iNexBot

# Network Function User Manual

# Catalogue

# Network Function User Manual

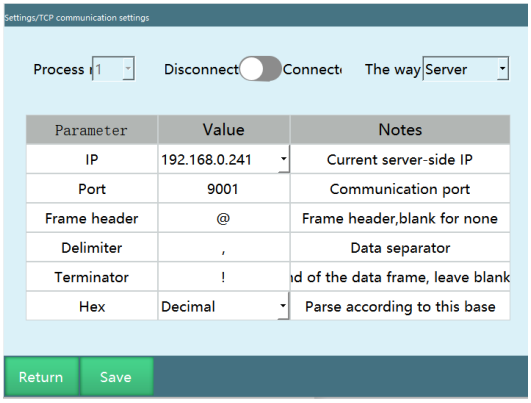
## TCP External Communication

### TCP Communication

When communicating with external devices, TCP communication can be selected.

Parameter settings

You can set TCP communication on the "Settings - TCP communication settings" interface.




Process number: 9 process numbers are supported.

Connection switch: When the mode is Client, white means disconnected, green means connected; when the mode is Server, white means closed, green means open. (The interface will indicate successful communication connection when connected)

Mode: Use the controller as a server or client. (When the controller is a client or a server, it can use different process numbers to communicate with multiple external devices and send and receive data; Note: The IP and port between the process numbers cannot be the same when the controller is a client with multiple process numbers connected to multiple server devices, and the port cannot be the same when the controller is a server with multiple process numbers connected to multiple client devices)

IP: When the controller is used as a server (the mode is Server), the IP here is the controller IP, no modification is required. When the controller is used as a client, the IP here needs to be set to the IP of the external communication device.

Port: When the mode is Server, it is the local listening port for client connection; when the mode is Client, it is the port for server connection.

Frame header: The frame header that is used when the controller receives messages from external devices during data communication; it can be modified.

Separator: The separator that is used when the controller receives messages from external devices during data communication; it can be modified.

Terminator: The terminator that is used when the controller receives messages from external devices during data communication; it can be modified.

Base: Select the corresponding base for the decimal or hexadecimal data to be received, then parse it in decimal and output.

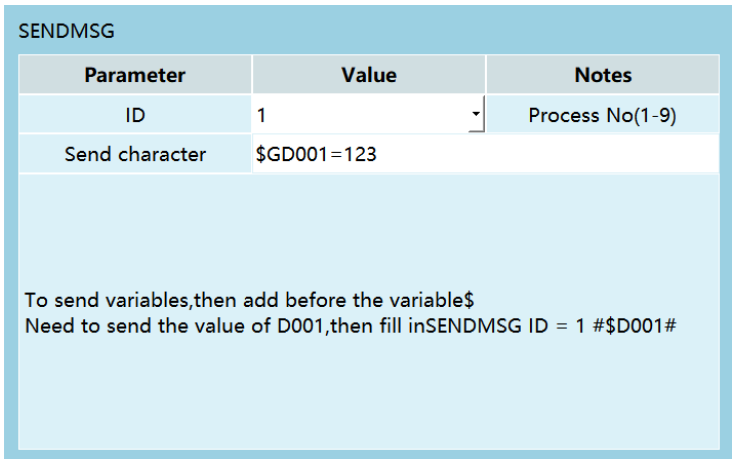
Note: When connecting through TCP communication, first set the IP of the controller and the IP of the external device to the same network segment, such as 192.168.1.xxx. If the controller is set as the client, the external device is the server, then set the IP and port in the network settings to be the same as those in the external device network debugging software, turn on the connection switch, the system will prompt that the connection is successful

## Network communication class instructions

### SENDMSG-Send data

This instruction is used to send data to the connected external device. Strings and variables can be sent by selecting the corresponding process number. Strings and variables can be mixed and sent. The frame header, separator, terminator and base set in the "Settings - TCP communication settings" interface are not used when sending data to external devices.

If you want to send a variable, add $ before the variable.




For example:

Prerequisite: GD001=123, I001=10

Need to send data "The value of GD001 is 123, and the value of I001 is 10" to the upper computer whose network setting process number is 3

Insert instruction SENDMSG:

ID=1

Send characters: The value of GD001 is $GD001, and the value of I001 is $I001

## PARSEMSG-Parse data

This instruction is used to parse a set of data from the external device.

