Communication Protocol User Manual

Catalog

1.0 Communication Protocol **2**

1.1 Package **2**

1.2 Response Package **3**

1.3 Instruction Type **4**

1.3.1 PING 4

1.3.2 READ DATA 5

1.3.3 WRITE DATA 5

1.3.4 REG WRITE 7

1.3.5 ACTION 8

1.3.6 SYNC WRITE 9

1.3.7 SYNC READ 10

1.3.8 RESET 12

1.0 Communication Protocol

The serial bus smart servo communication protocol is applicable to both potentiometer and magnetic encoder series servo. Potentiometer series servo uses TTL level and single bus communication (multiplexing of transmit and receive data signals when using one signal line). The physical connection consists of three wires, including positive, negative and power supply. Magnetic encoder series servo adopts an ARM 32-bit microcontroller as the main control core. The position sensing is based on a 360-degree, 12-bit precision magnetic angle scheme. The communication level adopts RS485 mode with strong anti-interference capability. The communication is still asynchronous duplex, where the transmission and reception signals are processed asynchronously.

The communication between the controller and the servo adopts question-and-answer mode. The controller sends out the instruction package and the servo returns the response package.

Multiple servos are allowed in a serial bus control network, so each servo is assigned a unique ID number in the network. The control instruction sent by the controller contains ID information, and only the servo matching the ID number can receive this instruction completely and return the reply information.

The communication mode is serial asynchronous, and a frame of data is divided into 1 start bit, 8 data bits and 1 stop bit, with no parity bit, and 10 bits in total.

The difference between the communication protocol of potentiometer series and magnetic coding series servos: when some parameters of the memory table are in the range of two bytes, the two bytes respectively represent the high byte and the low byte, in which the address of potentiometer series parameters is the high byte before the low byte, while the magnetic coding series servo is the low byte before the high byte.

In addition, each servo has slightly different functions, so please refer to the memory table of the specific model for actual control.

1.1 Instruction Package

Instruction package format:

Header	ID	Data length	Instruction	Parameters	Checksum
0XFF 0XFF	ID	Length	Instruction	Parameter1 ...Parameter N	Check Sum

Header: when two consecutive bytes of 0xFF are received, it indicates the data packet is received.

ID: Each servo has an ID number. The ID number from 0 to 253 can be represented in hexadecimal as 0x00 to 0xFD.

Broadcast ID: The ID number 254 is the broadcast ID. If the ID number sent by the controller is 254(0XFE, all the servos receive the command, and other commands do not return the response information except the PING command (broadcast PING command cannot be used when multiple servos are connected to the serial bus).

Data Length: equal to the parameter N to be sent plus 2, that is, "N+2".

Instruction: The data packet operation function codes can be found in section 1.3 of the instruction types.

Parameters: In addition to the instructions, there are additional control information that needs to be supplemented. The parameters can support a maximum of two-byte values to represent a memory value. The byte order should be referred to the memory control table in the servo user manual (the byte order may vary for different models of servos.

Check Sum: Checksum, which is calculated as follows:

Checksum = ~ (ID+length+instruction+parameter 1+... parameter n)

If the calculation within parentheses exceeds 255, then only the lowest byte is taken. The "~" symbol represents bitwise negation (taking the bitwise complement) of a byte.

1.2 Response Package

The response package is the response of the servo to the controller, and the format is as follows:

Header	ID	Data length	Instruction	Parameters	Checksum
0XFF 0XFF	ID	Length	ERROR	Parameter1 ...Parameter N	Check Sum

The response package contains the current status of the servo, indicated by the ERROR byte. If the current working status of the servo is abnormal, it will be reflected by this byte (see the manual memory control table for details of the meaning of each state). If ERROR is 0, it means that the servo has no error information.

If the instruction is READ DATA, then Parameter1 ...Parameter N represent the information that have been read.

