innotec parameters ,user information and messages are accessible using Modbus Function code. The following table maps the Xtender system function to the Modbus Function code.

Xtender System Functions	Modbus Function Name	Modbus Function Code
Read Parameter More than 1 is possible	Read Holding Registers	3 / 0x03
Read User Info More than 1 is possible	Read Input Registers	4 / 0x04
Write Parameter More than 1 is possible	Write Multiple Registers	16 / 0x10
Read messages	Read Input Registers	4 / 0x04

Modbus uses the same mechanism between the function code to access registers. The request sent by the master contains always the Function Code, the Register Starting Address and the Quantity of Registers to be access. Some more information can be mandatory inside the request depending on the Function Code. Please see Modbus specifications for more information.

Modbus Request PDU general structure :

Function Code	1 Byte	0x03 or 0x04 or 0x10
Register Starting Addres	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	2 to n (minimum 2 for 32 bits float access) Must be an even number (2, 4, 6, etc.)
…	…	…

Modbus registers are defined as 16 bits registers. These registers are accessible using a 16 bits register address. For the Studer Innotec implementation, values are mostly encoded in 32 bits float format. So it will be needed to read/write 2 consecutive 16 bits registers to perform the full transaction (read/write parameter or read user info). Modbus register address have been defined for every parameter and user info. ***Please see the "Technical specification - Modbus appendix.pdf" available on our website to get the full list of Modbus register.***

5 READING MESSAGES

Within the Xtender system, any device (Xtender, VarioTrack, VarioString, BSP, Xcom-CAN BMS) can send messages. These messages are displayed on the RCC and also available on the Studer Portal whenever the installation is connected to the Internet.

In case of a Modbus RTU implementation, there is no way for the slave to send messages directly to the Modbus master. So, the master should read some registers in order to know if some messages are pending. For this reason, there is a slave address to access the Xcom-485i gateway itself. It is the gateway that will record all the pending messages and make it available to the Modbus master. The Modbus master will just need to perform some Modbus Read Input Registers to the right registers to get the messages.

At address 0x0000, there is a first register that contains the number of currently pending messages. It should be read first using Modbus Read Input Registers with the "Quantity of Input

Registers" set to 1. It should be done like that, otherwise the Xcom-485i will send back an exception. The following table shows the register map

Xcom-485i register address	Register content	Register size
0x0000	Number of pending messages	16 bits unsigned

Modbus Request PDU :

Function Code	1 Byte	4
Register Starting Addres	2 Bytes	0x0000
Quantity of Registers	2 Bytes	1 **(always 1, otherwise exception is returned)**

Modbus Request on serial line :

To retrieve the numbers of pending messages, you will have to read the register 0x0000 on the gateway that has the slave address n° 0x01, assuming that Address Offset is set to 0.

MODBUS PDU

Slave Address 1 byte	Function Code 1 byte	Data		CRC 2 bytes
		Start register address 2 bytes	Quantity of registers 2 bytes	
0x01	0x04	0x0000	0x0001	0x31CA

Modbus Response PDU :

Function Code	1 Byte	4
Byte Count	1 Byte	2
Input Register 0	2 Bytes	12 (12 messages pending in this example)

Modbus Response on serial line :

In this example the following response will be present on the the serial line, as all registers are 16 bits length the byte count field will be 0x02. The register 0 contains 0x000C that means 12 messages are pending on the gateway.

MODBUS PDU

Slave Address 1 byte	Function Code 1 byte	Data		CRC 2 bytes
		Byte count 1 byte	Register 0 2 bytes	
0x01	0x04	0x02	0x000C	0xB935

At address 0x0001, the oldest message can be accessed using Modbus Read Input Registers with the "Quantity of Input Registers" set to 4. It should be done like that, otherwise the Xcom-485i will send back an exception. The following table shows the registers map.

