XY6020L 20A / 1200W Programmable Power Supply

Modbus Interface Documentation

Preface:
Those modules are currently sold at AliExpress (tm).
Unfortunately there was no feasible documentation about the advertised Modbus interface available.
I could not find out the original manufacturer, but a friendly seller provided at least a Chinese written documentation (on request).

This is now a compilation of the most relevant translated excerpts from the Chinese document (which has no no Copyright notice and does not mention the author).
Enriched with some more explanations and hints.

*Addressed to people who are already familiar with **Modbus RTU**.*
Other's may want to try a Google search for 'Modbus RTU Tutorial' first.

tinkering4fun at GitHub
April 2023

Communication Interface

The communication **protocol** is **MODBUS-RTU**.

Factory **Modbus ID** setting is **01H**

The communication **interface** is **TTL serial port**;

*The **data format** is **8,N,1** (8 data, no parity, one stop bit(s))*
*The factory **baudrate** setting is **115200***

Interface Connector:
VCC: 5V supply (used for e.g. isolated RS485 adapter)
TX: Serial data input (connect it to master's transmitter output)
RX: Serial data output (connect it to master's receiver input)
GND: Ground

Unfortunately there was nothing said about the expected 'TTL' signal levels.
As the module provides 5V on the VCC pin, it should be 5V tolerant on the input as ('TTL' would imply).
During my testing, it worked with either 3.3V or 5V.

Rem: The XY6020L module is typically bundled with a 'SiniLink_XY-WFPOW' Wifi interface board which is based on ESP8285. I guess this module works with 3.3V I/O levels.)

*Connection to a **RS485** interface line (real Modbus) requires e.g. **HW-519 RS485 adapter.** In this case TX goes to adapter's RXD and RX to adapter's TXD*

Function Codes

This machine only supports 0x03, 0x06, 0x10 function codes.

*This means, all mentioned registers are '**Holding Registers**'*

Function code	***Definition***	***Operation (binary)***
0x03	Read register data	Read data from one or more registers
0x06	Write a single register	Write a set of binary data to a single register
0x10	Write multiple registers	Write multiple sets of binary data to multiple registers

Register Set *(Holding Registers)*

Rem: Each register consists of 16 bits, transferred as 2 bytes.

Name	***Description***	***Bytes***	***Dec.***	***Unit***	***Read and write***	***Register address***
V-SET	Voltage setting	2	2	V	R/W	0000H
I-SET	Current setting	2	3	A	R/W	0001H
VOUT	Output voltage display value	2	2	V	R	0002H
IOUT	Output current display value	2	3	A	R	0003H
POWER	Output power display value	2	2	W	R	0004H
UIN	Input voltage display value	2	2	V	R	0005H
AH-LOW	Output AH low 16 bits	2	0	maH	R	0006H
AH-HIGH	Output AH high 16 bits	2	0	maH	R	0007H
WH-LOW	Output WH low 16 bits	2	0	mwH	R	0008H
WH-HIGH	Output WH high 16 bits	2	0	mwH	R	0009H
OUT_H	Open time - hours	2	0	H	R	000AH
OUT_M	Start time - minutes	2	0	M	R	000BH
OUT_S	Open time - seconds	2	0	S	R	000CH
T_IN	internal temperature value	2	1	F/C	R	000DH
T_EX	External temperature value	2	1	F/C	R	000EH
LOCK	key lock	2	0	-	R/W	000F
PROTECT	protection status	2	0	-	R/W	0010H
CVCC	Constant voltage and constant current state	2	0	-	R	0011H
ONOFF	Switch output	2	0	-	R/W	0012H
F-C	Temperature symbol	2	0	-	R/W	0013H
B-LED	Backlight brightness level	2	0	-	R/W	0014H
SLEEP	off screen time	2	0	M	R/W	0015H
MODEL	Product number	2	0	-	R	0016H

