Eastron SDM630-EV Smart Meter Modbus Protocol V1.23

Version History

Date	Rev	Edited	Notes
22/08/25	V1.21	Luke	Initial issue
14/04/26	V1.22	Obito	Update the initial version
28/07/26	V1.23	Claire	Modify the quantity and content of charging logs and event logs in the protocol.

Modbus Protocol Overview

This section provides basic information for interfacing the Eastron Smart meter to a Modbus Protocol network. If background information or more details of the Eastron implementation is required please refer to section 2 and 3 of this document.

Eastron offers the option of an RS485 communication facility for direct connection to SCADA or other communications systems using the Modbus Protocol RTU salve protocol. The Modbus Protocol establishes the format for the master's query by placing into it the device address, a function code defining the requested action, any data to be sent, and an error checking field. The slave's response message is also constructed using Modbus Protocol. It contains fields confirming the action taken, any data to be returned, and an error-checking field. If an error occurs in receipt of the message, SDM630-EV will make no response. If the SDM630-EV is unable to perform the requested action, it will construct an error message and send it as the response.

The electrical interface is 2-wire RS485, via 2 screw terminals. Connection should be made using twisted pair screened cable (Typically 22 gauge Belden 8761 or equivalent). All "A" and "B" connections are daisy chained together. Line topology may or may not require terminating loads depending on the type and length of cable used. Loop (ring) topology does not require any termination load. The impedance of the termination load should match the impedance of the cable and be at both ends of the line. The cable should be terminated at each end with a 120 ohm (0.25 Watt min.) resistor. A total maximum length of 3900 feet (1200 meters) is allowed for the RS485 network. A maximum of 32 electrical nodes can be connected, including the controller. The address of each Eastron can be set to any value between 1 and 247. Broadcast mode (address 0) is supported.

The format for each byte in RTU mode is:
Coding System: 8-bit per byte
Data Format: 4 bytes (2 registers) per parameter.
Floating point format (to IEEE 754)
Most significant register first (Default). The default may be changed if required -See Holding Register "Register Order" parameter.

Error Check Field: 2 byte Cyclical Redundancy Check (CRC)
Framing: 1 start bit
8 data bits, least significant bit sent first
1 bit for even/odd parity (or no parity)
1 stop bit if parity is used; 1 or 2 bits if no parity

Data Coding

All data values in the SDM630-EV smart meter are transferred as 32 bit IEEE754 floating point numbers, (input and output) therefore each SDM630-EV meter value is transferred using two Modbus Protocol registers. All register read requests and data write requests must specify an even number of registers. Attempts to read/write an odd number of registers prompt the SDM630-EV smart meter to return a Modbus Protocol exception message. However, for compatibility with some SCADA systems, SDM630-EV Smart meter will response to any single input or holding register read with an instrument type specific value.

The SDM630-EV can transfer a maximum of 40 values in a single transaction; therefore the maximum number of registers requestable is 80. Exceeding this limit prompts the SDM630-EV to generate an exception response.

Data transmission speed is selectable between 2400, 4800, 9600, 19200, 38400baud.

1 Input register

Input registers are used to indicate the present values of the measured and calculated electrical quantities. Each parameter is held in two consecutive16 bit register.The following table details the 3X register address, and the values of the address bytes within the message. A (*) in the column indicates that the parameter is valid for the particular wiring system. Any parameter with a cross(X) will return the value zero. Each parameter is held in the 3X registers. Modbus Protocol function code 04 is used to access all parameters.

For example, to request: Amps 1 Start address=0006
No. of registers =0002
Amps 2 Start address=0008
No. of registers=0002

Each request for data must be restricted to 40 parameters or less. Exceeding the 40 parameter limit will cause a Modbus Protocol exception code to be returned.

SDM630-EV Input Registers

Address (Register)	**Parameter Number**	**SDM630-EV Input Register Parameter**		**Modbus Protocol Start Address Hex**		3 Ø	3 Ø	1 Ø
		Description	**Units**	**Hi**	**Lo**	4	3	2

				Byte	**Byte**	W	W	W
30001	1	Phase 1 line to neutral volts.	Volts	00	00	√	X	√
30003	2	Phase 2 line to neutral volts.	Volts	00	02	√	X	X
30005	3	Phase 3 line to neutral volts.	Volts	00	04	√	X	X
30007	4	Phase 1 current.	Amps	00	06	√	√	√
30009	5	Phase 2 current.	Amps	00	08	√	√	X
30011	6	Phase 3 current.	Amps	00	0A	√	√	X
30013	7	Phase 1 power.	Watts	00	0C	√	X	√
30015	8	Phase 2 power.	Watts	00	0E	√	X	X
30017	9	Phase 3 power.	Watts	00	10	√	X	X
30019	10	Phase 1 volt amps.	VA	00	12	√	X	√
30021	11	Phase 2 volt amps.	VA	00	14	√	X	X
30023	12	Phase 3 volt amps.	VA	00	16	√	X	X
30025	13	Phase 1 volt amps reactive.	VAr	00	18	√	X	√
30027	14	Phase 2 volt amps reactive.	VAr	00	1A	√	X	X
30029	15	Phase 3 volt amps reactive.	VAr	00	1C	√	X	X
30031	16	Phase 1 power factor .	None	00	1E	√	X	√
30033	17	Phase 2 power factor .	None	00	20	√	X	X
30035	18	Phase 3 power factor .	None	00	22	√	X	X
30037	19	Phase 1 phase angle.	Degrees	00	24	√	X	√
30039	20	Phase 2 phase angle.	Degrees	00	26	√	X	X
30041	21	Phase 3 phase angle.	Degrees	00	28	√	X	X
30043	22	Average line to neutral volts.	Volts	00	2A	√	X	X
30047	24	Average line current.	Amps	00	2E	√	√	√
30049	25	Sum of line currents.	Amps	00	30	√	√	√
30053	27	Total system power.	Watts	00	34	√	√	√
30057	29	Total system volt amps.	VA	00	38	√	√	√
30061	31	Total system VAr.	VAr	00	3C	√	√	√
30063	32	Total system power factor	None	00	3E	√	√	√
30067	34	Total system phase angle.	Degrees	00	42	√	√	√
30071	36	Frequency of supply voltages.	Hz	00	46	√	√	√
30073	37	Total Import kWh	kWh	00	48	√	√	√
30075	38	Total Export kWh.	kWH	00	4A	√	√	√
30077	39	Total Import kVArh .	kVArh	00	4C	√	√	√
30079	40	Total Export kVArh .	kVArh	00	4E	√	√	√
30081	41	Total VAh	kVAh	00	50	√	√	√
30083	42	Ah	Ah	00	52	√	√	√
30201	101	Line 1 to Line 2 volts.	Volts	00	C8	√	√	X