This instruction will store the data from the external device in several global variables, and it need to set the first variable.

Not clear the cache after parsing: The data sent by the external device will be temporarily stored in the controller's cache; a. Not clear the cache: Data remains in the cache until the next set of data is sent after the first parsing; b. Clear the cache: The data in the cache will be cleared after the first parsing is completed.

PARSEMSG

| Parameter | Value | Notes |
|---|---|---|
| ID | 3 | Process No(1-9) |
| first variable in which the data is st | 001 | on where the queried data are stor |
| Clear cache after parsing | No | |
| Stored data number | | ecord the amount of extracted dat |

Example: PARSEMSG ID = 1 GI001 CLEARCACHE = 0 I001

When a TCP receives a multi-digit value, the value will be stored in multiple variables,

The variables used are the first Variables and the first variable are extended downwards.

That is, if a 3-digit value is sent, A, B, C, the first variable to be set is named GI006,

A is stored in GI006, and B is stored in GI007 , C is stored in GI008.

For example:

Frame header: @

Separator: ,

Terminator: !

The first variable type of the PARSEMSG instruction is GDOUBLE, and the first variable name is GD003.

External device sends data: @,12,6,47,102,77.88,!

Then the EXPLAIN instruction stores these five values in GD003, GD004, GD005, GD006, and GD007 respectively.

GD003=12, GD004=6, GD005=47, GD006=102, GDOO7=77.88

GD003=12

GD004=6

GD005=47

GD006=102

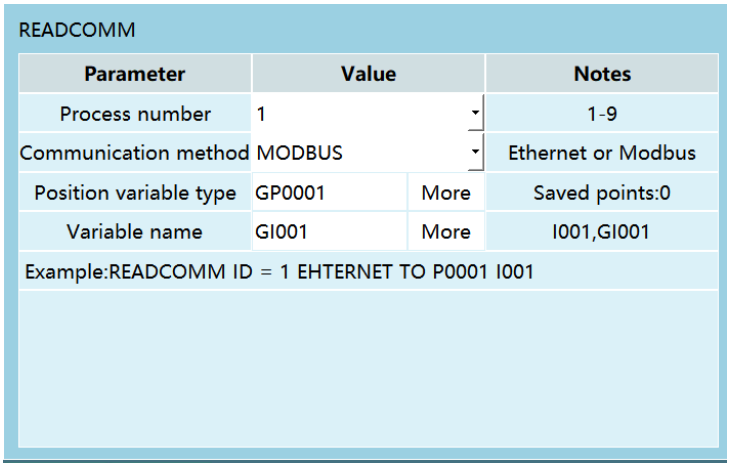
GD007=77.88

Clear cache after parsing: Yes or No

## READCOMM-Read

Read the points sent by Ethernet or Modbus and store them in the position variable, and store the number in the numerical variable.

Note: The method of use is the same as "External Point Function", this instruction currently only supports Modbus


READCOMM

| Parameter | Value | | Notes |
|---|---|---|---|
| Process number | 1 | | 1-9 |
| Communication method | MODBUS | | Ethernet or Modbus |
| Position variable type | GP0001 | More | Saved points:0 |
| Variable name | GI001 | More | I001,GI001 |
| Example:READCOMM ID = 1 EHTERNET TO P0001 I001 | | | |

Process number: The process number of the network communication to open the communication.

Communication method: Use Ethernet communication or Modbus communication.

Position variable type: Global position variable and local position variable can be selected.

Variable name: Store the number of received points.

## OPENMSG-Open data

Open the network communication corresponding to the process number. Run the OPENMSG instruction to open the communication. (Connect communication)

OPENMSG

| Parameter | Value | Notes |
|---|---|---|
| ID | 1 | Process No(1-9) |
| Example:OPENMSG ID = 1 | | |

Process number: The process number of the network communication to open the communication.

