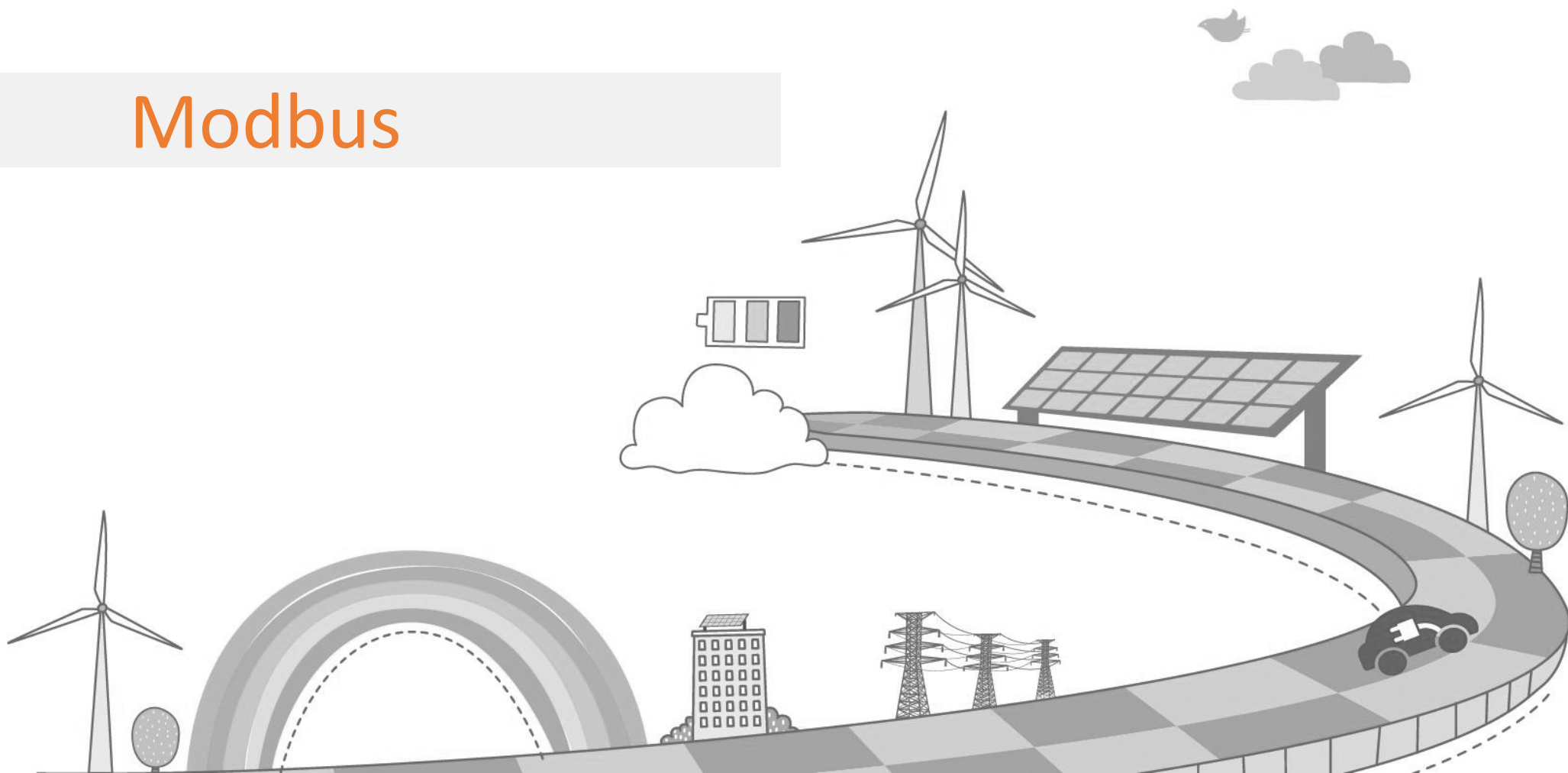


SUNGROW

EMEA COE Technical Team

让人人享用清洁电力
Clean power for all

Modbus



Modbus (Wikipedia)

Modbus is a data communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). Modbus has become a de facto standard communication protocol and is now a commonly available means of connecting industrial electronic devices.

Modbus is popular in industrial environments because it is openly published and royalty-free. It was developed for industrial applications, is relatively easy to deploy and maintain compared to other standards, and places few restrictions - other than the datagram (packet) size - on the format of the data to be transmitted.

The Modbus protocol uses character serial communication lines, Ethernet, or the Internet protocol suite as a transport layer.

Modbus supports communication to and from multiple devices connected to the same cable or Ethernet network. For example, there can be a device that measures temperature and another device to measure humidity connected to the same cable, both communicating measurements to the same computer.