30203	102	Line 2 to Line 3 volts.	Volts	00	CA	√	√	X
30205	103	Line 3 to Line 1 volts.	Volts	00	CC	√	√	X
30207	104	Average line to line volts.	Volts	00	CE	√	√	X
30225	113	Neutral current.	Amps	00	E0	√	X	X
30343	172	Total kwh	kwh	01	56	√	√	√
30345	173	Total kvarh	kvarh	01	58	√	√	√
30347	174	L1 import kwh	kwh	01	5a	√	X	√
30349	175	L2 import kwh	kwh	01	5c	√	X	X
30351	176	L3 import kWh	kwh	01	5e	√	X	X
30353	177	L1 export kWh	kwh	01	60	√	X	√
30355	178	L2 export kwh	kwh	01	62	√	X	X
30357	179	L3 export kWh	kwh	01	64	√	X	X
30359	180	L1 total kwh	kwh	01	66	√	X	√
30361	181	L2 total kWh	kwh	01	68	√	X	X
30363	182	L3 total kwh	kwh	01	6a	√	X	X
30365	183	L1 import kvarh	kvarh	01	6c	√	X	√
30367	184	L2 import kvarh	kvarh	01	6e	√	X	X
30369	185	L3 import kvarh	kvarh	01	70	√	X	X
30371	186	L1 export kvarh	kvarh	01	72	√	X	√
30373	187	L2 export kvarh	kvarh	01	74	√	X	X
30375	188	L3 export kvarh	kvarh	01	76	√	X	X
30377	189	L1 total kvarh	kvarh	01	78	√	X	√
30379	190	L2 total kvarh	kvarh	01	7a	√	X	X
30381	191	L3 total kvarh	kvarh	01	7c	√	X	X
320005	122	Positive line loss energy	kWh	4E	24	√	√	√
320007	123	Negative line loss energy	kWh	4E	26	√	√	√
320011	124	Total line loss energy	kWh	4E	2A	√	√	√
320013	125	Charging gun power	kWh	4E	2C	√	√	√
320015	126	Positive charging gun energy	kWh	4E	2E	√	√	√
320017	127	Negative charging gun energy	kWh	4E	30	√	√	√

2 Holding Registers

Holding registers are used to store and display instrument configuration settings. All holding registers not listed in the table below should be considered as reserved for manufacturer use and no attempt should be made to modify their values.

The holding register parameters may be viewed or changed using the Modbus Protocol. Each parameter is held in two consecutive 4X registers. Modbus Protocol Function Code 03 is used to read the parameter and Function Code 16 is used to write. Write to only one parameter per message.

SDM630-EV MODBUS Protocol Holding Register Parameters

Address Register	Parameter	Modbus Protocol Start Address Hex		Valid range	Mode
		High Byte	Low Byte		
40011	System Type	00	0A	Write system type: 3p4w = 3, 3p3w = 2 & 1p2w= 1 Requires password, see parameter 13 **Length : 4 byte** **Data Format : Float**	r/wp
40015	Password Lock	00	0E	Read password lock status: 0 = locked. 1 = unlocked. Write operation: Write the correct password to obtain access permissions **Length : 4 byte** **Data Format : Float**	r/w
40019	Network Parity Stop	00	12	Write the network port parity/stop bits for MODBUS Protocol, where: 0 = One stop bit and no parity, default. 1 = One stop bit and even parity. 2 = One stop bit and odd parity.3 = Two stop bits and no parity.Requires a restart to become effective. **Length : 4 byte** **Data Format : Float**	r/w
40021	Network Node	00	14	Write the network port node address: 1 to 247 for MODBUS Protocol, default 1. Requires a restart to become effective. **Length : 4 byte** **Data Format : Float**	r/w
40025	Password	00	18	Write password for access to protected registers. **Length : 4 byte** **Data Format : Float**	r/w
40029	Network Baud Rate	00	1C	Write the network port baud rate for MODBUS Protocol, where: 0 = 2400 baud. 1 = 4800 baud. 2 = 9600 baud, default. 3 = 19200 baud. 4 = 38400 baud **Length : 4 byte** **Data Format : Float**	r/w

40061	Backlight time	00	3C	Default: 60, unit: min Range:0~121 0 means the backlight will be always on. 121means the backlight will be always off. **Length : 4byte** **Data Format : Float**	r/w
40113	End of charging wheel display control	00	70	0000: Controlled by time (default) 0001: Controlled by register write value **Length : 2byte** **Data Format :Hex**	r/w
40114	Total wheel display time at the end of charging	00	71	Range:1~65535(default 30)s **Length : 2byte** **Data Format :Hex**	r/w
40115	Rotation time per screen	00	72	XX YY XX represents the screen rotation time during charging YY is the screen rotation time after charging is completed XX 和 YY Range:1~255 **Length : 2byte** **Data Format :Hex**	r/w
40116	Exit the wheel display control after charging is completed	00	73	Write 0xA55A to exit the charging wheel display When the 0070 register is 0000, the register is invalid**Length : 2byte** **Data Format :Hex**	wo
40117	Interface control during charging	00	74	See appendix 2 in last page **Length : 2byte** **Data Format :Hex**	R/w
40118	Charging end interface control	00	75	See appendix 2 in last page **Length : 2byte** **Data Format :Hex**	r/w
40119	Data signature during charging	00	76	Write 0xA55A to sign **Length : 2byte** **Data Format :Hex**	wo
401521	mode switching	05	F0	0x0002 ： Assembly Mode 0x0003 ： User Mode The meter cannot exit after entering the user **mode Length : 2 byte** **Data Format: Hex** **(KPPA is asked)**	r/w
401522	ALFEN MODE	05	F1	0x0000 ： OFF 0x0001 ： ON **mode Length : 2 byte** **Data Format: Hex** **(KPPA is asked)**	r/w

401536	Line loss mode of Charging station	06	00	0000：Do not use line loss mode (default) 0001：2-wire mode Available in Assembly Mode **Length : 2byte** **Data Format :Hex**	r/w
401537	Line loss percentage of Charging station	06	01	Charging station line loss percentage 0 ~ 0.999 % Example: 0x0a = 0.01 % Available in Assembly Mode **Length : 2byte** **Data Format :Hex**	r/w
401539	Total charging data	06	02	Charging quantity 0~250000 **Length : 4byte** **Data Format :Hex**	ro
401541	Search for charging data by charging ID	06	04	Charging ID 1~250000 **Length : 4byte** **Data Format :Hex**	r/w
401545	Retrieved charging data	06	08	**Length :1024byte** **Data Format :ASCII**	ro
402056	Total log data volume	08	08	Total number of logs 0~3000 **Length : 4byte** **Data Format :Hex**	ro
402058	Finding data by log ID	08	0A	log ID 1~3000 **Length : 4byte** **Data Format :Hex**	r/w
402062	Found charging data	08	0E	**Length :200byte** **Data Format :ASCII**	ro
46001	Charging duration	17	70	Seconds **Length : 4 byte** **Data Format : HEX**	ro
46003	charging energy	17	72	total wh **Length : 4 byte** **Data Format : UINT32**	ro
46005	Start Timestamp	17	74	Length : 4 byte Data Format : Unix	ro
46007	Stop Timestamp	17	76	Length : 4 byte Data Format : Unix	ro
46050	Charging status	17	A1	0: idle 1: charge in progress 2: the system was powered off during charging session	ro

