ROBS-181 MANUEL

OC-Servo Electronics Technology Co.,Ltd

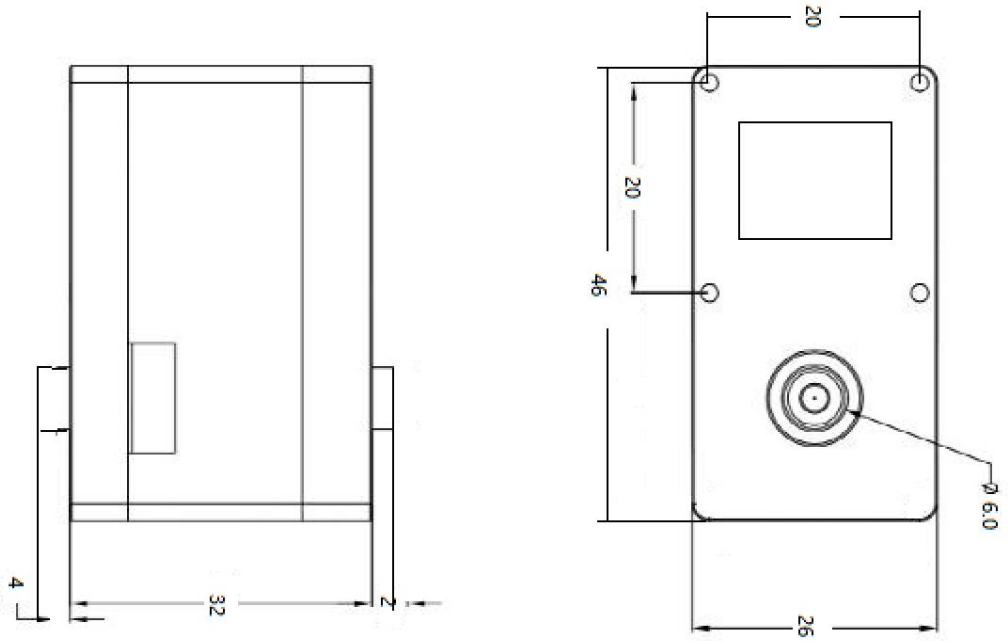
Chapter 1 Overview

1.1 Properties

ROBS-181 is a robot servo developed and produced as a set of motor, servo drives, and modbus communication interface in an integrated servo unit. It's mainly used for robot joints, wheel drives or mechanical arms, and also other situations that need precise position control. The product features are as follows:

- Large Torque: 18kgf·cm (12.0V)
- High Voltage supply： DC 7.0V ~ 12.0V
- High Resolution: 0.32°
- Unique connection method，suitable for a variety of combination assembled.
- High-precision full metal gears，double balls bearing
- High-Voltage high-torque iron core motor
- Rotation range 0~330°in servo mode
- Can be set as motor mode, rotating continuously
- Link: Multi Drop Bus，can be cascaded to 254 units in theory
- Communication speed: 38400bps-1Mbps
- With position，temperature，speed，voltage，and other feedbacks.

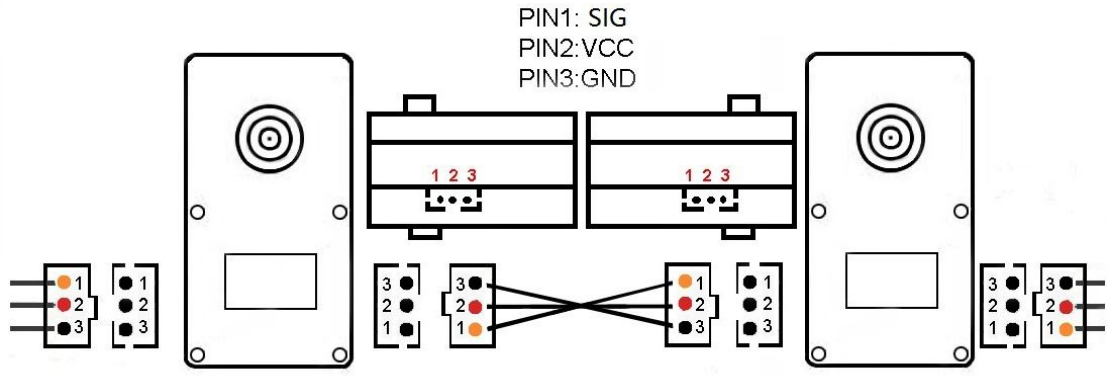
■ 1.2 Structure & Size



■ 1.3 Electrical connection

■ 1.3.1 Pin Definitions

The connection of ROBS-181 is as shown below, two series terminals can be individually connected in series.