Modbus

1. Free standard
2. Support RS-232, RS-485, Ethernet, Wireless and so on
3. Simple frame format, easy to learn and develop

Modbus

Modbus RTU: using 8 bit (byte) and 16 bit (2 bytes) data

Modbus ASCII: change all 8 bit data into ASCII code

RTU: transfer more data than ASCII mode. (CRC: Cyclic Redundancy Check)

ASCII: less communication error. (LRC: Longitudinal Redundancy Check)

Sungrow selects Modbus RTU



Modbus

01 Hex Data

02 Modbus Protocol

03 Sungrow ModbusProtocol



Hex Data

01

Hex Data

Hex Data

In all transmission medium we use binary:

decimal	0	1	2	3	4	5	6	7	8
binary	0	1	10	11	100	101	110	111	1000

Electric circuit can easily implement binary transmission, using „open“ and „close“ for „0“ and „1“

Hex Data

In order to make the data easier to understand, we use Hexadecimal

$$0_{\text{hex}} = 0_{\text{dec}} = 0_{\text{oct}} \quad 0 \ 0 \ 0 \ 0$$

$$1_{\text{hex}} = 1_{\text{dec}} = 1_{\text{oct}} \quad 0 \ 0 \ 0 \ 1$$

$$2_{\text{hex}} = 2_{\text{dec}} = 2_{\text{oct}} \quad 0 \ 0 \ 1 \ 0$$

$$3_{\text{hex}} = 3_{\text{dec}} = 3_{\text{oct}} \quad 0 \ 0 \ 1 \ 1$$

$$4_{\text{hex}} = 4_{\text{dec}} = 4_{\text{oct}} \quad 0 \ 1 \ 0 \ 0$$

$$5_{\text{hex}} = 5_{\text{dec}} = 5_{\text{oct}} \quad 0 \ 1 \ 0 \ 1$$

$$6_{\text{hex}} = 6_{\text{dec}} = 6_{\text{oct}} \quad 0 \ 1 \ 1 \ 0$$

$$7_{\text{hex}} = 7_{\text{dec}} = 7_{\text{oct}} \quad 0 \ 1 \ 1 \ 1$$

$$8_{\text{hex}} = 8_{\text{dec}} = 10_{\text{oct}} \quad 1 \ 0 \ 0 \ 0$$

$$9_{\text{hex}} = 9_{\text{dec}} = 11_{\text{oct}} \quad 1 \ 0 \ 0 \ 1$$

$$A_{\text{hex}} = 10_{\text{dec}} = 12_{\text{oct}} \quad 1 \ 0 \ 1 \ 0$$

$$B_{\text{hex}} = 11_{\text{dec}} = 13_{\text{oct}} \quad 1 \ 0 \ 1 \ 1$$

$$C_{\text{hex}} = 12_{\text{dec}} = 14_{\text{oct}} \quad 1 \ 1 \ 0 \ 0$$

$$D_{\text{hex}} = 13_{\text{dec}} = 15_{\text{oct}} \quad 1 \ 1 \ 0 \ 1$$

$$E_{\text{hex}} = 14_{\text{dec}} = 16_{\text{oct}} \quad 1 \ 1 \ 1 \ 0$$

$$F_{\text{hex}} = 15_{\text{dec}} = 17_{\text{oct}} \quad 1 \ 1 \ 1 \ 1$$

Hex Data

Maximum: $2^n - 1$

1 bit: 1

2 bit: 3

4 bit: 15 (0xF)

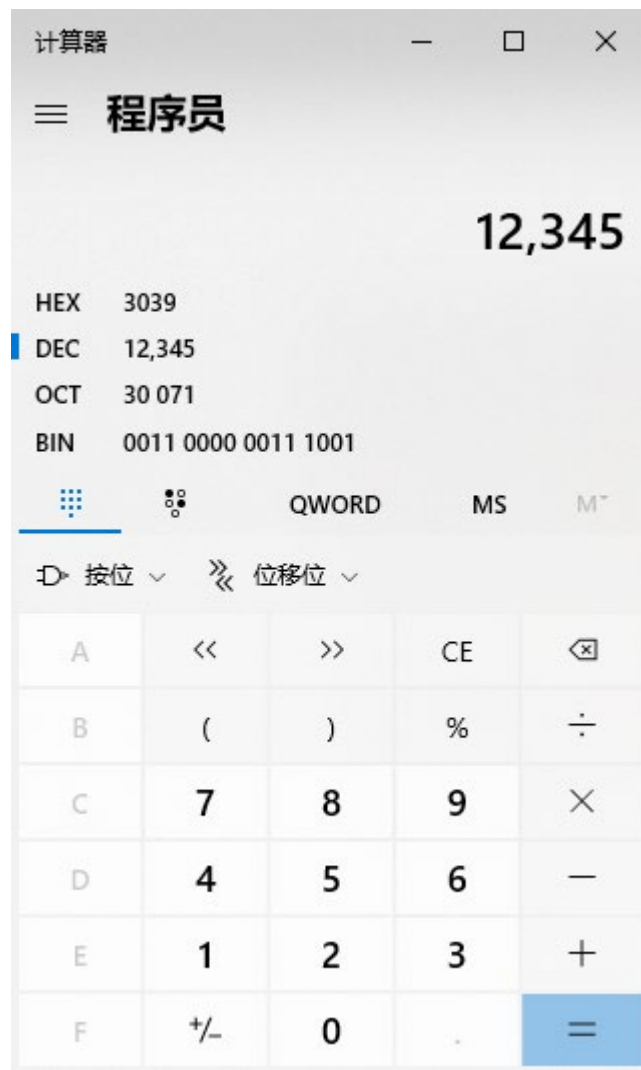
8 bit (1 byte): 255 (0xFF)

16 bit (2 byte): 65535 (0xFFFF)

32 bit (4 byte): 4,294,967,295 (0xFFFF FFFF)

...