				3: the system was reset during charging session **Length: 2 bytes** **Data Format: HEX**	
46051	charge control	17	A2	0x01:Begin measurement (B) 0x02:End measurement(E) **Length : 2 byte** **Data Format : HEX**	r/w
46061	Signature status	17	AC	0x00:Not initialised 0x01:Idle 0x02:Signature in progress 0x03:Signature OK 0x04:Invalid date time 0x05:Invalid measurement 0x06: signature state error 0x07:Keypair generation Error 0x08:SHA failed 0x09:Public key error 0x10:Invalid message format 0x11:Invalid message size 0x12:Signature error 0x13:Undefined error **Length:2 bytes** **Data Format: HEX**	ro
46065	Signature Format	17	B0	0 HEX (ASN.1) default 1 HEX 2 Base64 **Length:2 bytes** **Data Format: HEX**	r/w
46066	Signature Algorithm	17	B1	0 Without signature 1 ECDSA-secp256r1-SHA256 default **Length:2 bytes** **Data Format: HEX**	r/w
47425	Signature Length	1D	00	**Length: 2 bytes** **Data Format: HEX**	ro
47426	Signature (raw)	1D	01	**Data Format: HEX**	ro
47937	Output Message Length	1F	00	**Length: 2 bytes** **Data Format: HEX**	ro
47938	Output Message	1F	01	**Data Format: HEX**	ro
48449	Output Message Length (JSON)	21	00	**Length: 2 bytes** **Data Format: HEX**	ro

<table>
<tr><td>48450</td><td>Output Message (JSON)</td><td>21</td><td>01</td><td>Data Format: HEX</td><td>ro</td></tr>
<tr><td>48961</td><td>Public Key (raw)</td><td>23</td><td>00</td><td>Data Format: HEX</td><td>ro</td></tr>
<tr><td>49217</td><td>Output Message Length (OCMF)</td><td>24</td><td>00</td><td>Length: 2 bytes
Data Format: HEX</td><td>ro</td></tr>
<tr><td>49218</td><td>Output Message (OCMF)</td><td>24</td><td>01</td><td>Data Format: HEX</td><td>ro</td></tr>
<tr><td>461439</td><td>Zone</td><td>EF</td><td>FE</td><td>Time Zone Range：-12~12
Length : 4 byte
Data Format : float</td><td>r/w</td></tr>
<tr><td>461441</td><td>Time</td><td>F0</td><td>00</td><td>s-min-hour-week-Date-Month-Year-20
Length : 8 byte
Data Format:BCD</td><td>r/w</td></tr>
<tr><td>461457</td><td>Reset Historical Data</td><td>F0</td><td>10</td><td>00 05 : Clear charging records
Length : 2 byte
Data Format:Hex</td><td>wo</td></tr>
<tr><td>464507</td><td>Meter state</td><td>FB</td><td>FA</td><td>读取状态代码。<table><tr><td></td><td></td><td></td></tr><tr><td>Bit 0</td><td>时间同步</td><td>0:RTC 时间未同步 1:RTC</td></tr><tr><td>Bit 1</td><td>充电状态</td><td>0：未充电 1：充电中</td></tr><tr><td>Bit 2~3</td><td>线制</td><td>00：无线制
01：2 线制
11：4 限制</td></tr><tr><td>Bit4</td><td>可以正常工作</td><td>0：不可以 1：可以</td></tr></table>Length : 4 byte
Data Format : hex</td><td>ro</td></tr>
<tr><td>464513</td><td>Serial number</td><td>FC</td><td>00</td><td>Meter serial number
Length : 4 byte
Data Format : hex</td><td>R0</td></tr>
<tr><td>464549</td><td>Com2 mode</td><td>FC</td><td>24</td><td>0x00:Modbus mode
0x01:Pulse mode
Length : 2 byte
Data Format:HEX</td><td>r/w</td></tr>
</table>

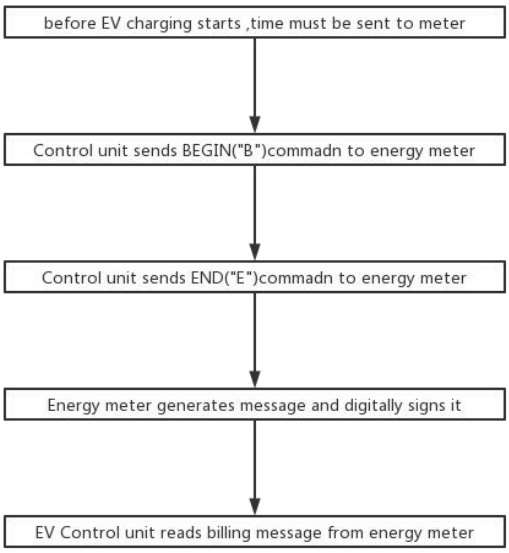
3 Digital signature

3. 1. Introduction

Energy meter supports digital signing of billing information to ensure integrity of data for end customer. All digital signing procedures are HW based with dedicated crypto chip, which supports ECDSA FIPS186-3 Elliptic Curve Digital Signature. Energy meter supports MODBUS over RS485 for communication with EV control unit.

3. 2. Digital signing procedure

EV charger control unit is responsible to send start and stop command to energy meter. Energy meter measures consumed energy during charging. When charging is finished, EV control unit provides billing dataset (customer info, time, etc.) to energy meter via MODBUS communication. Energy meter adds measured energy and generates final billing message with digital signature. EV charger control unit then reads complete billing information with measured energy consumption and digital signature.



3. 3. Energy meter cryptographic functions explanation

Energy meter has HW based cryptographic unit for digital signing of billing dataset.

3. 3. 1. Generation of private/public key par

This is one-time procedure made at production of energy meter. Generation of key pair is HW based

with dedicated crypto chip. Private key is stored internally within the crypto chip and there is no way of reading it.

3. 3. 2. public key read

Public key is available to end user for verification of digital signature. Therefore, public key is readable through MODBUS communication.
Public key is stored in 64 bytes raw format at MODBUS address 48961.
For **Transparenz Software** check, public key header should be prepended:
3059301306072A8648CE3D020106082A8648CE3D03010703420004
For checking with ECDSA, public key header is: 04.

3. 3. 3. Consumption measuring and digital signing procedure

EV charger control unit must use following procedure to measure charging consumption and sign dataset:
1. Set time, time zone, signature format
3. Send Begin command
4. Enter dataset size
4. Send intermediate reading commands (optional)
5. Send End command (triggers signing process)
6. Check signature status register until signature is ready
7. Read Output message length
8. Read Output message
9. Read signature length
10. Read signature
11. Read public key

3. 3. 4. Json data format

Format is compliant with OCMF v1.0.
Energy meter requires following fields in dataset:

```
OCMF|{
"FV":"1.0",                 //ocmf version
"GI":"",                    //mark
"GS":"",                    //serial number
"PG":"",                    //page
"MV":"",                    //meter supplier
```

```
"MM":"",                    //meter mark
"MS":"",                    //meter serial number
"MF":"",                    //meter firmware
"IS":true,                  //user assignment status
"IF":[],
"IT":"NONE",
"ID":",
"CT":"",
"CI":"",
"RD":[
{
"TM":"2019-11-11T13:22:28,000+0000 S",
"TX":"B",
"RV":123457.529,
"RI":"1-b:1.8.0",
"RU":"kWh",
"RT":"AC",
"EF":"",
"ST":"G"
},
{
"TM":"2019-11-11T13:24:12,000+0000 S",
"TX":"E",
"RV":123457.529,
"RI":"1-b:1.8.0",
"RU":"kWh",
"RT":"AC",
"EF":"",
"ST":"G"}
]
} |
{
"SD":string,
}
```

key	type	describe
FV	String	Format-Version: = "1.0"
GI	String	Gateway identification= "EASTRON EV".
GS	String	serial number (string of 8 char)
PG	String	Pagination of the entire dataset = string of "T<value>" with value increased for each read of transaction
MV	String	Meter-Vendor = "EASTRON"
MM	String	Instrument identification= “SDM630”
MS	String	serial number (string of 8 char)
MF	String	Meter-Firmware: "01.01"