Xcom-485i register address	Register content	Register size
0x0001	Oldest message: Device Source	16 bits unsigned
0x0002	Oldest message: Message Id	16 bits unsigned
0x0003	Oldest message: Optional value Most significant word	16 bits unsigned
0x0004	Oldest message: Optional value Least significant word	16 bits unsigned

Modbus Request PDU :

Function Code	1 Byte	4
Register Starting Addres	2 Bytes	0x0001
Quantity of Registers	2 Bytes	4 **(always 4, otherwise exception is returned)**

Modbus Request on serial line :

As the precedent example response indicates 12 pending messages on the gateway, it is necessary to perform 12 times the following request. Once a message is read, it is deleted inside the Xcom-485i. The number of messages that can be stored inside the Xcom-485i is 128.

MODBUS PDU

Slave Address	Function Code	Data		CRC
		Start register address	Quantity of registers	
1 byte	1 byte	2 bytes	2 bytes	2 bytes
0x01	0x04	0x0001	0x0004	0xA009

Modbus Response PDU :

Function Code	1 Byte	4
Byte Count	1 Bytes	8
Input Register 0	2 Bytes	Device source
Input Register 1	2 Bytes	Message Id
Input Register 2	2 Bytes	Optional value Most significant word
Input Register 3	2 Bytes	Optional value Least significant word

Modbus Response on serial line :

As a response, one of the 12 messages is encoded as shown below and indicates that a warning happened due to a fan error detected, because the message id 0x0010 is message n°16 on the RCC, please refer to appendix to a complete list of RCC messages.

MODBUS PDU

Slave Address	Function Code	Data					CRC
		Byte count	Reg 0	Reg 1	Reg 2	Reg 3	
1 byte	1 byte	1 byte	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes
0x01	0x04	0x08	0x000B	0x0010	0x0000	0x0000	0x5F0E

6 GATEWAY REGISTERS

The gateway itself stores read-only informations that may be useful to retreive as the number of each Studer Innotec devices present on the proprietary bus as defined under *3.1 Addressing Studer Devices* and also identiy card of the gateway itself containing software version, hardware version and the unique FID number.

6.1 READ NUMBER OF STUDER DEVICES

To read the number of Studer devices accessible the registers presented below must be read accessing the gateway itself. Note that the read access to those specific registers must be performed with a single request reading the 5 registers at once.

Xcom-485i register address	Register content	Register size
0x0005	Number of Xtender devices	16 bits unsigned
0x0006	Number of VarioTrack devices	16 bits unsigned
0x0007	Number of VarioString devices	16 bits unsigned
0x0008	Number of BSP devices (also Xcom-CAN with BMS mode)	16 bits unsigned
0x0009	Number of RCC and Gateways	16 bits unsigned

Modbus Request PDU :

Function Code	1 Byte	4
Register Starting Addres	2 Bytes	0x0005
Quantity of Registers	2 Bytes	5 **(always 5, otherwise exception is returned)**

Modbus Request on serial line :

To retrieve the numbers Studer devices, you will have to read the register 0x0005 to 0x0009 on the gateway that has the slave address n° 0x01, assuming that Address Offset is set to 0.

MODBUS PDU

Slave Address 1 byte	Function Code 1 byte	Data		CRC 2 bytes
		Start register address 2 bytes	Quantity of registers 2 bytes	
0x01	0x04	0x0005	0x0005	0x2008

Modbus Response PDU :

Function Code	1 Byte	4
Byte Count	1 Byte	10
Input Register 5	2 Bytes	1 (1 Xtender present in this example)
Input Register 6	2 Bytes	1 (1 VarioTrack present in this example)
Input Register 7	2 Bytes	1 (1 VarioString present in this example)
Input Register 8	2 Bytes	1 (1 BSP present in this example)
Input Register 9	2 Bytes	2 (1 RCC + 1 Xcom-485i present in this example)

Modbus Response on serial line :

In this example the following response will be present on the the serial line, as all registers are 16 bits length the byte count field will be 0x0A.

MODBUS PDU

Slave Address 1 byte	Function Code 1 byte	Data							CRC 2 bytes
		Byte count 1 byte	Reg 5 2 bytes	Reg 6 2 bytes	Reg 7 2 bytes	Reg 8 2 bytes	Reg 9 2 bytes		
0x01	0x04	0x0A	0x0001	0x0001	0x0001	0x0001	0x0002		0x21EC

6.2 READ GATEWAY IDENTIY CARD

To read the identity card of the gateway the registers presented below must be read accessing the gateway itself. Note that the read access to those specific registers must be performed with a single request reading the 5 registers at once.

