
# JT-DPM86XX

## Programmable lab power supply

Joy-IT powered by SIMAC Electronics GmbH - Pascalstr. 8 - 47506 Neukirchen-Vluyn - www.joy-it.net

## 1. DEBUG DPM86XX WITH SERIAL ASSISTANT

1. According to the "simple communication protocol" and "MODBUS communication protocol" provided by us, we can control the DPM86XX. The two communication protocols can be freely selected in the DPM86XX function "5-CS", 0 means "custom communication protocol", 1 means "MODBUS communication protocol".
2. You can use the serial port assistant to debug commands before programming with the communication protocol.
3. For example: 01w10 = 1234,\r\n, indicating that the voltage setting is; 12.34V.

## 2. SIMPLE COMMUNICATION PROTOCOL

### 2.1 Overview

The structure of the control command is a command line. The communication rate can be selected between seven baud rates (2400, 4800, 9600, 19200, 38400, 57600, 115200). The machine address code can be set in the range of 01-99. For the specific settings, please refer to the manual.

The command is sent from the PC and the device analyzes and executes it.

1. if the address code is the same, the result is returned to the PC.
2. if the address code is different, no information is returned.

This is very suitable for centralized control of multiple machines.

| start symbol | address code | function symbol | function member | equal symbol | operand | comma | end code |
|---|---|---|---|---|---|---|---|
| : | 01 - 99 | w, r | 00 - 99 | = | 0 - 65535 | , | \r\n |

Send command format is as follows:

1. The start symbol is ":".
2. The address code is the local address, and the setting range is 01 - 99.
3. The function code is "w" or "r", indicating write or read.
4. The function number is the number for different functions, and the different values represent different parameter settings.
5. The operand is the operand of the command.
6. Comma: each operand is distinguished by ",".
7. End code: The end of a command is "\r\n", this is actually a return character, and a newline character in ASCII, hexadecimal representation is 0x0d and 0x0a.

### 2.2 write instructions

| Com-mand | Command description | Format | Format description | Example | Example Description |
|---|---|---|---|---|---|
| 10 | Write voltage setting | :01w10=**** | "****" means a value, indicates the value of voltage setting | :01w10=1234,\r\n, | indicates that the voltage setting is: 12.34V |
| 11 | Write current setting | :01w11=**** | "****" means a value, indicating the value of current setting | :01w11=12345,\r\n | indicating that the current setting is: 12.345A |
| 12 | Output status setting | :01w12=* | "*" means a value, representing the output status | :01w12=0 | means to turn off the output |
| - | - | - | - | 01w12=1,\r\n, | means to turn on the output |

| Command | Command description | Format | Format description | Example | Example Description |
|---|---|---|---|---|---|
| 20 | set both voltage and current settings | :01w20=****,####, | "****" means the voltage setting, "####" means the current setting | :01w20=1234,12345, | means that the voltage and current settings qual 12.34V and 12.345A |

**2.3 read instructions**

| Command | Command description | Format | Example | Example Description |
|---|---|---|---|---|
| 00 | read maximum output voltage | :01r00=0, | Send: :01r00=0,<br>Return: :01r00=6000 | indicates the maximum output voltage is 60V |
| 01 | read maximum output current | :01r01=0, | Send: :01r01=0,<br>Return: :01r01=24000, | it means that the maximum output current of this model is 24A and the corresponding model is DPM-8624 |
| - | - | - | Return: :01r01=16000, | it means that the maximum output current of this model is 16A and the corresponding model is DPM-8616 |
| - | - | - | Return: :01r01=8000, | it means that the maximum output current of this model is 8A and the corresponding model is DPM-8608 |
| - | - | - | Return: :01r01=5000 | it means that the maximum output current of this model is 5A and the corresponding model is DPM-8605 |
| 10 | read the voltage setting | :01r10=0, | Send: :01w10=0,<br>Return: :01r10:1234, | Which indicates that the voltage setting is 12.34V |
| 11 | Read the current setting | :01r11=0, | Send: :01w11=0,<br>Return: :01r11:12345, | Which indicates that the current setting is 12.345A |
| 12 | Read output status | :01r12=0, | Send: :01w12=0,<br>Return: :01r12=0, | it indicates that the output status is currently off |
| - | - | - | Return: :01r12=1, | it indicates that the output status is currently on |
| 30 | read output voltage measurment value | :01r30=0, | Send: :01w30=0,<br>Return: :01r30=2345, | It indicates that the measured value of the output voltage is 23.45V |
| 31 | read output current measurment value | :01r31=0, | Send: :01w31=0,<br>Return: :01r31=12345, | It indicates that the measured value of the output current is 12.345A |
| 32 | read CC or CV status | :01r32=0, | Send: :01w32=0,<br>Return: :01r32=0, | it indicates the constant voltage output (CV) |
| - | - | - | Return: :01r32=1, | it indicates the constant current output (CC) |