IS	Boolean	Identification status: General status for user assignment: true: Users successfully assigned, false: Users not associated.
IF	Array of String	Identification flags for RFID, OCPP, ISO15118 and PLMN protocol
IT	String	Identification-Type: "string"
ID	String	Identification-Data: "string"
CT	String	Charge-Point-Identification-Type
CI	String	Charge-Point-Identification
RI	String	1-b:1.8.0 .Purchase of electrical energy (active energy) from the power grid (of the charging point operator) to the customer.
EF	String	"" No error "E" Error in the energy register "t" Error in the time status "Et" Error in the energy registers and the time status

3. 3. 5. OCMF Holding Registers

46145	IS	18	00	General status for user assignment 0x0000：true 0x0001：false Length : 2 byte Data Format : HEX	
46147	IF	18	02	Detailed statements on user assignment BIT0:RFID_NONE BIT1:RFID_PLAIN BIT2:RFID_RELATED BIT3:RFID_PSK BIT4:OCPP_NONE BIT5:OCPP_RS BIT6:OCPP_AUTH BIT7:OCPP_RS_TLS BIT8:OCPP_AUTH_TLS BIT9:OCPP_CACHE BIT10:OCPP_WHITELIST BIT11:OCPP_CERTIFIED BIT12:ISO15118_NONE BIT13:ISO15118_PNC BIT14:PLMN_NONE BIT15:PLMN_RING	

				BIT16:PLMN_SMS Length : 4 byte Data Format : HEX	
46149	IT	18	04	Type of identification data 0x0000:NONE 0x0001:DENIED 0x0002:UNDEFINED 0x0003:ISO14443 0x0004:ISO15693 0x0005:EMAID 0x0006:EVCCID 0x0007:EVCOID 0x0008:ISO7812 0x0009:CARD_TXN_NR 0x000A:CENTRAL 0x000B:CENTRAL_1 0x000C:CENTRAL_2 0x000D:LOCAL 0x000E:LOCAL_1 0x000F:LOCAL_2 0x0010:PHONE_NUMBER 0x0011:KEY_CODE Length : 2 byte Data Format : HEX	
46150	ID	18	05	Identification data Length : 40 byte Data Format : ASCII	
46170	CT	18	19	Charge-Point-Identification-Type 0X0001:EVSEID 0X0002:CBIDC Length : 2 byte Data Format : HEX	
46177	TT	18	20	TarifText Length:20bytes Data Format:ASCII	
46187	Cabe name	18	2A	Cable name Length : 8byte Available in Assembly Mode Data Format :ASCII	
46171	CI	18	30	Identification data Available in Assembly Mode Length : 26 byte	

				Data Format :ASCII	
46214	CF	18	45	Charge controller firmware The length of CF field is 25 bytes.So The last byte is 0x00. Length : 26 byte Data Format :ASCII	

Register Order controls the order in which the Eastron Digital meter receives or sends floating-point numbers: - normal or reversed register order. In normal mode, the two registers that make up a floating point number are sent most significant register first. In reversed register mode, the two registers that make up a floating point number are sent least significant register first. To set the mode, write the value '2141.0' into this register - the instrument will detect the order used to send this value and set that order for all Modbus Protocol transactions involving floating point numbers.

It is perfectly feasible to change Eastron Digital meter set-up using a general purpose Modbus Protocol master, but often easier to use the Eastron Digital meter display or Eastron Digital meter configurator software, especially for gaining password protected access. The Eastron Digital meter configurator software has facilities to store configurations to disk for later retrieval and rapid set up of similarly configured products.

Password

Some of the parameters described above are password protected and thus require the password to be entered at the Password register before they can be changed. The default password is 0000. When the password has been entered it will timeout in one minute unless the Password or Password Lock register is read to reset the timeout timer. Once the required changes have been made to the protected parameters the password lock should be reapplied by

a) allowing the password to timeout, or

b) writing any value to the Password Lock register, or

c) power cycling the instrument.

2 RS485 General Information

Some of the information in this section relates to other Eastron Digital meter product families, and is included to assist where a mixed network is implemented.RS485 or EIA (Electronic Industries Association) RS485 is a balanced line, half-duplex transmission system allowing transmission distances of up to 1.2 km. The following table summarizes the RS-485 Standard:

PARAMETER	
Mode of Operation	Differential

Number of Drivers and Receivers	32 Drivers, 32 Receivers
Maximum Cable Length	1200 m
Maximum Data Rate	10 M baud
Maximum Common Mode Voltage	12 V to –7 V
Minimum Driver Output Levels (Loaded)	+/– 1.5 V
Minimum Driver Output Levels (Unloaded)	+/– 6 V
Drive Load	Minimum 60 ohms
Driver Output Short Circuit Current Limit	150 mA to Gnd, 250 mA to 12 V 250 mA to –7 V
Minimum Receiver Input Resistance	12 kohms
Receiver Sensitivity	+/– 200 mV

Further information relating to RS485 may be obtained from either the EIA or the various RS485 device manufacturers, for example Texas Instruments or Maxim Semiconductors. This list is not exhaustive.

2.1 Half Duplex

Half duplex is a system in which one or more transmitters (talkers) can communicate with one or more receivers (listeners) with only one transmitter being active at any one time. For example, a “conversation” is started by asking a question, the person who has asked the question will then listen until he gets an answer or until he decides that the individual who was asked the question is not going to reply.

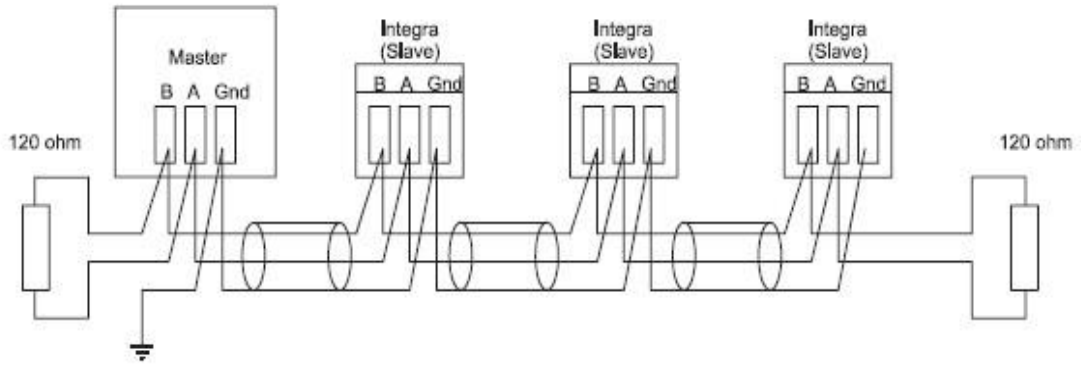
In a 485 network the “master” will start the “conversation” with a “query” addressed to a specific “slave”, the “master” will then listen for the “slave’s” response. If the “slave” does not respond within a pre-defined period, (set by control software in the “master”), the “master” will abandon the “conversation”.