Xcom-485i register address	Register content	Register size
0x000A	Version Software MSB	16 bits unsigned
0x000B	Version Software LSB	16 bits unsigned
0x000C	Version Hardware	16 bits unsigned
0x000D	FID MSB	16 bits unsigned
0x000E	FID LSB	16 bits unsigned

Modbus Request PDU :

Function Code	1 Byte	4
Register Starting Addres	2 Bytes	0x000A
Quantity of Registers	2 Bytes	5 **(always 5, otherwise exception is returned)**

Modbus Request on serial line :

To retrieve the identity card, you will have to read the register 0x000A to 0x000E on the gateway that has the slave address n° 0x01, assuming that Address Offset is set to 0.

MODBUS PDU

Slave Address 1 byte	Function Code 1 byte	Data		CRC 2 bytes
		Start register address 2 bytes	Quantity of registers 2 bytes	
0x01	0x04	0x000A	0x0005	0x100B

Modbus Response PDU :

Function Code	1 Byte	4
Byte Count	1 Byte	10
Input Register 10	2 Bytes	0x0100
Input Register 11	2 Bytes	0x0654
Input Register 12	2 Bytes	0x0100
Input Register 13	2 Bytes	0x4E72
Input Register 14	2 Bytes	0x0048

Modbus Response on serial line :

In this example the following response will be present on the the serial line, as all registers are 16 bits length the byte count field will be 0x0A. Note that the FID value shown as a response is specific to one device only. See below to decode each of the register content.

MODBUS PDU

Slave Address	Function Code	Data						CRC
		Byte count	Reg 10	Reg 11	Reg 12	Reg 13	Reg 14	
1 byte	1 byte	1 byte	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes
0x01	0x04	0x0A	0x0100	0x0654	0x0100	0x4E72	0x0048	0xD543

Software version encoding

The software version is of the the form X.Y.Z. encoded in two 16 bits unsigned values :

8 bit (MSB) : X	8 bit : reserved	8 bit: Y	8 bit (LSB) : Z
Input Register 10 (Version Software MSB)		Input Register 11 (Version Software LSB)	

Following the example presented above the software version is 1.6.84 :

0x01 : X	0x00 : non-used	0x06 : Y	0x54 : Z
Input Register 10 (Version Software MSB)		Input Register 11 (Version Software LSB)	

Hardware version encoding

The software version is of the the form X.Y. encoded in one 16 bits unsigned values. Following the example presented above the hardware version is 1.0 :

0x01 : X	0x00 : Y
Input Register 12 (Version Hardware)	

FID encoding

The FID is a unique identifier for the gateway encoded a 32 bit unsigned value. The value is formed by combining FID MSB and FID LSB.

16 bit (MSB) : X	16 bit (LSB) : Y
Input Register 13 (FID MSB)	Input Register 14 (FID LSB)

Following the example presented above the FID is 0x4E720048.

7 USER INFOS : READ INPUT REGISTERS

The available user information is the same as the values that can be chosen to be displayed on the RCC. This user information gives the current state of the system. The user information can not be modified, and their values change during the operation of the system.

The corresponding Modbus function code to read User Infos is Modbus Read Input Registers. The "Quantity of Registers" need to be set to 2, otherwise the Xcom-485i will send back an exception.

7.1 READ USER INFO EXAMPLE

Read the battery temperature (info user 3001) on Xtender 1.

Modbus Request PDU :

Function Code	1 Byte	4
Register Starting Address	2 Bytes	2
Quantity of Registers	2 Bytes	2 **(always 2, otherwise exception is returned)**

Modbus Request on serial line :

The value of user info 3001 is stored in register 2 and 3 and in order to read this user info on Xtender 1, the request must be addressed to slave n°11 (0x0B) , assuming that Address Offset is set to 0.