| Com-mand | Command description | Format | Example | Example Description |
|---|---|---|---|---|
| 33 | Read tempera-ture | :01r33=0, | Send: :01w33=0,<br>Return: :01r33=30, | It indicates that the internal temperature of the machine is 30°C. |

## 3. MODBUS COMMUNICATION PROTOCOL

### 3.1 Communication protocol overview:

The communication protocol for this device is specifically MODBUS RTU. This can be used via RS485. Note: This product only supports MODBUS RTU function codes 0x03 (03H), 0x06 (06H), 0x10 (10H).

### 3.2 Communication protocol described:

(For a complete description of the general MODBUS RTU protocol, see http://modbus.org/docs/PI_MBUS_300.pdf). Modbus is a general protocol. The implementation of the location and function of the registers is specific by the designer of this device, however the format of the command structure for Modbus communication is defined and detailed in the above document. Below is a brief introduction to the standard Modbus RTU protocol and the registers and Modbus RTU function codes used with this particular device.

Data Format:

| Serial data format | 10 (binary system) |
|---|---|
| Start bit | 1 |
| Data bits | 8 |
| Parity bit | No |
| Stop bit | 1 |

Data frame structure:

| Data-frame interval | Address code | Function code | Data area | CRC check |
|---|---|---|---|---|
| 3.5 Bytes | 1 Byte | 1 Byte | N Bytes | 2 Bytes |

Before sending data, the bus silent period (no data transmission time) is more than 3.5 characters (e.g.: baud rate is 9600, time is 5ms).
RTU messages start with a silent interval of at least 3.5 characters, and after the last character sent, a similar interval of at least 3.5 characters marks the end of the message. After this interval a new message can start.
The entire message frame must be transmitted as a continuous stream. If a pause of more than 3.5 characters occurs before the frame is completed, the receiving device deletes the incomplete message and assumes that the next byte is the address field of a new message.
If a new message starts earlier than 3.5 characters after a previous message, it is considered a continuation of the previous message by the receiving device. This results in an error because the value in the last CRC field is not valid for the combined messages.

### 3.3 Address field

The address field of a message contains eight bits, from 1 to 255. This byte indicates the slaver whose address has been set by the user and which receives the message from the control panel. Each slaver must have a unique address field, and only the slaver that fits into the address field can answer a loopback message. When the slaver sends the message back, the loopback data starts with its own address field. The address field sent by the main is the address the slaver received, and the loopback address field is the address the slaver stored for the loopback. The corresponding address field shows where the message came from.

**3.4 Function field:**
The Function code field tells the addressed slave which function it should execute. This device supports only the functions 0x03 0x06 0x10.

| Function code | Definition | Operations (binary) |
|---|---|---|
| 0x03 | Read holding register | Read one or more registers in the slave. |
| 0x06 | Write single rgister | Writes binary data to a single register. |
| 0x10 | Write multiple registers | Writes values into a sequence of holding registers. |

**3.5 Data area**
The data bytes contain all additional information that the slave needs to execute the function. For example, function code 03 will request the slave to read holding registers and respond with their contents. The data field must contain the information that tells the slave which register to start at and how many registers to read. In the response, the data bytes contain the data collected by the slave, such as register values or status. When an error occurs, the function code is changed to indicate that the response is an error response, and the data bytes contain a code describing the error.