## CLOSEMSG-Close data

Close the network communication corresponding to the process number. Run the CLOSEMSG instruction to close the communication. (Disconnect communication)

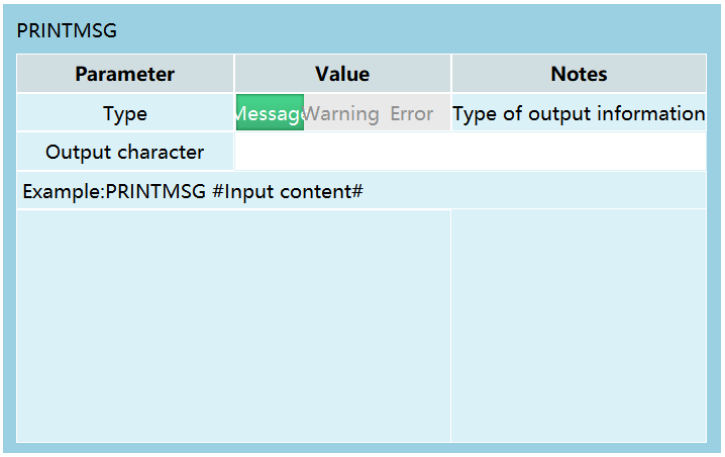
CLOSEMSG

| Parameter | Value | Notes |
|---|---|---|
| ID | 1 | Process No(1-9) |
| Example:CLOSEMSG ID = 1 | | |

Process number: The process number of the network communication to close the communication.

## PRINT-Output information

Screen output instruction; display the content on the teach pendant in three forms. Can output the data of custom characters or variables


PRINTMSG

| Parameter | Value | Notes |
|---|---|---|
| Type | Messag Warning Error | Type of output information |
| Output character | | |
| Example:PRINTMSG #Input content# | | |

Output information is now divided into three types: message, warning and error.

Output character: Output characters. Can input any character (support escape character), also can output variables, such as GD001 variable, GD001=10;

In the output information's message, warning and error instructions, enter $GD001 separately

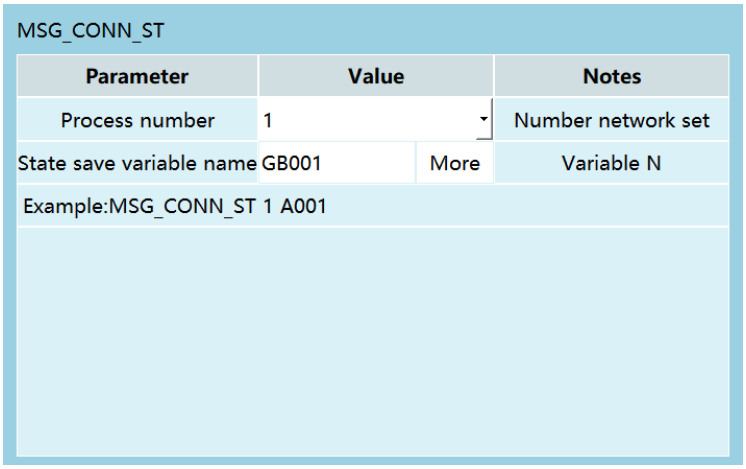
Now when running or stepping this instruction, the lower right corner of the teach pendant will display like this:

The message is a small white bar with the following content: 10;

The warning is a small yellow bar with the following content: 10;

The error is a small red bar with the following content: 10;

## MSG_CONN_ST-Get information connection status


MSG_CONN_ST

| Parameter | Value | | Notes |
|---|---|---|---|
| Process number | 1 | | Number network set |
| State save variable name | GB001 | More | Variable N |
| Example:MSG_CONN_ST 1 A001 | | | |

Process number: The process number for judging the connection status of network communication.

Stored variable type: Store the communication status into the local BOOL variable or the global GBOOL variable.

Stored variable name: The variable name of the variable that stores the communication status.

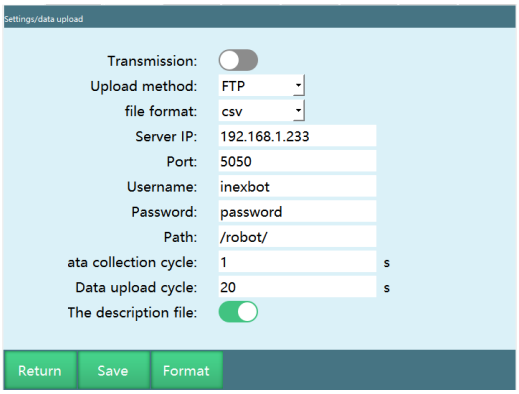
Read the current process number network communication status into the corresponding global Boolean or local Boolean variable. If the communication is normal, the stored value is 1, and if the communication fails, the stored value is 0.