MODBUS PDU

Slave Address 1 byte	Function Code 1 byte	Data		CRC 2 bytes
		Start register address 2 bytes	Quantity of registers 2 bytes	
0x0B	0x04	0x0002	0x0002	0xD0A1

Modbus Response PDU :

Function Code	1 Byte	4
Byte Count	1 Bytes	4
Input Register 0	2 Bytes	0x41D3
Input Register 1	2 Bytes	0x3333

The battery temperature on Xtender 1 is 0x41D33333 => 26.4 °C

Modbus Response on serial line :

MODBUS PDU

Slave Address 1 byte	Function Code 1 byte	Data			CRC 2 bytes
		Byte count 1 byte	Register 0 2 bytes	Register 1 2 bytes	
0x0B	0x04	0x04	0x41D3	0x3333	0xE0A4

8 READ PARAMETERS : READ HOLDING REGISTERS

All parameters accessible from RCC can also be accessed with the Modbus protocol. The corresponding Modbus function code to read Parameters is Modbus Read Holding Registers. The “Quantity of Registers” need to be set to 2, otherwise the Xcom-485i will send back an exception. It is possible to read the actual value of the parameter from Flash, but also the minimum and the maximum value.

To distinguish between these, we use a different register address offset as explained below:

- ***Read value from Flash : offset = 0***
- ***Read minimum possible value : offset = 2000***
- ***Read maximum possible value : offset = 4000***

8.1 READ PARAMETER VALUE FROM FLASH EXAMPLE

Read the value of the parameter {1107} (Maximum current on AC source) on Xtender 1.

Modbus Request PDU :

Function Code	1 Byte	3
Register Starting Addres	2 Bytes	14
Quantity of Registers	2 Bytes	2 **(always 2, otherwise exception is returned)**

Modbus Request on serial line :

The flash value of parameter 1107 is stored in register 14 and 15 and in order to read this user info on Xtender 1, the request must be addressed to slave n°11 (0x0B) , assuming that Address Offset is set to 0.

MODBUS PDU

Slave Address	Function Code	Data		CRC
		Start register address	Quantity of registers	
1 byte	1 byte	2 bytes	2 bytes	2 bytes
0x0B	0x03	0x000E	0x0002	0xA562

Modbus Response PDU :

Function Code	1 Byte	3
Byte Count	1 Bytes	4
Input Register 0	2 Bytes	0x4170
Input Register 1	2 Bytes	0x0000

The value of parameter {1107} is 0x41700000 => 15.0 A

Modbus Response on serial line :

Slave Address	Function Code	Data			CRC
		Byte count	Register 0	Register 1	
1 byte	1 byte	1 byte	2 bytes	2 bytes	2 bytes
0x0B	0x03	0x04	0x41F0	0x0000	0x443C

MODBUS PDU: Function Code, Data

8.2 READ THE MINIMUM POSSIBLE VALUE FROM FLASH EXAMPLE

Read the minimum possible value of parameter {1107} (Maximum current on AC source) on Xtender.

Modbus Request PDU :

Function Code	1 Byte	3
Register Starting Addres	2 Bytes	2014 = 14 **+ 2000**
Quantity of Registers	2 Bytes	2 **(always 2, otherwise exception is returned)**

Modbus Request on serial line :

The minimal allowed value of parameter 1107 is stored in register 2014 and 2015 and in order to read this user info on Xtender 1, the request must be addressed to slave n°11 (0x0B) , assuming that Address Offset is set to 0.

Slave Address	Function Code	Data		CRC
		Start register address	Quantity of registers	
1 byte	1 byte	2 bytes	2 bytes	2 bytes
0x0B	0x03	0x07DE	0x0002	0xA5EF

MODBUS PDU: Function Code, Data

Modbus Response PDU :

Function Code	1 Byte	3
Byte Count	1 Bytes	4
Input Register 0	2 Bytes	0x4000
Input Register 1	2 Bytes	0x0000

The minimum possible value of parameter {1107} is 0x40000000 => 2.0 A

Modbus Response on serial line :

Slave Address	Function Code	Data			CRC
		Byte count	Register 0	Register 1	
1 byte	1 byte	1 byte	2 bytes	2 bytes	2 bytes
0x0B	0x03	0x04	0x4000	0x0000	0x45F3

MODBUS PDU: Function Code, Data

8.3 READ THE MAXIMUM POSSIBLE VALUE FROM FLASH EXAMPLE

Read the maximum possible value of parameter {1107} (Maximum current on AC source) on Xtender.