**3.6 The 2 MODBUS RTU function codes used by this device:**
**(ONLY these three codes)**
0x03 Read holding register(s):

| Address code | Function code | Register starting address | Number of register addresses n | CRC check |
|---|---|---|---|---|
| 1 Byte | 1 Byte | 2 Bytes | 2 Bytes | 2 Bytes |

0x03 Read holding register(s) function returns the following format:

| Address code | Function code | Returns the number ob bytes 2*n | Register Data | CRC check |
|---|---|---|---|---|
| 1 Byte | 1 Byte | 1 Byte | 2*n Bytes | 2 Bytes |

0x06 Write single register

| Address code | Function code | Register address | Register data | CRC check |
|---|---|---|---|---|
| 1 Byte | 1 Byte | 2 Bytes | 2 Bytes | 2 Bytes |

0x06 Write single register function returns the following format:

| Address code | Function code | Register address | Register data | CRC check |
|---|---|---|---|---|
| 1 Byte | 1 Byte | 2 Bytes | 2 Bytes | 2 Bytes |

0x10 Write multiple registers:

| Address code | Function code | Register starting address | Number of register address n | Number of bytes written 2*n | Register data | CRC check code |
|---|---|---|---|---|---|---|
| 1 Byte | 1 Byte | 2 Bytes | 2 Bytes | 1 Byte | 2*n Bytes | 2 Bytes |

0x10 Write multiple registers function returns the following format:

| Address code | Function code | Register starting address | Number of register address n | CRC check code |
|---|---|---|---|---|
| 1 Byte | 1 Byte | 2 Bytes | 2 Bytes | 2 Bytes |

### 3.7 Register map for this device

| Name | Explanation | Number of Bytes | radix point | unit | read \| write | postal address |
|---|---|---|---|---|---|---|
| Set-U | Voltage setting value | 2 | 2 | V | r \| w | 0000H |
| Set-I | Current setting value | 2 | 3 | A | r \| w | 0001H |
| SW | out switch | 2 | 0 | - | r \| w | 0002H |
| CCCV | output state | 2 | 0 | - | r | 1000H |
| U | Voltage display value | 2 | 2 | V | r | 1001H |
| I | Current display value | 2 | 3 | A | r | 1002H |
| T | Temperature display value | 2 | 0 | °C | r | 1003H |

Note:

1. The Set-UFT-Set-I is two set values used to control the voltage and current settings of the power supply.
2. Control whether the output is turned on, write 0 off, write 1 on, and can also read the state.
3. This is a read-only register, readout 0 means no output, 1 for CV status and 2 for CC state.
4. U, I is two real-time measurments used to read back the voltage and current values of the power supply.
5. T is the temperature measurment value, which is used to read the internal temperature of the power supply.

### 3.8 Error-checking

In RTU mode, the messages contain an error check field based on a cyclic redundancy check (CRC). The CRC field checks the contents of the entire message. It is applied regardless of the parity check method used for the individual characters of the message.

### 3.9 CRC Calculation method

The CRC (Cyclical Redundancy Check) field consists of two bytes containing a 16-bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC value to the message.

The receiving device recalculates a CRC while receiving the message and compares the calculated value with the actual value it received in the CRC field. If the two values are not equal, an error occurs.

The CRC is started by first preloading a 16-bit register to all 1s. Then a process begins in which successive eight-bit bytes of the message are applied to the current contents of the register. Only the eight data bits of each character are used to generate the CRC. Start and stop bits and the parity bit are not used for the CRC.

When generating the CRC, each eight-bit character is exclusively ORed with the register contents. The result is shifted in the direction of the least significant bit (LSB), with the most significant bit (MSB) being padded with a zero.

The LSB is extracted and examined. If the LSB was a 1, an exclusive OR operation (XOR) of the register with a specified, fixed value takes place. If the LSB was a 0, no exclusive OR operation (XOR) takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next eight-bit character is exclusive ORed with the current value of the register, and the process is repeated for eight more shifts as described above. The final content of the register, after all characters of the message have been transferred, is the CRC value.