Hex Data



Calculator in Windows is a very useful tool

We can convert the data in “Programmer”

The default BIN data of Calculator is 64 bit, We cannot use it directly for negative number

Hex Data

Negative value minimum:

16 bit: 1000 0000 0000 0000, 0x8000, -32768

32 bit: 1000 0000 ... 0000 0000, 0x8000 0000, -2147483648

Negative value maximum:

16 bit: 1111 1111 1111 1111, 0xFFFF, -1

32 bit: 1111 1111 ... 1111 1111, 0xFFFF FFFF, -1

To calculate the negative hex value x:

$0xFFFF (0xFFFF FFFF) + 1 + x$

for example -20 into 16bit:

$0xFFFF + 1 - 20 = 0xFFFF - 19 = 65535 - 19 = 65516 \Rightarrow 0xFFEC$

Hex Data

Float 32 bit

Only be used for 3rd party device, and communicated through sungrow logger

First 9 bit for sign and decimal point position, last 23 bit as mantissa, so float value cannot be simply converted by calculator

<https://www.h-schmidt.net/FloatConverter/IEEE754.html>

IEEE 754 Converter (JavaScript), V0.22

	Sign	Exponent	Mantissa
Value:	-1	2^0	1.1233999729156494
Encoded as:	1	127	1035154
Binary:	<input checked="" type="checkbox"/>	<input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
You entered	<input type="text" value="-1.1234"/>		
Value actually stored in float:	<input type="text" value="-1.1233999729156494140625"/>		
Error due to conversion:	<input type="text" value="2.70843505859375E-8"/>		
Binary Representation	<input type="text" value="10111111100011111100101110010010"/>		
Hexadecimal Representation	<input type="text" value="0xbf8fcb92"/>		

Hex Data

Exercise

Please turn 5000 (decimal) into a hex value

Please turn 0x1B57 (Hex) into a decimal value

Please turn -1000 (decimal) into a 32 bit hex value

Please turn 0xD8F0 into a decimal value.