1.3 Instruction Type

The available instructions for the communication protocol of the serial bus smart servos are as follows:

Instruction	Function	Value	Parameter Length
PING	Query Working Status	0x01	0
READ DATA	Query the character in the control table	0x02	2
WRITE DATA	Write the character into the control table	0x03	Greater than or equal to 1
REGWRITE DATA	Similar to WRITE DATA, but the control character does not act immediately after writing until ACTION.	0x04	Not less than 2
ACTION	Triggering the action of REG WRITE operation	0x05	0
SYCNREAD DATA	Query multiple servos at the same time.	0x82	Greater than or equal to 3
SYCNWRITE DATA	Controlling multiple servos at the same time.	0x83	Not less than 2
RESET	Reset the control table to the factory value.	0x06	0

1.3.1 Query Status Instruction PING

Function: read the working state of the servo

Length 0X02

Instruction: 0X01

Parameter: none

The PING command uses the broadcast address, and the servo also returns the response information.

Example 1 Read the working state of the servo with ID number 1:

Instruction frame: FF FF FF 01 02 01 FB' (sent in hexadecimal)

Header	ID	Effective data length	Instruction	Checksum
0XFF 0XFF	0X01	0X02	0X01	0XFB

Returned data frame: FF FF 01 02 00 FC (hexadecimal)

Header	ID	Effective data length	Working Status	Checksum
0XFF 0XFF	0X01	0X02	0X00	0XFC

1.3.2 READ DATA

Function: Read data from the servo memory control table.

Length:0X04

Instruction: 0X02

Parameter 1: the first address of the data readout segment

Parameter 2: Length of read data

Example 2: Read the current position of the servo with ID 1 (low byte first, high byte last).

Read two bytes from address 0X38 in the control table.

Command frame: FF FF 01 04 02 38 02 BE (sent in hexadecimal).

Header	ID	Effective Data length	Instruction	Parameters	Checksum
0XFF 0XFF	0X01	0X04	0X02	0X38 0x02	0XBE

Returned data frame: FF FF 01 04 00 18 05 DD (hexadecimal)

Header	ID	Effective Data length	Working Status	Parameters	Checksum
0XFF 0XFF	0X01	0X04	0X00	0X18 0X05	0XDD

The two bytes of data have been read are: low byte 0X18 and high byte 0X05, and these two bytes can be combined into a 16-bit data value 0X0518, the current position is represented in decimal as 1304.

1.3.3 WRITE DATA

Function: writing data to the servo memory control table

Length: N+3 (N is the parameter length).

Instruction: 0X03

Parameter 1: The first address of the data writing segment.

Parameter 2: The first written data

Parameter 3: The second data

Parameter N+1: the N+1 nth data

Example 3 sets an arbitrarily numbered ID to 1.

The address where the ID number is stored in the control table is 5, so just write 1 at address 5.

The ID of the sending instruction package is broadcast ID (0xFE).

Instruction frame: FF FF FE 04 03 05 01 F4 (sent in hexadecimal)

Header	ID	Effective Data length	Instruction	Parameters	Checksum
0XFF 0XFF	0XFE	0X04	0X03	0X05 0X01	0xF4

No data return as the broadcast ID is used to send instructions.

In addition, the memory table **EPROM** has a protection lock switch, which needs to be disabled (set to 0) before modifying the ID. Otherwise, the example ID will not be saved when power is off.

For detailed operation, please refer to the memory table or operation manual of the specific servo model number.

Example 4 controls the servo of ID1 to rotate to the position of 2048 at the speed of 1000 seconds.

The first address of the target location in the control table is 0X2A, so the writing of six consecutive bytes starts at address 0X2A.

They are position data 0x0800 (2048);

Time data 0x0000 (0);

Speed data 0X03E8(1000).

The instruction package is sent using a non-broadcast ID (0xFE), so after the instruction is received, the servo will return a status package.