VERSION	Firmware version number	2	0	-	R	0017H
SLAVE-ADD	slave address	2	0	-	R/W	0018H
BAUDRATE_L	baud rate	2	0	-	R/W	0019H
T-IN-OFFSET	internal temperature correction	2	1	F/C	R/W	001AH
T-EX-OFFSET	External temperature correction	2	1	F/C	R/W	001BH
BUZZER	buzzer switch	2	0	-	R/W	001CH
EXTRACT-M	Quickly call out data sets	2	0	-	R/W	001DH
DEVICE	device status	2	0	-	R/W	001EH
MASTER	host type	2	0	0	R/W	0030H
V-SET	voltage setting	2	2	V	R/W	0050H
I-SET	current setting	2	3	A	R/W	0051H
S-LVP	Low voltage protection value	2	2	V	R/W	0052H
S-OVP	Overvoltage protection value	2	2	V	R/W	0053H
S-OCP	Overcurrent protection value	2	3	A	R/W	0054H
S-OPP	Over power protection value	2	1	W	R/W	0055H
S-OHP_H	Maximum output time--hours	2	0	H	R/W	0056H
S-OHP_M	Maximum output duration—minutes	2	0	M	R/W	0057H
S-OAH_L	Maximum output AH low 16 bits	2	0	maH	R/W	0058H
S-OAH_H	Maximum output AH high 16 bits	2	0	maH	R/W	0059H
S-OWH_L	Maximum output WH low 16 bits	2	0	10mwH	R/W	005AH
S-OWH_H	Maximum output WH high 16 bits	2	0	10mwH	R/W	005BH
S-OTP	Over temperature protection value	2	1	F/C	R/W	005CH
S-INI	Power-on output switch	2	0	-	R/W	005DH

Note 1:
This product is designed with a total of **10 sets of storage data sets M0-M9**, each set of serial number 20-2D a total of 14 data, of which the M0 data set is.
The data groups called by default when the product is powered on, the M1 and M2 data groups are the data groups quickly called out by the product panel, and the M3-M9 are common storage data groups
Group, the calculation method of the starting address of the data group is: 0020H+data group number*0010H, for example, the starting address of the M3 data group is:
0050H+3*0010H=0080H.

Note 2:
The reading and writing values of the **key lock function** are 0 and 1, 0 is unlocked, and 1 is locked.

Note 3:
The read value of **protection status** is 0-3:
0: normal operation, 1: OVP, 2: OCP, 3: OPP, 4: LVP, 5: OAH, 6: OHP, 7: OTP, 8: OEP, 9: OWH, 10: ICP.

Note 4:
The reading value of **constant voltage and constant current state** is 0 and 1, 0 is CV state, 1 is CC state.

Note 5:
The reading and writing values of the **switch output function** are 0 and 1, 0 is off, and 1 is on.

Note 6:
The reading and writing range of **backlight brightness level** is 0-5, level 0 is the darkest, level 5 is the brightest.

Note 7:
The written value of the **quick recall data group function** is 0-9, and the corresponding data group data will be automatically called out after writing.

Note 8:
Description of **WiFi related registers**
Rem: Only relevant with Wifi module (not tested)

Name	***Detailed description***	***Register address***
MASTER	Host type (0x3B3A: WIFI, others to be determined)	0030H
WIFI-CONFIG	WIFI pairing status (0: Invalid 1: Touch pairing 2: AP pairing)	0031H
WIFI-STATUS	WIFI status (0: Invalid network 1: Connected to the router 2: Successfully connected to the server 3: Touch pairing 4: AP pairing)	0032H
IPV4-H	The first two bytes of the IP address are 0xC0A8	0033H
IPV4-L	The last two bytes of the IP address are 0x0108	0034H

IPV4-H: 0xC0A8 - IPV4-L: 0x0108
IPV4 Address = 192.168.1.8

Communication example

The host reads the output voltage and output current display values

The message format sent by the host:

Data Item	***Number of bytes***	***Data sent***	***Remark***
Slave address	1	01	Send to slave with address 01
Function code	1	03	Read register
Register start address	2	0002H	Register start address
Number of register addresses	2	0002H	2 bytes in total
CRC code	2	65CBH	The CRC code calculated by the host

For example, if the current display value is 05.00V, 1.500A, then the

Message format returned by the slave machine:

Data Item	***Number of bytes***	***Data sent***	***Remark***
Slave address	1	01	Received from slave with address 01
Function code	1	03	Read register
Bytes in response	1	04	Number of data bytes in response
Content of register 0002H	2	01F4H	Output voltage display value
Content of register 0003H	2	05DCH	Output current display value
CRC code	2	B8F4H	The CRC code calculated by the slave

Output of Modbus test program

The following command of my Modbus utility, reads 2 holding registers, starting at register number 2.
The program prints out the bytes sent and received at the communication interface. Just to illustrate the message exchanged ...

```
./mbpoll -b 115200,8,N,1  -h 2 -n 2
```

Please consider that the current reading here was zero, and not 1.5 A

```
Opening /dev/ttyUSB0 at 115200 bauds (N, 8, 1)
[01][03][00][02][00][02][65][CB]          << request sent
Waiting for a confirmation...
<01><03><04><01><F4><00><00><BA><3D>      << response received

500                 << Register #2  decimal value of voltage
0                   << Register #3 decimal value of current
```