Hex Data

Exercise

Please turn 5000 (decimal) into a hex value 0x1388

Please turn 0x1B57 (Hex) into a decimal value 0x6999

Please turn -1000 (decimal) into a 32 bit hex value 0xFC18

Please turn 0xD8F0 into a decimal value. -10000



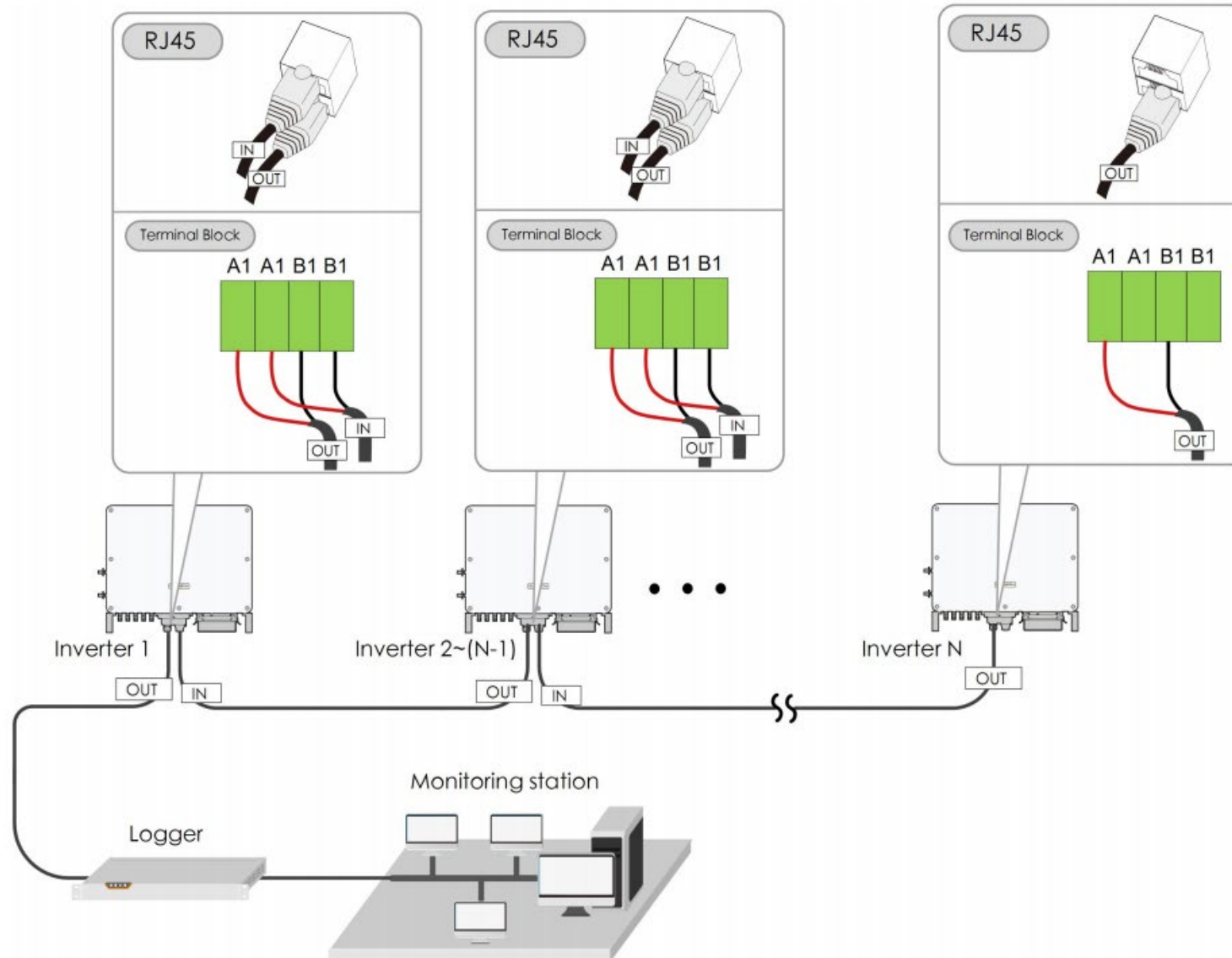
02

**Modbus
Protocol**

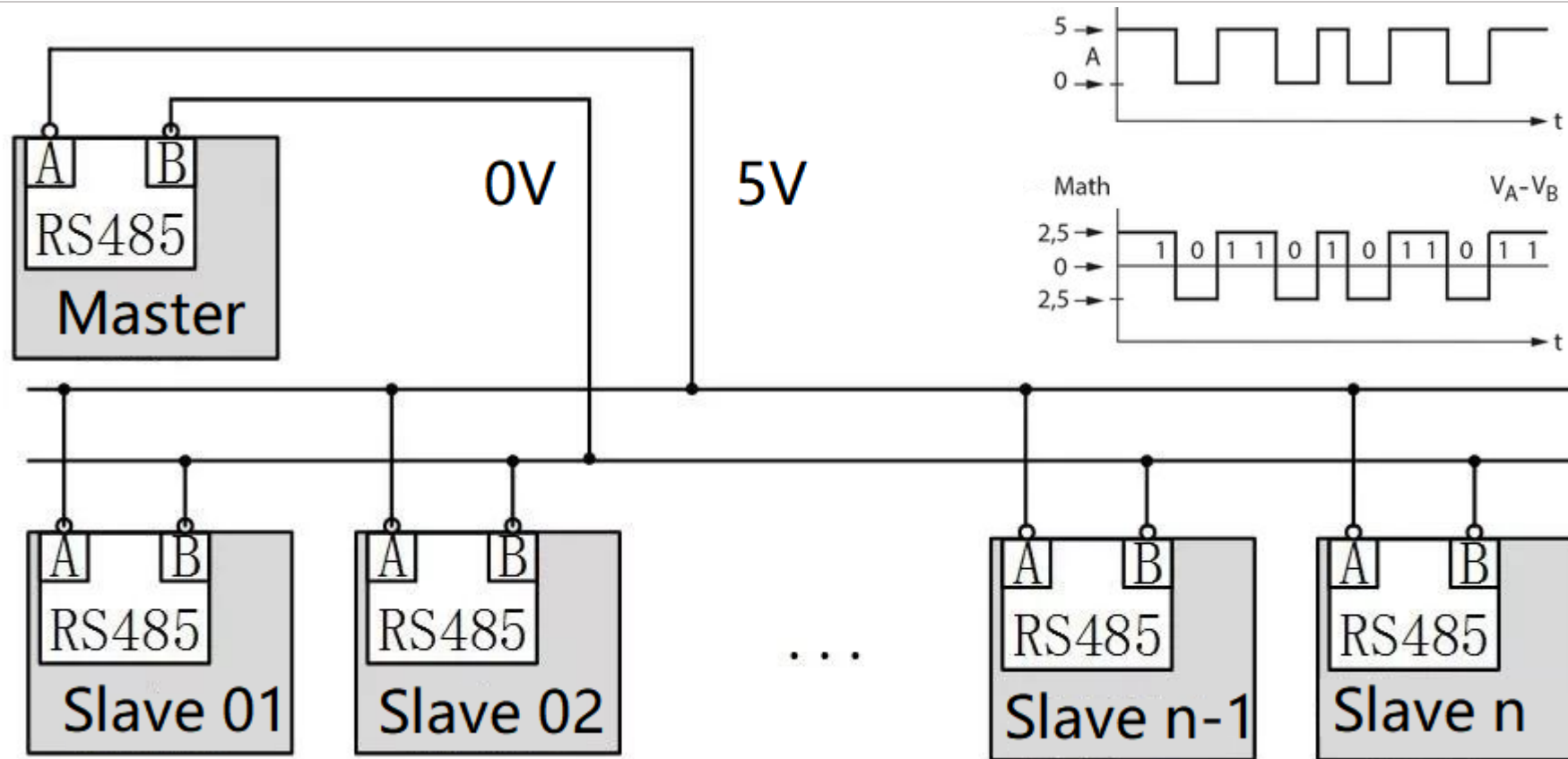
Modbus Protocol

Slave: Inverter
One or more slaves

Master: Logger
Only one Master



Modbus Protocol



In RS485 bus there is voltage signal. Sungrow Device has 5V voltage between A and B when there is no signal. With signal multimeter will read 2 - 3 V between A and B

DC voltage will be converted into digital 0 and 1 according to voltage and baudrate (bit/s).

Normally program will convert the digital signal into hex message, for example:

01 04 13 87 00 01

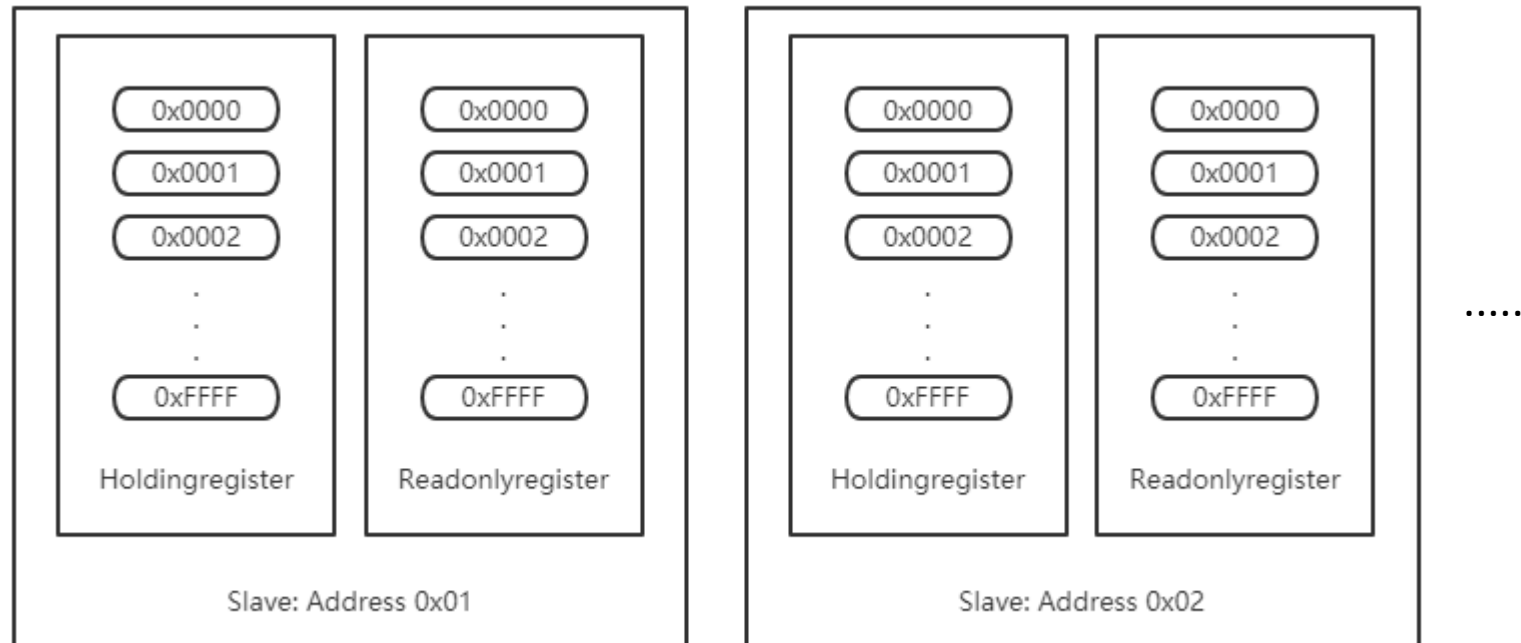
Modbus Protocol

```
[2021/03/04 15:00:45.319 COM3 Send:] 00 06 13 8F 00 00 BD 74
[2021/03/04 15:00:46.111] No answer!
[2021/03/04 15:02:40.936 COM3 Send:] 00 06 13 8F 03 20 BC 5C
[2021/03/04 15:02:41.706] No answer!
[2021/03/04 15:40:55.876 COM3 Send:] 00 06 13 8F 03 20 BC 5C
[2021/03/04 15:40:56.664] No answer!
[2021/03/04 15:41:38.163 COM3 Send:] 00 06 13 8F 00 64 BC 9F
[2021/03/04 15:41:38.942] No answer!
[2021/03/04 15:41:39.805 COM3 Send:] 00 06 13 8F 00 64 BC 9F
[2021/03/04 15:41:40.573] No answer!
[2021/03/04 15:43:04.998 COM3 Send:] 04 03 13 8F 00 01 B1 30
[2021/03/04 15:43:05.374 COM3 Receive:] 04 03 02 00 64 75 AF
[2021/03/04 15:43:30.576 COM3 Send:] 00 06 13 8F 00 FF FD 34
[2021/03/04 15:43:31.375] No answer!
[2021/03/04 15:43:49.612 COM3 Send:] 04 03 13 8F 00 01 B1 30
[2021/03/04 15:43:49.990 COM3 Receive:] 04 03 02 00 FF 34 04
[2021/03/04 15:44:40.348 COM3 Send:] 04 03 13 8F 00 01 B1 30
[2021/03/04 15:44:40.750 COM3 Receive:] 04 03 02 00 FF 34 04
[2021/03/04 15:45:08.937 COM3 Send:] 00 06 13 8F 03 E8 BD CA
```

Target:

- to understand the message in RS485 bus,
- and to be able to create correct message,
- and find the error of wrong message.

Modbus Protocol



Each slave device has a unique address (Modbus ID), range 1 - 255 (0x01 - 0xFF)

Normally Address 0 means broadcast, to all slave device.

Each slave device has several types of **register**. Normally Sungrow will use holdingregister and readonlyregister. Holdingregister can be read and written, readonlyregister can only be read.

Each type of register has register address from 0x0000 to 0xFFFF

Register is like a memory, saving data. Master reads slave register to get its information, and writes slave register to send value or command. Register will be defined by device supplier (like Sungrow).

All modbus messages are operating slave register.

Modbus Protocol

Several concept:

1. Address (Modbus ID), only for Slave Devices
2. Register, the momory of Slave Devices, always operating register!
3. Register address, different from Modbus ID!
4. Holdingregister, read/write
5. Readonlyregister, readonly
6. **Modbus communication: Master operates the register of slave devices. (Slave will reply or execute the command from Master, Slave will never proactively send message!)**

Modbus Protocol

Modbus RTU has fixed frame format (primarily used on asynchronous serial data lines like RS-485/EIA-485)
Start and End will be recognized by system automatically.