# Data Upload

## Basic settings

The data upload function can automatically collect and upload the current robot operating status and parameters at regular intervals, integrate the data into csv and txt files and upload them to the designated server.

Click the [Modify] button in the "Settings - Data upload" to set the parameters required to connect to the ftp server.




Transmission: Once turned on, it starts to connect to the ftp server and upload data. After all parameters are filled in, turn on this switch. After this switch is turned on, the controller will automatically start collecting and uploading data when it is started up.

Upload method: Currently, only ftp protocol is supported. So please have an ftp server before using this function.

File format: Currently supports csv and txt formats. The file content is same, but the file format is different. The csv format is more convenient for data statistics.

Server IP: The IP address of the ftp server. Please ensure that the controller and the ftp server are in the same network, and ensure that their gateways are the same (the controller gateway can be viewed and modified in "Settings - System settings - IP settings").

Port: The port used by the ftp protocol of the ftp server. The default port used by the general ftp protocol is 21.

Username: The username used to log in to the ftp server. You need to create a user on the ftp server first.

Password: The password used to log in to the ftp server.

Path: The path used when the file is uploaded to the ftp server. This path is relative to the ftp root directory.

Data collection cycle: According to the set time, the controller collects the current data once and stores it in the file to be sent at regular intervals.

Data upload cycle: According to the set time, the controller sends the file with collected data to the specified directory of the ftp server at regular intervals.

Send description file: The description file is sent before the first sending of the data file after starting up the controller or turning on the "Transmission" enable

switch. The content is customizable and is generally used to describe the current robot's serial number and other information. If this switch is turned off, the description file will not be sent.

## Data format

Once you have configured the connection parameters for ftp, you need to configure the data format of the data file to be sent. When setting the data format, use a special string to represent the parameters to be sent. For example, if you want to send the current date in the following format "2019-03-07", you need to fill in the data format as follows: "$Y$%-$m$%-$d$%" (excluding quotation marks).

If the generated file is in csv format, each item should be separated by English commas (,).

The parameters represented by special strings are as follows:

### Example of generating a csv file

The desired results are as follows:

Description document file name:
Robot-R1_Year-Month-Day_Hour:Minute:Second_INFO

Description document content: Robot-R1, year-month-day, hour: minute: second, local IP, local MAC, technical department, machining parts, 1-axis motor speed, 2-axis motor speed, 3-axis motor speed, 4-axis motor speed, 5-axis motor speed, 6-axis motor speed, 1-axis motor torque, 2-axis motor torque, 3-axis motor torque, 4-axis motor torque, 5-axis motor torque, 6-axis motor torque, 1-axis motor load, 2-axis motor load, 3-axis motor load, 4-axis motor load, 5-axis motor load, 6-axis motor load, current controller status, current error code

Data document file name:
Robot-R1_Year-Month-Day_Hour:Minute:Second_DATA

Data content: Robot-R1, year-month-day, hour: minute: second, local IP, local MAC, 1-axis motor speed, 2-axis motor speed, 3-axis motor speed, 4-axis motor speed, 5-axis motor Speed, 6-axis motor speed, 1-axis motor torque, 2-axis

motor torque, 3-axis motor torque, 4-axis motor torque, 5-axis motor torque, 6-axis motor torque, 1-axis motor load, 2-axis motor load, 3-axis motor Load, 4-axis motor load, 5-axis motor load, 6-axis motor load, current controller status, current error code

The format of the data written is as follows:

Description document file name: Robot-R1_ $Y%-$m%-$d%_ $H%:$M%:$S%_INFO

Description content: Robot-R1,$Y%-$m%-$d%,$H%:$M%:$S%, $IP%,$MAC%,Technology Department,Machining Parts,$RPM_J1%,$RPM_J2%,$RPM_J3%,$RPM_J4%,$RPM_J5%,$RPM_J6%,$Torsion_J1%,$Torsion_J2%,$Torsion_J3%,$Torsion_J4%,$Torsion_J5%,$Torsion_J6%,$Load_J1%,$Load_J2%,$Load_J3%,$Load_J4%,$Load_J5%,$Load_J6%,$StatusCode%,$ErrorCode%