Instruction frame:FF FF 01 09 03 2A 00 08 00 00 E8 03 D5 (sent in hexadecimal)

Header	ID	Effective data length	Instruction	Parameter	Checksum
0XFF 0XFF	0X01	0X09	0X03	0X2A	0XD5
				0X00 0X08	
				0X00 0X00	
				0XE8 0X03	

Returned data frame: FF FF 01 02 00 FC (hexadecimal)

Header	ID	Effective data length	Working status	Checksum
0XFF 0XFF	0X01	0X02	0X00	0XFC

The return work status of 0 indicates that the servo has correctly received the instruction and has started executing it without any errors.

1.3.4 REG WRITE

The REG WRITE instruction is similar to WRITE DATA, but the execution time is different.

When the REG WRITE instruction frame is received, the received data is stored in the buffer for standby, and the Registered Instruction register is set to 1.

When the ACTION command is received, the stored command is finally executed.

Length: N+3 (N is the number of data to be written)

Instruction: 0X04

Parameter 1: The first address of the area where the data is to be written.

Parameter 2: The first data to be written.

Parameter 3: The second data to be written

Parameter N+1: The Nth data to be written.

Example 5 controls the servo from ID1 to ID10 to rotate to the position of 2048 at the speed of 1000 per seconds. In the following instruction packets, only the ID 1 is on the bus and receives the instruction and returns, and other ID are not returned on the bus.

ID 1 Asynchronous write instruction package: FF FF 01 09 04 2A 00 08 00 00 E8 03 D4

ID 1 Return package: FF FF 01 02 00 FC

ID 2 Asynchronous write instruction package: FF FF 02 09 04 2A 00 08 00 00 E8 03 D3

ID 3 Asynchronous write instruction package: FF FF 03 09 04 2A 00 08 00 00 E8 03 D2

ID 4 Asynchronous write instruction package: FF FF 04 09 04 2A 00 08 00 00 E8 03 D1

ID 5 Asynchronous write instruction package: FF FF 05 09 04 2A 00 08 00 00 E8 03 D0

ID 6 Asynchronous write instruction package: FF FF 06 09 04 2A 00 08 00 00 E8 03 CF

ID 7 Asynchronous write instruction package: FF FF 07 09 04 2A 00 08 00 00 E8 03 CE

ID 8 Asynchronous write instruction package: FF FF 08 09 04 2A 00 08 00 00 E8 03 CD

ID 9 Asynchronous write instruction package: FF FF 09 09 04 2A 00 08 00 00 E8 03 CC

ID 10 Asynchronous write instruction package: FF FF 0A 09 04 2A 00 08 00 00 E8 03 CB

1.3.5 Execute Asynchronous Write Instruction ACTION

Function: trigger REG WRITE instruction

Length: 0X02

Instruction: 0X05

Parameter: None

ACTION command is very useful when controlling multiple servos at the same time.

When controlling multiple servos, the ACTION instruction can be used to make the first and last servos simultaneously execute their respective actions without any delay in between.

When sending an ACTION command to multiple servos, the broadcast ID(0xFE) is needed, so no data frame will be returned when sending this command.

Example 6 After sending the asynchronous write instruction to control servos ID1 to ID10 to rotate to the position of 2048 at a speed of 1000 units per second, when it is necessary to execute the asynchronous write command, the following command package (FF FF FE 02 05 FA) needs to be sent.

After receiving this command, all the servos on the serial bus will run the asynchronous write command received earlier.

1.3.6 SYNC WRITE

Function: used to control multiple servos at the same time.

ID: 0XFE

Length: (L+1) * N+4 (L:Length of data sent to each servo, N: servo numbers)

Instruction: 0X83

Parameter 1: writes the first address of data.

Parameter 2: written data length (L)
Parameter 3: the first servo ID.

Parameter 4: the first written data of the first servo.

Parameter 5: the second written data of the first servo.

...

Parameter L+3: the Lth written data of the first servo.
Parameter L+4: the second servo ID.

Parameter L+5: the first data written in the second servo.

Parameter L+6: the second data written in the second servo.

...