Modbus Request PDU :

Function Code	1 Byte	3
Register Starting Addres	2 Bytes	4014 = 14 **+ 4000**
Quantity of Registers	2 Bytes	2 **(always 2, otherwise exception is returned)**

Modbus Request on serial line :

The maximal allowed value of parameter 1107 is stored in register 4014 and 4015 and in order to read this user info on Xtender 1, the request must be addressed to slave n°11 (0x0B) , assuming that Address Offset is set to 0.

MODBUS PDU

Slave Address 1 byte	Function Code 1 byte	Data		CRC 2 bytes
		Start register address 2 bytes	Quantity of registers 2 bytes	
0x0B	0x03	0x0FAE	0x0002	0xA654

Modbus Response PDU :

Function Code	1 Byte	3
Byte Count	1 Bytes	4
Input Register 0	2 Bytes	0x4248
Input Register 1	2 Bytes	0x0000

The maximum possible value of parameter {1107} is 0x42480000 => 50.0 A

Modbus Response on serial line :

MODBUS PDU

Slave Address 1 byte	Function Code 1 byte	Data			CRC 2 bytes
		Byte count 1 byte	Register 0 2 bytes	Register 1 2 bytes	
0x0B	0x03	0x04	0x4248	0x0000	0xC45D

9 WRITE PARAMETERS : WRITE MULTIPLE REGISTERS

All parameters accessible from RCC can also be accessed with the Modbus protocol. The corresponding Modbus function code to write parameters is Modbus Write Multiple Registers. The "Quantity of Registers" need to be set to 2, otherwise the Xcom-485i will send back an exception. In the Xtender system, it is possible to write a value in Flash (and RAM) or in RAM only. To distinguish between both, we use a different register address offset as explained below:

- ***Write in Flash (and Ram) : offset = 0***
- ***Write in RAM only : offset = 6000***

9.1 WRITE IN FLASH EXAMPLE

Write in Flash (and RAM) the Maximum current of AC source {1107} on Xtender 1.

Modbus Request PDU :

Function Code	1 Byte	16
Register Starting Addres	2 Bytes	14
Quantity of Registers	2 Bytes	2 **(always 2, otherwise exception is returned)**
Byte Count	1 Byte	4
Register value 0	2 Bytes	0x4180
Register value 1	2 Bytes	0x0000

Modbus Request on serial line :

The value of parameter 1107, that is stored into flash memory, is written in register 14 and 15 and in order to write this parameter value on Xtender 1, the request must be addressed to slave n°11 (0x0B) , assuming that Address Offset is set to 0.

	MODBUS PDU						
Slave Address	Function Code	Data					CRC
		Start reg addr	Qty of reg	Byte Count	Reg 0	Reg 1	
1 byte	1 byte	2 bytes	2 bytes	1 byte	2 bytes	2 bytes	2 bytes
0x0B	0x10	0x000E	0x0002	0x04	0x4180	0x0000	0x462F

Modbus Response PDU :

Function Code	1 Byte	16
Register Starting Addres	2 Bytes	14
Quantity of Registers	2 Bytes	2

A write operation responds with the quantity of written registers, that is always 2 in this case.

Modbus Response on serial line :

Slave Address 1 byte	Function Code 1 byte	Data Start register address 2 bytes	Data Quantity of registers 2 bytes	CRC 2 bytes
0x0B	0x10	0x000E	0x0002	0x20A1

(MODBUS PDU: Function Code and Data)

9.2 Write in RAM only example

Write in RAM only the Maximum current of AC source {1107} on Xtender 1.

Modbus Request PDU :

Function Code	1 Byte	16
Register Starting Addres	2 Bytes	6014 = 14 **+ 6000**
Quantity of Registers	2 Bytes	2 **(always 2, otherwise exception is returned)**
Byte Count	1 Byte	4
Register value 0	2 Bytes	0x4180
Register value 1	2 Bytes	0x0000

Modbus Request on serial line :

The value of parameter 1107, that is stored into RAM memory, is written in register 6014 and 6015 and in order to write this parameter value on Xtender 1, the request must be addressed to slave n°11 (0x0B) , assuming that Address Offset is set to 0.