Data document file name: Robot-R1_ $Y%-$m%-$d%_ $H%:$M%:$S%_DATA

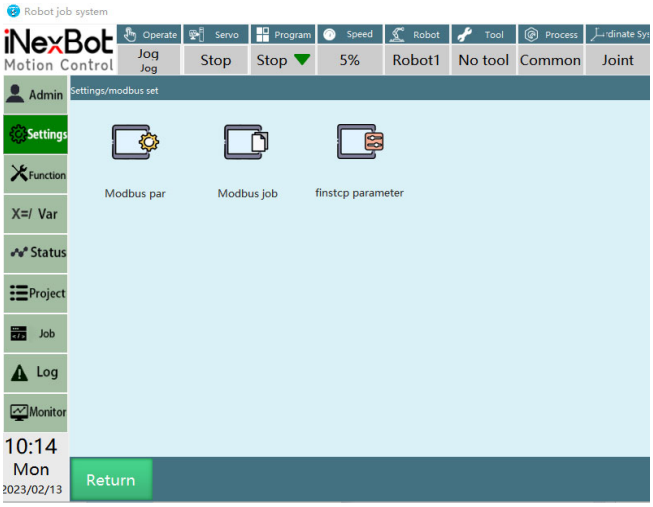
Data content: Robot-R1,$Y%-$m%-$d%,$H%:$M%:$S%, $IP%,$MAC%,$RPM_J1%,$RPM_J2%,$RPM_J3%,$RPM_J4%,$RPM_J5%,$RPM_J6%,$Torsion_J1%,$Torsion_J2%,$Torsion_J3%,$Torsion_J4%,$Torsion_J5%,$Torsion_J6%,$Load_J1%,$Load_J2%,$Load_J3%,$Load_J4%,$Load_J5%,$Load_J6%,$StatusCode%,$ErrorCode%

*For the parameters related to the axis, you need to manually input that axis, such as 1 axis speed: $RPM_J%, you needs to write 1 after J

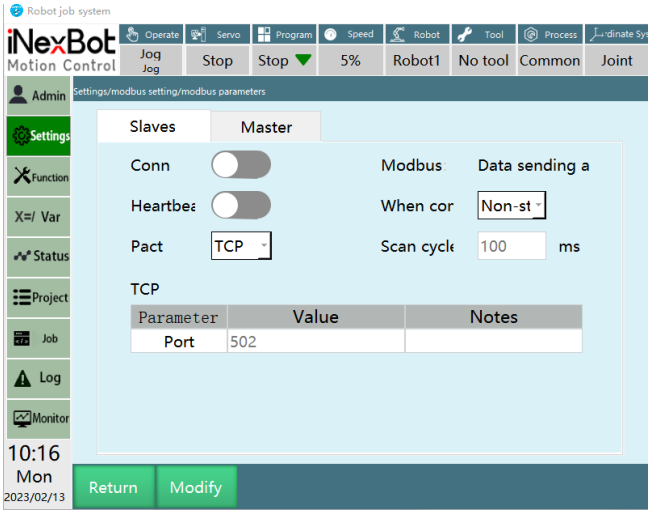
## External Transmission Point

### Parameter settings

External communication can use modbus. To set the parameters, you need to enter the "Settings - modbus settings - modbus parameters" interface. (You can also check modbus-related manual)




Parameters for detection and status display of switches




Connection: modbus-related switch, check the modbus signal after it is turned on.

Heartbeat detection: Turn on to detect the sending and receiving frequency between modbus and the controller. After disconnecting the modbus connection, the heartbeat detection shows that the data sending and receiving is closed.

Modbus: Display the connection status between modbus and the controller.

When the communication is disconnected:

No shutdown: when the Modbus slave is disconnected and the communication is disconnected, the controller will not stop running or power off;

Shutdown: When the Modbus slave is disconnected and the communication is disconnected, the controller will stop running or power off.

Scan cycle: The time required to perform a scan operation.

In this interface, you can set whether modbus is connected, the protocol used for modbus connection, the controller as a modbus master/slave, and each parameter when connected.