Parameter 2L+4: is written into the first data of the second servo.

Different from the **REG WRITE+ACTION** command, the real-time performance of this command is higher. A **SYNC WRITE** command can modify the contents of control tables of multiple servos at one time, while the **REG WRITE+ACTION** command is done step by step.

Nevertheless, when using the **SYNC WRITE** command, the length of the written data and the first address of the saved data must be the same.

Example 7, Set the initial addresses of four servos, ID1 to ID4, to 0X2A. Set the write position to 0X0800, time to 0X0000, and speed to 0X03E8 (with the low-byte first and the high-byte last).

Command frame: FF FF FE 20 83 2A 06 01 00 08 00 00 E8 03 02 00 08 00 00 E8 03 03 00 08 00 00 E8 03 04 00 08 00 00 E8 03 58 (sent in hexadecimal)

Header	ID	Effective data length	Instruction	Parameter	Checksum
0XFF 0XFF	0XFE	0X20	0X83	0X2A 0X06	0X58
				0X01 0X00 0X08 0X00 0X00 0XE8 0X03	
				0X02 0X00 0X08 0X00 0X00 0XE8 0X03	
				0X03 0X00 0X08 0X00 0X00 0XE8 0X03	
				0X04 0X00 0X08 0X00 0X00 0XE8 0X03	

No data return as the broadcast ID is used to send instructions.

1.3.7 SYNC READ

Function: Query multiple servos at the same time.

ID: 0XFE

Length: N+4 (N:Number of servos)

Instruction: 0X82

Parameter 1: The first address of reading data

Parameter 2: Length of read data

Parameter 3: The ID of the first servo

Parameter 4: The ID of the second servo

...

Parameter N+2: The ID of the Nth servo.

...

A SYNC READ command can query the contents of the control tables of multiple servos at one time. The synchronous read command specifies the ID of the servos to be queried, and the order in which the servos return the response packages is the ID order in the command package. When using the SYNC READ command, the data length and the first address of all queries must be the same (this command is open to some serial bus servos).

Example 8 Inquires about the first address 0X38 of two servos in ID1-ID2, including the data of current position, current speed, current load, current voltage and current temperature, totally 8 words (the low byte comes first and the high node comes last).

Synchronous reading instruction package: FF FF FE 06 82 38 08 01 02 36

Header	ID	Effective data length	Instruction	Parameter	Checksum
0XFF 0XFF	0XFE	0X06	0X82	0X38	0X36
				0X08	
				0X01	
				0X02	

Return package:

FF FF 01 0A 00 00 08 00 00 00 00 79 1E 55

FF FF 02 0A 00 FF 07 00 00 00 00 77 23 53

Decoding the return packet: ID1 return package: FF FF 01 0A 00 08 00 00 00 00 79 1E 55
00

Header	ID	Effective Data length	Working status	Parameters	Checksum
0XFF 0XFF	0X01	0X0A	0X00	0X00 0X08	0X55
				0X00 0X00	
				0X00 0X00	
				0X79	
				0X1E	

ID2 response package: FF FF 02 0A 00 FF 07 00 00 00 00 77 23 53

Header	ID	Effective Data length	Working status	Parameters	Checksum
0XFF 0XFF	0X02	0X0A	0X00	0XFF 0X07	0X53
				0X00 0X00	
				0X00 0X00	
				0X77	
				0X23	

1.3.8 RESET Instruction

Function: reset specific data in the memory control table (adopted for specific rudder model number)

Length 0X02

Instruction： 0X06

Parameter： None

For example, Reset the servo, the ID number is 0.

Command frame: FF FF 01 02 06 F6 (sent in hexadecimal)

Header	ID	Effective data length	Instruction	Checksum
0XFF 0XFF	0X00	0X02	0X06	0XF7

Returned data frame: FF FF 01 02 00 FC (hexadecimal)

Header	ID	Effective data length	Working Status	Checksum
0XFF 0XFF	0X01	0X02	0X00	0XFC