Normally we need to edit the:

Address + Function + Data + CRC

Most software is able to calculate CRC automatically

Name	Length (bits)	Function
Start	28	At least 3½ character times of silence (mark condition)
Address	8 (1 byte)	Station address
Function	8 (1 byte)	Indicates the function code; e.g., read coils/holding registers
Data	n x 8 (n bytes)	Data + length will be filled depending on the message type
CRC	16 (2 bytes)	Cyclic redundancy check
End	28	At least 3½ character times of silence between frames

Modbus Protocol

04 03 13 8F 00 01 B1 30

Analyse:

04 : Address (Modbus ID of device)

03 : Function (Read holding register)

13 8F 00 01 : Data

B1 30: CRC

In Frame of Modbus RTU in RS485, the position and length of address, function, CRC are always fixed. We need to encode Data according to function.

Modbus Protocol

Sungrow use normally only 4 function code:

- 0x03: read holdingregister
- 0x04: read readonlyregister
- 0x06: write one holdingregister
- 0x10: write one or more holdingregisters

Error, when master send invalid command to slave, slave will reply with error message, following function code will only be send by slave devices, adding 0x80 to the original function code:

- 0x83: invalid command with 0x03 function
- 0x84: invalid command with 0x04 function
- 0x86: invalid command with 0x06 function
- 0x90: invalid command with 0x10 function

Modbus Protocol

Master sends:

ID	Function	Register address	Register quantity	CRC
01	03	1B 57	00 10	F3 32
05	04	13 87	00 20	44 FB

Slave answers:

ID	Function	quantity of Bytes	Value	CRC
01	03	00 20	00 01 02 03 ... (totally 20 bytes, 10 registers)	(CRC)
05	04	00 40	FF FE FD 01 ... (totally 40 bytes, 20 registers)	(CRC)

Please make sure which type of register you want to read, holdingregister (03) or readonlyregister (04)?

The value will be according to real slave devices.

CRC will be calculated according all above hex number.

The rest message in answer, like ID, Function, Quantity of Bytes must match strictly the command from master.

Modbus Protocol

Master sends:

ID	Function	Register address	Value	CRC
01	06	13 87	07 E5	FE DC

Slave answers:

ID	Function	Register address	Value	CRC
01	06	13 87	07 E5	FE DC

Function 06 will write only one 16 bit register, so after the register address will follow only the 16 bit value. The answer will be a copy of the command.

If ID = 00, broadcast, slave will not answer!

Modbus Protocol

Master sends:

ID	Function	Register address	Quantity of Register	Quantity of Bytes	Value	CRC
01	10	13 87	00 03	06	07 E5 00 07 00 09	36 90

Slave answers:

ID	Function	Register address	Quantity of register	CRC
01	10	13 87	00 03	FE DC

Function 10 will write more register, the value should be input one by one from the first register.

If ID = 00, broadcast, slave will not answer!

Modbus Protocol

Slave replies error message:

ID	Function	Error Info	CRC
01	83	01	CRC
03	84	02	CRC
05	86	03	CRC
07	90	04	CRC

Error Codes	Name
0x0001	ILLEGAL FUNCTION
0x0002	ILLEGAL DATA ADDRESS
0x0003	ILLEGAL DATA VALUE
0x0004	SLAVE DEVICE FAILURE

When there is error in command from master, or the command does not match the slave protocol, slave device will reply with error message.

Error message function code = 0x80 + original function code.

Modbus Protocol

Modbus TCP frame format (primarily used on [Ethernet networks](#)) [\[edit \]](#)

Name	Length (bytes)	Function
Transaction identifier	2	For synchronization between messages of server and client
Protocol identifier	2	0 for Modbus/TCP
Length field	2	Number of remaining bytes in this frame
Unit identifier	1	Slave address (255 if not used)
Function code	1	Function codes as in other variants
Data bytes	<i>n</i>	Data as response or commands

Example:

Original Modbus RTU: 01 04 13 87 00 01 **85 67**

Modbus TCP: 00 00 00 00 00 06 01 04 13 87 00 01

Reply : 00 00 00 00 00 05 01 04 02 12 34

Transaction identifier and protocol identifier will be copied into reply.

Modbus TCP has no CRC, but more 6 bytes before main Modbus RTU message

Modbus Protocol

Exercise:

Address 5 for all

- 1) Create a command, to read readonlyregister 7058 - 7081
- 2) Create a command, to read holdingregister 4999 - 5004
- 3) Create a command, to write register 5018 as 800
- 4) Create a command, to write register 5006 as 170, 5007 as 900

Modbus Protocol

Exercise:

Are the message from Master or from Slave?

01 04 02 2C 02 24 31

30 06 13 8F 03 E8 B8 3A

11 04 1A 01 56 01 5B 01 64 01 66 01 70 01 60 01 89 01 84 01 92 01 82 01 57 00 00 00 00 1E FD

01 90 02 CD C1

00 10 13 AA 00 01 02 00 55 51 34



03

Sungrow Modbus Protocol

Sungrow Modbus Protocol

Information for Modbus Frame:

- 1) Address (Modbus ID), Setting of device
- 2) Function, defined by modbus protocol
- 3) Register, defined by device supplier (Sungrow)

To understand the modbus message we need the meaning of each register, including the meaning of the value in register. All this information is in device communication protocol.

Sungrow Modbus Protocol

String inverter protocol:

<https://sungrow.egnyte.com/dl/ogPz0pFEqz>

Residential Hybrid Inverter protocol:

<https://sungrow.egnyte.com/dl/D1ZncjZFOg>

Logger protocol:

<https://sungrow.egnyte.com/dl/EaP2HIPM8H>

Logger 3rd party device protocol

<https://sungrow.egnyte.com/dl/2vFXPH2DCb>

DC combine box protocol:


<https://sungrow.egnyte.com/dl/V3soS3wBIG>

Turnkey system protocol:

<https://sungrow.egnyte.com/dl/PcqFDCNKV4>

Some projects may have their special communication protocol !