2.2 Connecting the Instruments

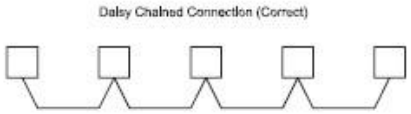
If connecting an RS485 network to a PC use caution if contemplating the use of an RS232 to 485 converter together with a USB to RS485 adapter. Consider either an RS232 to RS485 converter, connected directly to a suitable RS232 jack on the PC, or use a USB to RS485 converter or, for desktop PCs a suitable plug in RS485 card. (*Many 232:485 converters draw power from the RS232 socket. If using a USB to RS232 adapter, the adapter may not have enough power available to run the 232:485 converter.*)

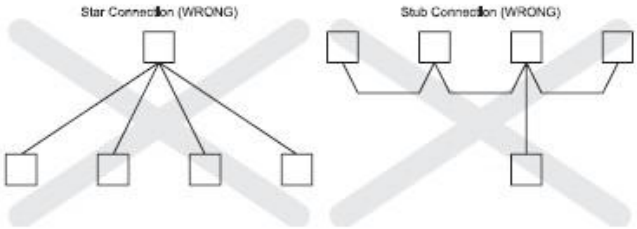
Screened twisted pair cable should be used. For longer cable runs or noisier environments, use of a cable specifically designed for RS485 may be necessary to achieve optimum performance. All “A” terminals should be connected together using one conductor of the twisted pair cable, all “B” terminals should be connected together using the other conductor in the pair. The cable screen should be connected to the “Gnd” terminals.

A Belden 9841 (Single pair) or 9842 (Two pair) or similar cable with a characteristic impedance of 120 ohms is recommended. The cable should be terminated at each end with a 120 ohm, quarter watt (or greater) resistor. Note: Diagram shows wiring topology only. Always follow terminal identification on Eastron Digital meter product label.



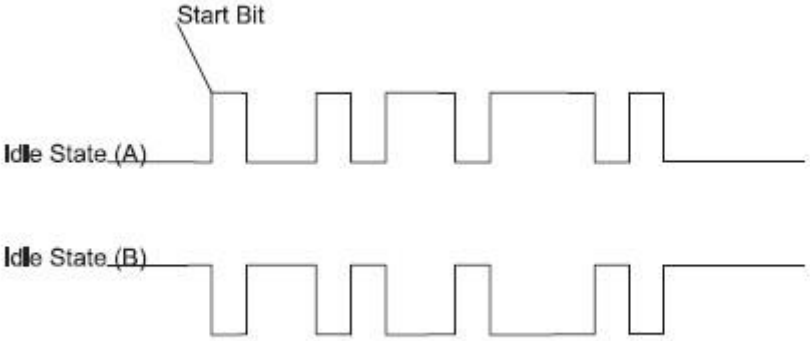
There must be no more than two wires connected to each terminal, this ensures that a “Daisy Chain or “straight line” configuration is used. A “Star” or a network with “Stubs (Tees)” is not recommended as reflections within the cable may result in data corruption.





2.3 A and B terminals

The A and B connections to the Eastron Digital meter products can be identified by the signals present on them whilst there is activity on the RS485 bus:



2.4 Troubleshooting

•Start with a simple network, one master and one slave. With Eastron Digital meter products this is easily achieved as the network can be left intact whilst individual instruments are disconnected by removing the RS485 connection from the rear of the instrument.

• Check that the network is connected together correctly. That is all of the “A’s” are connected together, and all of the “B’s” are connected together, and also that all of the “Gnd’s” are connected together.

• Confirm that the data “transmitted” onto the RS485 is not echoed back to the PC on the RS232 lines.(This facility is sometimes a link option within the converter). Many PC based packages seem to not perform well when they receive an echo of the message they are transmitting. SpecView and PCView (PC software) with a RS232 to RS485 converter are believed to include this feature.

• Confirm that the Address of the instrument is the same as the “master” is expecting.

• If the “network” operates with one instrument but not more than one check that each instrument has a unique address.

• Each request for data must be restricted to 40 parameters or less. Violating this requirement will impact the performance of the instrument and may result in a response time in excess of the specification.

• Check that the MODBUS Protocol mode (RTU or ASCII) and serial parameters (baud rate, number of data bits, number of stop bits and parity) are the same for all devices on the network.

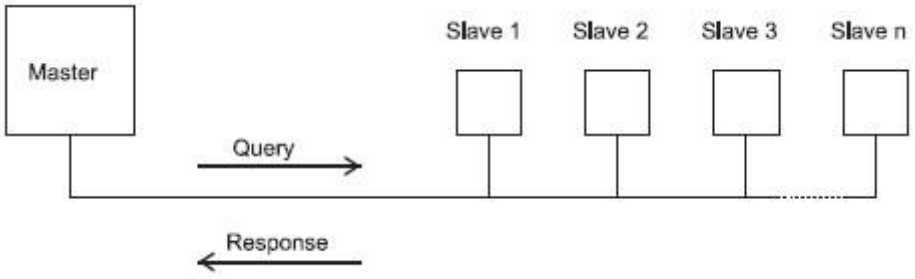
• Check that the “master” is requesting floating-point variables (pairs of registers placed on floating point boundaries) and is not “splitting” floating point variables.

• Check that the floating-point byte order expected by the “master” is the same as that used by Eastron Digital meter products. (PCView and Citect packages can use a number of formats including that supported by Eastron Digital meter).

• If possible obtain a second RS232 to RS485 converter and connect it between the RS485 bus and an additional PC equipped with a software package, which can display the data on the bus. Check for the existence of valid requests.

3 MODBUS Protocol General Information

Communication on a MODBUS Protocol Network is initiated (started) by a “Master” sending a query to a “Slave”. The “Slave“, which is constantly monitoring the network for queries addressed to it, will respond by performing the requested action and sending a response back to the ”Master”. Only the “Master” can initiate a query.



In the MODBUS Protocol the master can address individual slaves, or, using a special “Broadcast” address, can initiate a broadcast message to all slaves. The Eastron Digital meter do not support the broadcast address.

3.1 MODBUS Protocol Message Format

The MODBUS Protocol defines the format for the master’s query and the slave’s response. The query contains the device (or broadcast) address, a function code defining the requested action, any data to be sent, and an error-checking field.

The response contains fields confirming the action taken, any data to be returned, and an error-checking field. If an error occurred in receipt of the message then the message is ignored, if the slave is unable to perform the requested action, then it will construct an error message

and send it as its response. The MODBUS Protocol functions used by the Eastron Digital meters copy 16 bit register values between master and slaves. However, the data used by the Eastron Digital meter is in 32 bit IEEE 754 floating point format. Thus each instrument parameter is conceptually held in two adjacent MODBUS Protocol registers. Query
The following example illustrates a request for a single floating point parameter i.e. two 16-bit Modbus Protocol Registers.