Slave Address 1 byte	Function Code 1 byte	Data Start reg addr 2 bytes	Data Qty of reg 2 bytes	Data Byte Count 1 byte	Data Reg 0 2 bytes	Data Reg 1 2 bytes	CRC 2 bytes
0x0B	0x10	0x177E	0x0002	0x04	0x4180	0x0000	0xAAFB

(MODBUS PDU: Function Code and Data)

Modbus Response PDU :

Function Code	1 Byte	16
Register Starting Addres	2 Bytes	6014 = 14 **+ 6000**
Quantity of Registers	2 Bytes	2

A write operation responds with the quantity of written registers, that is always 2 in this case.

Modbus Response on serial line :

Slave Address 1 byte	Function Code 1 byte	Data Start register address 2 bytes	Data Quantity of registers 2 bytes	CRC 2 bytes
0x0B	0x10	0x177E	0x0002	0x24CE

(MODBUS PDU: Function Code and Data)

9.3 Change value of parameters on the Xtender inverter

Changing parameters when the inverters are in operation should be done carefully. The modification of parameters can restart the corresponding algorithm inside the inverter. For example, the change of a delay can restarts the timer attached to it.

9.4 Cyclic write of parameters

When you are using the RCC remote control, the Xtender inverter/charger, VarioTrack and VarioString MPPT solar chargers store their parameter values in a non-volatile flash memory. Because of the endurance of this memory, the number of writes on a single parameter is only guaranteed for 1000 write operations.

To allow the cyclic write of parameters without count limit, we suggest you to write the parameters in RAM only (see chapter 9.2).

9.5 Data encoding

In appendix, there are tables presenting the user information and the parameters of each device. The format for each information/parameter is specified. All data are encoded according to the standard format IEEE 754-2008: single precision floating point. There is plenty of documentation regarding this format on the Internet. However, the format is explained for your convenience below:

sign	exponent	mantissa
1 bit	8 bits	23 bits

IEEE 754-2008 single precition 32 bits floating point format

As an example, the decimal value 0.0 is coded 0x00000000 and the decimal value 1.0 is coded 0x3F800000.

Below we have briefly explained all Studer specific formats.

9.5.1 “ENUM” format encoded as a float

An "ENUM" is an enumeration and can basically take one value in between a set of possible values. Each value has its own signification. For parameter, ENUM values are always power of two values (e.g. 0, 1, 2, 4, 8, etc.). For user info, ENUM values are always incremental values (e.g. 0, 1, 2, 3, 4, 5, 6, etc.). Remember that these values are encoded as a float, so:

0 will be encoded as 0.0 = 0x00000000,
1 will be encoded as 1.0 = 0x3F800000,
2 will be encoded as 2.0 = 0x40000000,
3 will be encoded as 3.0 = 0x40400000,
etc…

9.5.2 “BOOL” format encoded as a float

The "BOOL" format represents "TRUE" and "FALSE", encoded respectively as "+1.0" and "0.0". So:

"TRUE" will be encoded as 1.0 = 0x3F800000,
"FALSE" will be encoded as 0.0 = 0x00000000

9.5.3 “SIGNAL” format encoded as a float

The Signal format (e.g. parameter {1468}) is coded as a float. To send an active signal, you must write the value “1.0” (encoded 0x3F800000) as the parameter value.

9.5.4 “HOUR” format encoded as a float

The hour format encoding is in minutes beginning at 00:00 and terminating at 23:59. There is no field available for seconds. So:

13:41 = 13*60 + 41 = 821 minutes will be encoded as 821.0 = 0x444D4000

9.5.5 “DAYS of WEEK” format encoded as a float

The days of the week (e.g. parameter {1205}) is coded as a bit field in a float. To select a day, set its corresponding bit to 1.

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sunday	Saturday	Friday	Thursday	Wednesday	Tuesday	Monday

So:

Monday will be encoded as 2.0^0 = 1.0 = 0x3f800000
Sunday will be encoded as 2.0^6 = 64.0 = 0x42800000
Thursday + Saturday will be encoded as $2.0^3 + 2.0^5$ = 40.0 = 0x42200000

All days will be encoded as $2.0^6 + 2.0^5 + 2.0^4 + 2.0^3 + 2.0^2 + 2.0^1 + 2.0^0$ = 127.0 = 0x42fe0000