Sungrow Modbus Protocol



Clean power for all

Public

Communication Protocol of PV Grid-Connected String Inverters

V1.1.32

Version number	Date	Note
V1.1.0	2016-4-11	initial version. Unofficial version(V1.0.13) is no longer used.
V1.1.1	2016-5-12	modify the register address and some related content.

Public: document is free to share to customer

Version number: to verify if current protocol the actual is.

Version table: the changing history of this protocol

Sungrow Modbus Protocol

Valid for device types:

In production:

SG30KTL-M, SG30KTL-M-V31, SG33KTL-M, SG36KTL-M, SG33K3J, SG49K5J, SG34KJ, LP_P34KSG,
SG50KTL-M-20, SG60KTL, G80KTL, SG80KTL-20, SG60KU-M
SG5KTL-MT, SG6KTL-MT, SG8KTL-M, SG10KTL-M, SG10KTL-MT, SG12KTL-M, SG15KTL-M,
SG17KTL-M, SG20KTL-M,
SG80KTL-M, SG111HV, SG125HV, SG125HV-20, SG33CX, SG40CX, SG50CX, SG110CX, SG250HX,
SG30CX,SG33CX-US,SG55CX-US,SG250HX-US
SG25CX-SA,SG100CX-JP,SG250HX-IN

Discontinued:

SG30KTL, SG10KTL, SG12KTL, SG15KTL, SG20KTL, SG30KU, SG36KTL, SG36KU, SG40KTL,
SG40KTL-M, SG50KTL-M, SG60KTL-M, SG60KU

Statement:

All hardware versions of SG60KTL share one device type code.

All Sungrow communication protocol will declare the devices, for which this protocol is valid.

Sungrow Modbus Protocol

1. Introduction

This communication adopts modbus RTU protocol, and applies to the communication between Sungrow PV grid-connected string inverters and the upper computer (PC) monitoring software. This protocol can read the real-time operating data and fault states of inverters.

2. Communication Interface

1) RS485

	Default setting
Address	Inverter: 1 – 247 settable PC: 1 – 247 settable
Broadcast	Yes
Baud rate	9600bit/s
Check bit	Null or settable
Data bit	8
Stop bit	1
Mode	RTU
Appliance interface	RS485-2W cable connection

2) Ethernet (optional)

Default:

- IP: 192.168.1.100;
- Sub-Net: 255.255.0.0
- Port: 502

Till now, only EC serial, SH serial, Turnkey System, Logger can be communicated via Ethernet/Modbus TCP.

Normally String inverter will only be communicated via RS485/Modbus RTU

Please notice the default setting for RS485, like Baudrate, Check bit, Data bit, and Stop bit.

Sungrow Modbus Protocol

3. Definition of Address

4. Data type

U16: 16-bit unsigned integer, big-endian

S16: 16-bit signed integer, big-endian

U32: 32-bit unsigned integer; little-endian for double-word data. Big-endian for byte data

S32: 32-bit signed integer; little-endian for double-word data. Big-endian for byte data

Example:

transmission order of U16 data 0x0102 is 01, 02

transmission order of U32 data 0x01020304 is 03, 04, 01, 02

The transmission order of multibyte data UTF-8: the high-byte data is in the front and the low-byte data is at back.

Example: transmission order of UTF-8 data ABCD is A, B, C, D.

Inverter will use 2 x 16 bit register to save a 32 bit Data. Please notice the special rule of Sungrow 32 bit data!

Sungrow Modbus Protocol

11	Total power yields	5004~5005	U32		kWh	
----	--------------------	-----------	-----	--	-----	--

For example, if we read the total yields from inverter:

Master: 01 04 13 8B 00 02 05 65

Slave : 01 04 04 12 34 00 00 BF 32

Correct process:

Original data from message: 12 34 00 00

Words exchange according to Sungrow 32 bit data rule:

12 34 00 00 => 00 00 12 34 => (convert to decimal) 4660 kWh

Wrong process:

12 34 00 00 => (convert to decimal) 305,397,760 kWh

Sungrow Modbus Protocol

2. Value description

The decimal parameters are transmitted as integer after expansion. For example: 10.333 KW is transmitted as 10333; 800.5 V is transmitted as 8005. Negative numbers are transmitted as complement, 0xFFFF signifying -1. Unavailable register cannot be viewed or set. The return of unsigned number is F, For example: “0xFFFF” is the return for U16, “0xFFFFFFFF” is the return for U32; the return of signed number is the max. positive number, e.g. “0x7FFF” for S16, “0x7FFFFFFF” for S32; 0x00 for UTF-8. UTF-8 occupies 1 byte. The length of odd number is complemented by 0x00.

Sungrow Modbus Protocol

3. Address type

Address of 3x type is read-only register, supporting the cmdcode inquiry of 0x04.

Address of 4x type is holding register, supporting the cmdcode inquiry of 0x03, and cmdcodeswrite-in of 0x10 and 0x06. Cmdcodes 0x10 and 0x06 support the broadcast address.

Support Modbus error code 02 (address error), 04 (setting failure).

Visit all registers by subtracting 1 from the register address. Example: if the address is 5000 –5001, visit it using address 4999 –5000. Entering “01 04 1387 00 02 + CRC” to check the data of address 5000 –5001.

4. Verify type

CRC16 generates polynomial 0xA001, little-endian.