First Byte Last Byte

Slave Address	Function Code	Start Address (Hi)	Start Address (Lo)	Number of Points (Hi)	Number of Points (Lo)	Number of Points (Lo)	Error Check (Lo)	Error Check (Hi)

Slave Address: 8-bit value representing the slave being addressed (1 to 247), 0 is reserved for the broadcast address. The Eastron Digital meters do not support the broadcast address.
Function Code: 8-bit value telling the addressed slave what action is to be performed. (3, 4, 8 or 16 are valid for Eastron Digital meter)
Start Address (Hi): The top (most significant) eight bits of a 16-bit number specifying the start address of the data being requested.
Start Address (Lo): The bottom (least significant) eight bits of a 16-bit number specifying the start address of the data being requested. As registers are used in pairs and start at zero, then this must be an even number.
Number of Points (Hi): The top (most significant) eight bits of a 16-bit number specifying the number of registers being requested.
Number of Points (Lo): The bottom (least significant) eight bits of a 16-bit number specifying the number of registers being requested. As registers are used in pairs, then this must be an even number.
Error Check (Lo): The bottom (least significant) eight bits of a 16-bit number representing the error check value.
Error Check (Hi): The top (most significant) eight bits of a 16-bit number representing the error check value.

Response
The example illustrates the normal response to a request for a single floating point parameter i.e. two 16-bit Modbus Protocol Registers.

First Byte Last Byte

Slave Address	Function Code	Byte Count	First Register (Hi)	First Register (Lo)	Second Register (Hi)	Second Register (Lo)	Error Check (Lo)	Error Check (Hi)

Slave Address: 8-bit value representing the address of slave that is responding.
Function Code: 8-bit value which, when a copy of the function code in the query, indicates that the slave recognised the query and has responded. (See also Exception Response).
Byte Count: 8-bit value indicating the number of data bytes contained within this response

First Register (Hi)*: The top (most significant) eight bits of a 16-bit number representing the first register requested in the query.
First Register (Lo)*: The bottom (least significant) eight bits of a 16-bit number representing the first register requested in the query.
Second Register (Hi)*: The top (most significant) eight bits of a 16-bit number representing the second register requested in the query.
Second Register (Lo)*: The bottom (least significant) eight bits of a 16-bit number representing the second register requested in the query.
Error Check (Lo): The bottom (least significant) eight bits of a 16-bit number representing the error check value.
Error Check (Hi): The top (most significant) eight bits of a 16-bit number representing the error check value.

*These four bytes together give the value of the floating point parameter requested.

Exception Response
If an error is detected in the content of the query (excluding parity errors and Error Check mismatch), then an error response (called an exception response), will be sent to the master. The exception response is identified by the function code being a copy of the query function code but with the most-significant bit set. The data contained in an exception response is a single byte error code.

First Byte				Last Byte
Slave Address	Function Code	Error Code	Error Check (Lo)	Error Check (Hi)

Slave Address: 8-bit value representing the address of slave that is responding.
Function Code: 8 bit value which is the function code in the query OR'ed with 80 hex, indicating that the slave either does not recognise the query or could not carry out the action requested.
Error Code: 8-bit value indicating the nature of the exception detected. (See “Table Of Exception Codes“ later).
Error Check (Lo): The bottom (least significant) eight bits of a 16-bit number representing the error check value.
Error Check (Hi): The top (most significant) eight bits of a 16-bit number representing the error check value.

3.2 Serial Transmission Modes
There are two MODBUS Protocol serial transmission modes, ASCII and RTU. Eastron Digital meters do not support the ASCII mode.
In RTU (Remote Terminal Unit) mode, each 8-bit byte is used in the full binary range and is not limited to ASCII characters as in ASCII Mode. The greater data density allows better data throughput for the same baud rate, however each message must be transmitted in a continuous stream. This is very unlikely to be a problem for modern communications equipment.

Coding System: Full 8-bit binary per byte. In this document, the value of each byte will be shown as two hexadecimal characters each in the range 0-9 or A-F.
Line Protocol: 1 start bit, followed by the 8 data bits. The 8 data bits are sent with least significant bit first.
User Option Of Parity No Parity and 2 Stop Bits
And Stop Bits: No Parity and 1 Stop Bit
Even Parity and 1 Stop Bit
Odd Parity and 1 Stop Bit.
User Option of Baud 2400; 4800 ; 9600 ; 19200 ; 38400

The baud rate, parity and stop bits must be selected to match the master's settings.

3.3 MODBUS Protocol Message Timing (RTU Mode)

A MODBUS Protocol message has defined beginning and ending points. The receiving devices recognizes the start of the message, reads the "Slave Address" to determine if they are being addressed and knowing when the message is completed they can use the Error Check bytes and parity bits to confirm the integrity of the message. If the Error Check or parity fails then the message is discarded.
In RTU mode, messages starts with a silent interval of at least 3.5 character times.
The first byte of a message is then transmitted, the device address.
Master and slave devices monitor the network continuously, including during the 'silent' intervals. When the first byte (the address byte) is received, each device checks it to find out if it is the addressed device. If the device determines that it is the one being addressed it records the whole message and acts accordingly, if it is not being addressed it continues monitoring for the next message.
Following the last transmitted byte, a silent interval of at least 3.5 character times marks the end of the message. A new message can begin after this interval.
In the Eastron 1000 and 2000, a silent interval of 60msec minimum is required in order to guarantee successful reception of the next request.
The entire message must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the message, the receiving device flushes the incomplete message and assumes that the next byte will be the address byte of a new message.
Similarly, if a new message begins earlier than 3.5 character times following a previous message, the receiving device may consider it a continuation of the previous message. This will result in an error, as the value in the final CRC field will not be valid for the combined messages.

3.4 How Characters are Transmitted Serially

When messages are transmitted on standard MODBUS Protocol serial networks each byte is sent in this order (left to right):
Transmit Character = Start Bit + Data Byte + Parity Bit + 1 Stop Bit (11 bits total):
Least Significant Bit (LSB) Most Significant Bit (MSB)

Start	1	2	3	4	5	6	7	8	Party	Stop

Transmit Character = Start Bit + Data Byte + 2 Stop Bits (11 bits total):

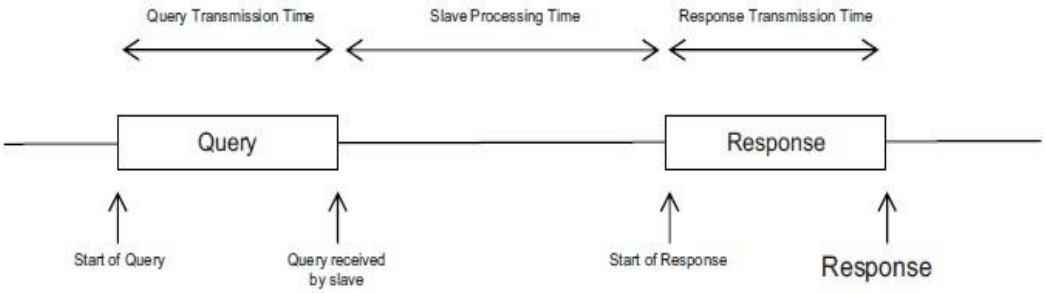
Start	1	2	3	4	5	6	7	8	Stop	Stop

Eastron Digital meters additionally support No parity, One stop bit.

Transmit Character = Start Bit + Data Byte + 1 Stop Bit (10 bits total):

Start	1	2	3	4	5	6	7	8	Stop

The master is configured by the user to wait for a predetermined timeout interval. The master will wait for this period of time before deciding that the slave is not going to respond and that the transaction should be aborted. Care must be taken when determining the timeout period from both the master and the slaves' specifications. The slave may define the 'response time' as being the period from the receipt of the last bit of the query to the transmission of the first bit of the response. The master may define the 'response time' as period between transmitting the first bit of the query to the receipt of the last bit of the response. It can be seen that message transmission time, which is a function of the baud rate, must be included in the timeout calculation.