## 5. THREE EXAMPLES OF COMMUNICATION USING THE THREE AVAILABLE MODBUS FUNCTION CODES SUPPORTED ON THIS DEVICE. (0X03) (0X03) AND (0X10)

**1. Read the displayed output voltage and current value. (Using function 0x03)**
Message format sent from Host:

| Host sends | Number of Bytes | Information sent | Notes |
|---|---|---|---|
| Slave address | 1 | 01H | From host to slave address 01H |
| Function code | 1 | 03H | Reading holding register(s) |
| Register starting address | 2 | 0000H | Register starting address |
| Number of registers to read | 2 | 0002H | A total of 2 registers (4 bytes = 2 words) |
| CRC checksum | 2 | C40BH | CRC checksum from host |

For example if the displayed values are 05.00V, 5.000A, the slave return response would be:

| HSlave responds | Number of Bytes | Information returned | Notes |
|---|---|---|---|
| Slave address | 1 | 01H | From slave address 01H |
| Function code | 1 | 03H | Read holding register |
| Number of bytes read | 1 | 04H | A total of 4 Bytes |
| The contents of the 1st register read | 2 | 01F4H | Output voltage display value |
| The contents of the 2nd register read | 2 | 1388H | Output current display value |
| CRC checksum | 2 | B76BH | CRC checksum from slave |

**2. Host to set the voltage to 24.00V (Using function 0x06)**
Message format sent from Host:

| Host sends | Number of Bytes | Information sent | Notes |
|---|---|---|---|
| Slave address | 1 | 01H | From host to slave address 01H |
| Function code | 1 | 06H | Write single register |
| Register address | 2 | 0000H | Register address |
| The value to write to the register | 2 | 0960H | Set the output voltage |
| CRC checksum | 2 | 8FB2H | CRC checksum from host |

Message format returned from slave:

| Slave responds | Number of Bytes | Information returned | Notes |
|---|---|---|---|
| Slave address | 1 | 01H | From slave address 01H |
| Function code | 1 | 06H | Write single register |
| Register address | 2 | 0000H | Register address |
| contents of the read register | 2 | 0960H | Value read |
| CRC checksum | 2 | 8FB2H | CRC checksum from slave |

**3. Host to set the voltage to 24.00V and the current to 1.500A. (Using function 0x10(decimal 16))**

Message format sent from host:

| Host sends | Number of Bytes | Information sent | Notes |
|---|---|---|---|
| Slave address | 1 | 01H | From host to slave address 01H |
| Function code | 1 | 10H | Write multiple registers |
| Register starting address | 2 | 0000H | Register starting address |
| Quantity of registers to write to | 2 | 0002H | 2 registers |
| Number of bytes to write | 1 | 04H | 4 Bytes = 2 words = 2 registers |
| Contents of 1st register | 2 | 0960H | Set the output voltage |
| Contents of 2nd register | 2 | 05DCH | Set the output current |
| CRC checksum | 2 | F2E4H | CRC checksum from host |

Message format returned from slave:

| From machine responses | Number of Bytes | Information returned | Notes |
|---|---|---|---|
| Slave address | 1 | 01H | From slave address 01H |
| Function code | 1 | 10H | Write multiple registers |
| Register starting address | 2 | 0000H | Register starting address |
| Number of registers written | 2 | 0002H | 2 registers written to |
| CRC checksum | 2 | 41C8H | CRC checksum from slave |

## 6. SONSTIGE INFORMATIONEN

Unsere Informations- und Rücknahmepflichten nach dem Elektrogesetz (ElektroG)

**Symbol auf Elektro- und Elektronikgeräten:**

Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte **nicht** in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben. Vor der Abgabe haben Sie Altbatterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

**Rückgabemöglichkeiten:**
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben. Kleingeräte, bei denen keine äußere Abmessungen größer als 25 cm sind können unabhängig vom Kauf eines Neugerätes in haushaltsüblichen Mengen abgeben werden.

**Möglichkeit Rückgabe an unserem Firmenstandort während der Öffnungszeiten:**
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

**Möglichkeit Rückgabe in Ihrer Nähe:**
Wir senden Ihnen eine Paketmarke zu, mit der Sie das Gerät kostenlos an uns zurücksenden können. Hierzu wenden Sie sich bitte per E-Mail an Service@joy-it.net oder per Telefon an uns.

**Informationen zur Verpackung:**
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

## 7. SUPPORT

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen, stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefon: +49 (0)2845 9360-50 (10 - 17 Uhr)

Für weitere Informationen besuchen Sie unsere Website:
**www.joy-it.net**

Veröffentlicht: 05.01.2023