Sungrow Modbus Protocol

3.1 Running information variable address definition (read-only register, Address type: 3X)

No.	Name	Address	Data type	Data range	Unit	Note
5	Reserved	4984 – 4989	U16			
6	SN	4990 – 4999	UTF-8			Data type :UTF-8
7	Device type code	5000	U16			See Appendix 6
8	Nominal active power	5001	U16		0.1kW	

a) Parameter setting address definition (holding register, Address type: 4X)

No.	Name	Address	Data type	Data range	Unit	Note
Setting data						
1	System clock: Year	5000	U16			

Sungrow Modbus Protocol

Appendix

Appendix 1 Device Work State 1

Device state (register 5038)	
State	Value read by register 5038
Run	0x0
Stop	0x8000
Key stop	0x1300
Emergency Stop	0x1500
Standby	0x1400
Initial standby	0x1200
Starting	0x1600
Alarm run	0x9100
Derating run	0x8100
Dispatch run	0x8200
Fault	0x5500
Communicate fault	0x2500

Appendix 2 Device Work State2

Work State (5081 – 5082)		Note
State	Corresponding BIT in address 5081-5082	
Run	0	Total run state bit BIT17
Stop	1	1
Key stop	3	3
Emergency Stop	5	5
Standby	4	4
Initial standby	2	2
Starting	6	6
Alarm run	10	Total run state bit BIT17
Derating run	11	Total run state bit BIT17
Dispatch run	12	Total run state bit BIT17
Fault	9	Total fault state bit BIT18
Communicate fault	13	Total fault state bit BIT18
Total run bit (device is grid-connected running)	17	
Total fault bit (device is in fault stop state)	18	

Sungrow Modbus Protocol

Appendix for bit meaning

Fault state 1				
No.	Item	BIT	Data range	Note
1	DC undervoltage	BIT 0	0: normal/1: fault	
2	DC overvoltage	BIT 1	0: normal/1: fault	
3	AC undervoltage	BIT 2	0: normal/1: fault	
4	AC overvoltage	BIT 3	0: normal/1: fault	
5	Underfrequency	BIT 4	0: normal/1: fault	
6	Overfrequency	BIT 5	0: normal/1: fault	
7	Contactora fault	BIT 6	0: normal/1: fault	
8	Islanding protection	BIT 7	0: normal/1: fault	
9	Sensor failure	BIT 8	0: normal/1: fault	
10	PDP protection	BIT 9	0: normal/1: fault	
11	Module overtemperature	BIT 10	0: normal/1: fault	
12	Reactor overtemperature	BIT 11	0: normal/1: fault	
13	Transformer overtemperature	BIT 12	0: normal/1: fault	
14	DC leakage current protection	BIT 13	0: normal/1: fault	
15	AC leakage current protection	BIT 14	0: normal/1: fault	
16	Overload protection	BIT 15	0: normal/1: fault	

Sungrow Modbus Protocol

8. Examples

Take ComTest for example.

a) Acquire one piece of running information

Supposed that the inverter address is 1, it needs to acquire data from address 5000 of 3x address type.

The PC sends (HEX):

01 04 13 87 00 01 85 67

The inverter replies (HEX):

01 04 02 01 32 39 75

Note: The type code of inverter SG60KU-M is 0x0132.

The Examples at the end of protocol is very useful for customer.

Sungrow Modbus Protocol

a) Parameter setting address definition (holding register, Address type: 4X)

No.	Name	Address	Data type	Data range	Unit	Note
Setting data						
1	System clock: Year	5000	U16			Receive time synchronization setting of the monitoring system
2	System clock: Month	5001	U16			
3	System clock: Day	5002	U16			
4	System clock: Hour	5003	U16			
5	System clock: Minute	5004	U16			
6	System clock: Second	5005	U16			
7	Start/Stop	5006	U16	0xCF (Start) 0xCE (Stop)		
8	Power limitation switch	5007	U16	0xAA: Enable; 0x55: Disable		
9	Power limitation setting	5008	U16	See Appendix 6	0.1%	Available when the power limitation switch (5007) is enabled

Meaning of each column:

Name
Address
Data type
Data range
Unit
Note

Sungrow Modbus Protocol

The rare UTF-8 register

6	SN	4990 – 4999	UTF-8			Data type :UTF-8
---	----	-------------	-------	--	--	------------------

Register using Appendix

7	Device type code	5000	U16			See Appendix 6
---	------------------	------	-----	--	--	----------------

Register value can only be several special value

9	Output type	5002	U16	0-two phase; 1-3P4L; 2-3P3L		
---	-------------	------	-----	-----------------------------------	--	--

Some register not work for all devices

14	Total apparent power	5009~5010	U32		VA	Valid for inverters: SG5KTL-MT
----	----------------------	-----------	-----	--	----	-----------------------------------

Sungrow Modbus Protocol

Default address (modbus ID) of devices:

All Sungrow Inverter : 1.

(notice: Logger3000 can discover connected inverter and set address to another unique value) for it.

DC combine box : 1

Devices, which are connected to logger, have forwarding ID. Forwarding ID is the value,

Sungrow Modbus Protocol

Exercise:

- 1) Create a command, to read device code of SG250HX, address 1
- 2) Create a command, to stop SH10RT, address 2
- 3) Create a command, to set power limit as 60% for SG3125HV, address 3
- 4) Create a command, to set reactive power factor as -0.9 for SG110CX, address 4
- 5) Create a command, to set power limit as 30% for Logger1000.

THANK YOU!