3.5 Error Checking Methods

Standard MODBUS Protocol serial networks use two error checking processes, the error check bytes mentioned above check message integrity whilst Parity checking (even or odd) can be applied to each byte in the message.

3.5.1 Parity Checking

If parity checking is enabled – by selecting either Even or Odd Parity - the quantity of "1's" will be counted in the data portion of each transmit character. The parity bit will then be set to a 0 or 1 to result in an Even or Odd total of "1's".

Note that parity checking can only detect an error if an odd number of bits are picked up or dropped in a transmit character during transmission, if for example two 1's are corrupted to 0's the parity check will not find the error.

If No Parity checking is specified, no parity bit is transmitted and no parity check can be made. Also, if No Parity checking is specified and one stop bit is selected the transmit character is

effectively shortened by one bit.

3.5.2 CRC Checking

The error check bytes of the MODBUS Protocol messages contain a Cyclical Redundancy Check (CRC) value that is used to check the content of the entire message. The error check bytes must always be present to comply with the MODBUS Protocol, there is no option to disable it.

The error check bytes represent a 16-bit binary value, calculated by the transmitting device. The receiving device must recalculate the CRC during receipt of the message and compare the calculated value to the value received in the error check bytes. If the two values are not equal, the message should be discarded.

The error check calculation is started by first pre-loading a 16-bit register to all 1's (i.e. Hex (FFFF)) each successive 8-bit byte of the message is applied to the current contents of the register. Note: only the eight bits of data in each transmit character are used for generating the CRC, start bits, stop bits and the parity bit, if one is used, are not included in the error check bytes.

During generation of the error check bytes, each 8-bit message byte is exclusive OR'ed with the lower half of the 16 bit register. The register is then shifted eight times in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. After each shift the LSB prior to the shift is extracted and examined. If the LSB was a 1, the register is then exclusive OR'ed with a pre-set, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until all eight shifts have been performed. After the last shift, the next 8-bit message byte is exclusive OR'ed with the lower half of the 16 bit register, and the process repeated. The final contents of the register, after all the bytes of the message have been applied, is the error check value. In the following pseudo code "Error Word" is a 16-bit value representing the error check values.

```
BEGIN
        Error Word = Hex (FFFF)
        FOR Each byte in message
                Error Word = Error Word XOR byte in message
                FOR Each bit in byte
                        LSB = Error Word AND Hex (0001)
                        IF LSB = 1 THEN Error Word = Error Word – 1
                        Error Word = Error Word / 2
                        IF LSB = 1 THEN Error Word = Error Word XOR Hex (A001)
                 NEXT bit in byte
NEXT Byte in message
END
```

3.6 Function Codes

The function code part of a MODBUS Protocol message defines the action to be taken by the slave. Eastron Digital meters support the following function codes:

Code	MODBUS Protocol name	Description
03	Read Holding Registers	Read the contents of read/write location(4X references)
04	Read Input Registers	Read the contents of read only location(3X references)
08	Diagnostics	Only sub-function zero is supported. This returns the data element of the query unchanged.
15	Pre-set Multiple Registers	Set the contents of read/write location (4X references)

3.7 IEEE floating point format

The MODBUS Protocol defines 16 bit “Registers” for the data variables. A 16-bit number would prove too restrictive, for energy parameters for example, as the maximum range of a 16-bit number is 65535.

However, there are a number of approaches that have been adopted to overcome this restriction. Eastron Digital meters use two consecutive registers to represent a floating-point number, effectively expanding the range to +/- $1x10^{37}$.

The values produced by Eastron Digital meters can be used directly without any requirement to “scale” the values, for example, the units for the voltage parameters are volts, the units for the power parameters are watts etc.

What is a floating point Number?

A floating-point number is a number with two parts, a mantissa and an exponent and is written in the form 1.234×10^5. The mantissa (1.234 in this example) must have the decimal point moved to the right with the number of places determined by the exponent (5 places in this example) i.e. $1.234x\ 10^5 = 123400$. If the exponent is negative the decimal point is moved to the left.

What is an IEEE 754 format floating-point number?

An IEEE 754 floating point number is the binary equivalent of the decimal floating-point number shown above. The major difference being that the most significant bit of the mantissa is always arranged to be 1 and is thus not needed in the representation of the number. The process by which the most significant bit is arranged to be 1 is called normalization, the mantissa is thus referred to as a “normal mantissa”. During normalization the bits in the mantissa are shifted to the left whilst the exponent is decremented until the most significant bit of the mantissa is one. In the special case where the number is zero both mantissa and exponent are zero.

The bits in an IEEE 754 format have the following significance:

Data Hi Reg, Hi Byte.	Data Hi Reg, Lo Byte.	Data Lo Reg, Hi Byte.	Data Lo Reg, Lo Byte.
SEEE EEEE	EMMM MMMM	MMMM MMMM	MMMM MMMM

Where:
S represents the sign bit where 1 is negative and 0 is positive
E is the 8-bit exponent with an offset of 127 i.e. an exponent of zero is represented by 127, an exponent of 1 by 128 etc.
M is the 23-bit normal mantissa. The 24th bit is always 1 and, therefore, is not stored.

Using the above format the floating point number 240.5 is represented as 43708000 hex:

Data Hi Reg, Hi Byte	Data Hi Reg, Lo Byte	Data Lo Reg, Hi Byte	Data Lo Reg, Lo Byte
43	**70**	**80**	**00**

The following example demonstrates how to convert IEEE 754 floating-point numbers from their hexadecimal form to decimal form. For this example, we will use the value for 240.5 shown above
Note that the floating-point storage representation is not an intuitive format. To convert this value to decimal, the bits should be separated as specified in the floating-point number storage format table shown above.
For example:

Data Hi Reg, Hi Byte	Data Hi Reg, Lo Byte	Data Lo Reg, Hi Byte	Data Lo Reg, Lo Byte
0100 0011	**0111 0000**	**1000 0000**	**0000 0000**

From this you can determine the following information.
• The sign bit is 0, indicating a positive number.
• The exponent value is 10000110 binary or 134 decimal. Subtracting 127 from 134 leaves 7, which is the actual exponent.
• The mantissa appears as the binary number 11100001000000000000000
There is an implied binary point at the left of the mantissa that is always preceded by a 1. This bit is not stored in the hexadecimal representation of the floating-point number. Adding 1 and the binary point to the beginning of the mantissa gives the following:
1.11100001000000000000000
Now, we adjust the mantissa for the exponent. A negative exponent moves the binary point to the left. A positive exponent moves the binary point to the right. Because the exponent is 7, the mantissa is adjusted as follows:
11110000.1000000000000000
Finally, we have a binary floating-point number. Binary bits that are to the left of the binary point represent
the power of two corresponding to their position. For example, 11110000 represents $(1 \times 2^7) + (1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 240$.
Binary bits that are to the right of the binary point also represent a power of 2 corresponding to their position. As the digits are to the right of the binary point the powers are negative. For example: .100 represents $(1 \times 2^{-1}) + (0 \times 2^{-2}) + (0 \times 2^{-3}) + \ldots$ which equals 0.5.
Adding these two numbers together and making reference to the sign bit produces the number +240.5.