Controller is the master

TCP (port): the connection port of the slave

RTU (port): the port to which the slave modbus is connected

RTU (ID): ID of the slave

RTU (baud rate): baud rate of modbus, need to set

Controller is the slave

TCP (port): the port used by the controller to connect, need to set

RTU (ID): ID used by the controller to connect, need to set

RTU (port): the port used by the controller to connect, need to set

RTU (baud rate): the baud rate used by the controller to connect, need to set

## Communication method

Due to the limitation of address code, if there are too many points, then they need to be sent in batches, and a maximum of 30 points can be sent each time.

As long as the controller is connected to the PLC, the points can be sent, and the controller will automatically store them.

| Purpose | Address code | Process |
|---|---|---|
| All points sending flag | 1001 | Set it to 1 when PLC needs to send points, set it to 2 after sending all points, and set it to 0 after the controller receives them. |

| | | |
|---|---|---|
| The sending flag for sending once | 1002 | Set it to 1 when PLC needs to send points, set it to 0 after the controller receives them, and PLC sets it to 1 again for the next sending process. |
| The number of points sent once | 1003 | The number of points sent by PLC at one time, up to 30 |
| Data stored at points | According to the number | See below for detailed explanation |
| Frame number for each frame of data | 1004 | The number must be changed every time the points are sent, and it cannot be the same as the last time |
| Clear controller point queue flag | 1005 | To discard the point queue that has been sent to the controller, the PLC will set it to 1, and the controller will set it to 0 after the queue has been cleared. |

## Data Stored At Points

One point data contains the values of one coordinate system and six axes (if it is a 4-axis robot, it contains the values of one coordinate system and four axes).

| I-th point | Address code | Notes |
|---|---|---|
| Coordinate system | 1010+20*(i-1) | 1≤i≤32 |
| Whether to use | 1011+20*(i-1) | 1≤i≤32; use: send 0; not use: send 1 |
| The value of the j-th axis | 1010+2+20*(i-1)+2*(j-1) | 1≤i≤32, 1≤j≤9, the value of the axis uses float type, so it occupies two addresses |

Examples

Need to send 88 points. Since only 32 points can be sent each time, they need to be divided into 3 transmissions, and the number of transmissions is 32, 32, and 24 respectively.

The process is as follows:

PLC sets 1003 to 32, sets the value of each address code used by the point to store data, sets 1001 to 1, and sets 1002 to 1;

The controller detects that 1002 is 1,1001 is 1, then takes out the data of the point storage address code according to the value of 1003, and then sets 1002 to 0;

PLC detects that 1002 is 0, sets 1003 to 32, sets the data of the point storage address code, and then sets 1002 to 1;

The controller detects that the value of 1002 is 1, and the value of 1001 is 1, takes out the data of the point storage address code according to the value of 1003, and then sets 1002 to 0;

PLC detects that the value of 1002 is 0, sets 1003 to 24, sets the data of the point storage address code, sets 1001 to 2, and sets 1002 to 1;

The controller determines that 1002 is 1 and 1001 is 2, takes out the data of the point storage address code according to the value of 1003, and then sets the value of 1002 to 0, and sets the value of 1001 to 0.

# Instructions

## MOVCOMM-External point

This instruction is used to move the points stored in the controller in accordance with the set interpolation method.

MOVCOMM

| Parameter name | Parameter | | Notes |
|---|---|---|---|
| Runin | Joint | | |
| VJ | 10 | More | Speed range 1-100 |
| PL | 0 | More | Smooth transition(0-5) |
| ACC | 20 | More | Motion ACC |
| DEC | 20 | More | Motion DEC |
| TIME | 0 | More | Early execution,N(ms) |
| Example: MOVCOMM MOVL VJ = 10% PL = 0 ACC = 10 DEC = 10 | | | |

Interpolation method:

a. Joint

b. Linear

c. Curve

The interpolation method used during movement, all points are moved in this interpolation method.

VJ: The maximum speed during movement. Joint interpolation: 1-100; Other interpolation methods: 2-1000

PL: Position level, 0-5, 0 can be filled when the interpolation method is curve.

ACC: The maximum acceleration during movement.

DEC: The maximum deceleration during movement.

TIME: Early execution time; The next early executable instruction can be executed early.

Parameter source: Can be customized or bound to variables.