For each floating point value requested two MODBUS Protocol registers (four bytes) must be requested. The received order and significance of these four bytes for Eastron Digital meters is shown below:

Data Hi Reg, Hi Byte	Data Hi Reg, Lo Byte	Data Lo Reg, Hi Byte	Data Lo Reg, Lo Byte

3.8 MODBUS Protocol Commands supported

All Eastron Digital meters support the “Read Input Register” (3X registers), the “Read Holding Register” (4X registers) and the “Pre-set Multiple Registers” (write 4X registers) commands of the MODBUS Protocol RTU protocol. All values stored and returned are in floating point format to IEEE 754 with the most significant register first.

3.8.1 Read Input Registers

MODBUS Protocol code 04 reads the contents of the 3X registers.

Example

The following query will request ‘Volts 1’ from an instrument with node address 1:

Field Name	Example(Hex)
Slave Address	01
Function	04
Starting Address High	00
Starting Address Low	00
Number of Points High	00
Number of Points Low	02
Error Check Low	71
Error Check High	CB

Note: Data must be requested in register pairs i.e. the “Starting Address“ and the “Number of Points” must be even numbers to request a floating point variable. If the “Starting Address” or the “Number of points” is odd then the query will fall in the middle of a floating point variable the product will return an error message.

The following response returns the contents of Volts 1 as 230.2. But see also “Exception Response” later.

Field Name	Example (Hex)
Slave Address	01
Function	04
Byte Count	04
Data, High Reg, High Byte	43
Data, High Reg, Low Byte	66
Data, Low Reg, High Byte	33
Data, Low Reg, Low Byte	34
Error Check Low	1B
Error Check High	38

3.9 Holding Registers

3.9.1 Read Holding Registers

MODBUS Protocol code 03 reads the contents of the 4X registers.

Example

The following query will request the prevailing 'Demand Time':

Field Name	Example (Hex)
Slave Address	01
Function	03
Starting Address High	00
Starting Address Low	00
Number of Points High	00
Number of Points Low	02
Error Check Low	C4
Error Check High	0B

Note: Data must be requested in register pairs i.e. the "Starting Address" and the "Number of Points" must be even numbers to request a floating point variable. If the "Starting Address" or the "Number of points" is odd then the query will fall in the middle of a floating point variable the product will return an error message.

The following response returns the contents of Demand Time as 1, But see also "Exception Response" later.

Field Name	Example (Hex)
Slave Address	01
Function	03
Byte Count	04
Data, High Reg, High Byte	3F
Data, High Reg, Low Byte	80
Data, Low Reg, High Byte	00
Data, Low Reg, Low Byte	00
Error Check Low	F7
Error Check High	CF

3.9.2 Write Holding Registers

MODBUS Protocol code 10 (16 decimal) writes the contents of the 4X registers.

Example

The following query will set the Demand Period to 60, which effectively resets the Demand Time:

Field Name	Example (Hex)
Slave Address	01
Function	10
Starting Address High	00
Starting Address Low	02

Number of Registers High	00
Number of Registers Low	02
Byte Count	04
Data, High Reg, High Byte	42
Data, High Reg, Low Byte	70
Data, Low Reg, High Byte	00
Data, Low Reg, Low Byte	00
Error Check Low	67
Error Check High	D5

Note: Data must be written in register pairs i.e. the “Starting Address“ and the “Number of Points” must be even numbers to write a floating point variable. If the “Starting Address” or the “Number of points” is odd then the query will fall in the middle of a floating point variable the product will return an error message. In general only one floating point value can be written per query

The following response indicates that the write has been successful. But see also “Exception Response”later.

Field Name	**Example (Hex)**
Slave Address	01
Function	10
Starting Address High	00
Starting Address Low	02
Number of Registers High	00
Number of Registers Low	02
Error Check Low	E0
Error Check High	08

3.10 Exception Response

If the slave in the “Write Holding Register” example above, did not support that function then it would have replied with an Exception Response as shown below. The exception function code is the original function code from the query with the MSB set i.e. it has had 80 hex logically ORed with it. The exception code indicates the reason for the exception. The slave will not respond at all if there is an error with the parity or CRC of the query. However, if the slave can not process the query then it will respond with an exception. In this case a code 01, the requested function is not support by this slave.

Field Name	**Example (Hex)**
Slave Address	01
Function	10 OR 80 = 90
Exception Code	01
Error Check Low	8D
Error Check High	C0

3.11 Exception Codes

3.11.1 Table of Exception Codes

Eastron Digital meters support the following function codes:

Exception Code	MODBUS Protocol name	Description
01	Illegal Function	The function code is not supported by the product
02	Illegal Data Address	Attempt to access an invalid address or an attempt to read or write part of a floating point value
03	Illegal Data Value	Attempt to set a floating point variable to an invalid value
05	Slave Device Failure	An error occurred when the instrument attempted to store an update to it's configuration

3.12 Diagnostics

MODBUS Protocol code 08 provides a number of diagnostic sub-functions. Only the "Return Query Data" sub-function (sub-function 0) is supported on Eastron Digital meters.

Example

The following query will send a diagnostic "return query data" query with the data elements set to Hex(AA) and Hex(55) and will expect these to be returned in the response:

Field Name	Example (Hex)
Slave Address	01
Function	08
Sub-Function High	00
Sub-Function Low	00
Data Byte 1	AA
Data Byte 2	55
Error Check Low	5E
Error Check High	94

Note: Exactly one register of data (two bytes) must be sent with this function.

The following response indicates the correct reply to the query, i.e. the same bytes as the query.

Field Name	Example (Hex)
Slave Address	01
Function	08
Sub-Function High	00
Sub-Function Low	00
Data Byte 1	AA
Data Byte 2	55
Error Check Low	5E
Error Check High	94

Byte 0 ~byte 3	Unix 时间戳	
Byte 4~Byte7	data type	0x01: Line loss mode change 0x02: Impedance Change 0x03: User Mode Change 0x04: Charging identification point change 0x05: Out of memory 0x06: Firmware Update 0x07: Charging distribution point change 0x0A: Charging station firmware change 0x0B: Line system change 0x0C: Measurement value damaged 0x0D: Internal module function damaged
Byte 8	Current time zone	-12~12
Byte9~byte12	The current system type	0x01:1P2W 0x02:3P3W 0x03:3P4W 0x04:1P3W
Byte13~byte16	The current line loss mode	0x00：off 0x01: on
Byte17~byte20	Current user mode	0x00:normal 0x01:Assembly Mode
Byte 21~Byte22	The current line resistance	1~999 *
Byte23~byte26	Current CT	0x00:CHARGE_POINT_NULL 0x01:EVSEID , 0x02:CBIDC
Byte27~ byte 52	Current CI	ASCII
Byte53~byte77	Current CF	ASCII
Byte 78~byte 103	Change original data	This data will change its content according to its data type. The data format corresponds to the format of the data content. For example, when the data type is 0x01 and the line loss mode is changed, This field is parsed as:

		0x00: Not enabled 0x01: Enable line loss
Byte 104~ byte 167	Signature	The first byte of the signature data is the header, and the second byte is the length

appendix 1

Appendix 2

BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT9	BIT8
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
		Public key QR code	TT	CF	Prat ID	CI	Time and energy