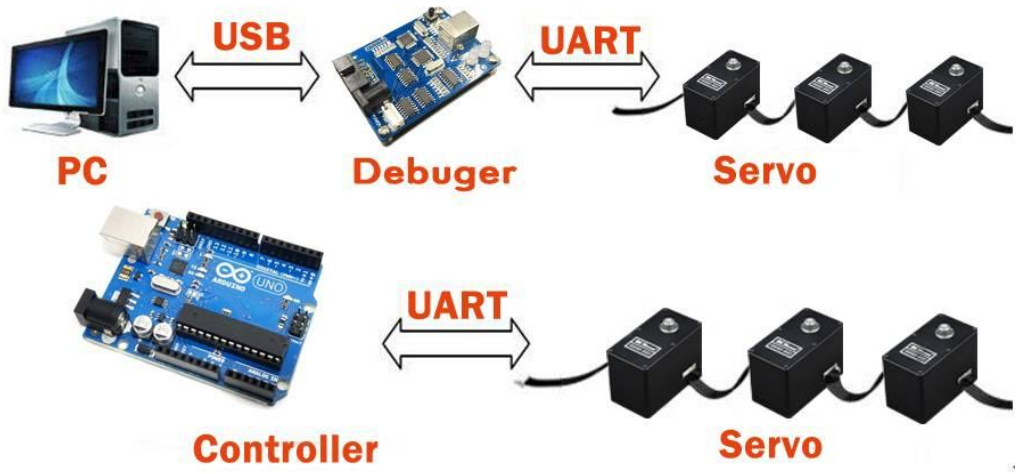


■ 1.3.2 Servo Communication Protocol

ROBS-181 uses the full-duplex asynchronous serial communication modbus. Theoretically, 254 robot servos can be grouped by modbus in a series， controlled by serial interface UART asynchronous. Each servo can be as different ID，multiple servos can move unified，and can be also controlled independently.

Its communication instruction is open，communicates with user's PC (controller) through asynchronous serial interface; You are able to set up its parameters and its control mode. By sending commands through the asynchronous serial interface, ROBS-181 can be set to motor mode or servo mode. In the motor mode，we can use it as a DC gear motor in an adjustable speed. In the servo mode，ROBS-181 has the rotation range 0~360°,with high precision in position control and has an adjustable speed in servo mode.

You can communicate with the ROBS-181 if you use the Full-duplex UART asynchronous serial interface which is in keeping with the protocol. To control the ROBS-181，you have several ways as follows：



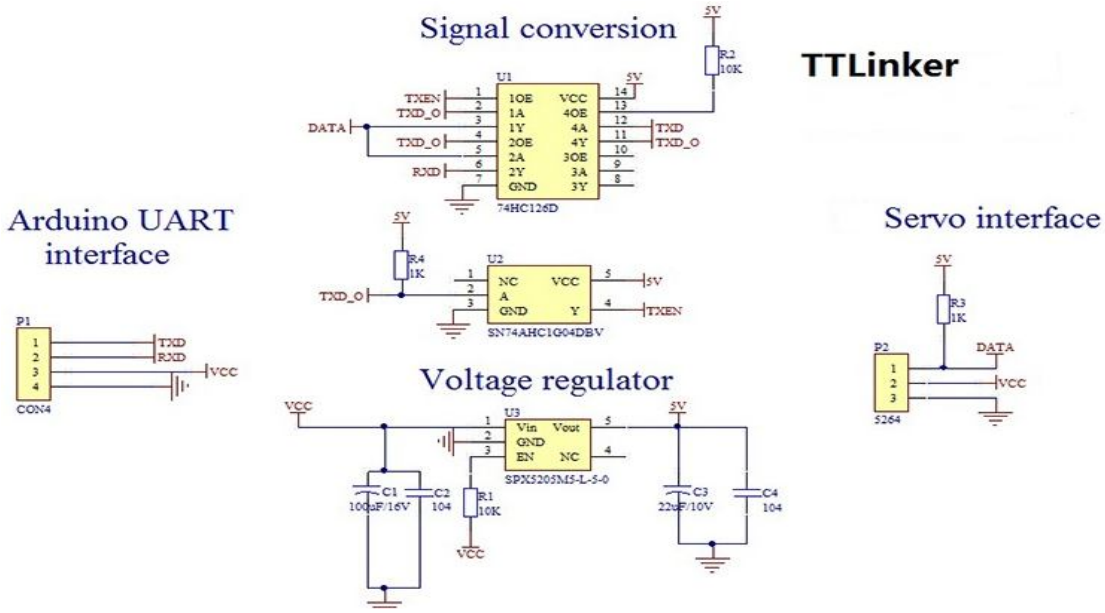
Method 1：Debuger

PC will recognize the debuger as a serial device，data packet will be sent to the servo through the serial ports. ROBS-181 executes the instructions on the packet and if it's a check command, servos will go back to the data packet.

You can also design your own software according to the protocol provided in the Manuel.

Method 2：Arduino board or other controllers

Method 1 enables us to debug our robot servo rapidly，and change its function parameters fast. But in this way，you can't do it without your PC，so you can't make a robot configuration independently. While using Arduino board or other DIY controller, you can control servo through UART port. You need to make a 4-wire to 3-wire TTL level conversion, the schematic is as follows:



■ Chapter 2 Communication Protocol

2.1 Protocol

Between the controller and the servo using question and response mode of communication, the controller sends a command packet, the servo return response packet.

A number of servos are allowed in a network, so each servo is assigned an ID number. The control command issued by the controller contains the ID information, only the servo that matches the ID number can receive the command completely and return the response information.

Asynchronous is a serial communication mode, one frame data is divided into one start bit, eight data bits and one stop bit, no parity bit,

and total of 10 bits.

2.2 OCS instruction packet

Prefix	ID	Data Length	Instruction	Parameters	Sum
0XFF 0XFF	ID	Length	Instruction	Parameter1 ...Parameter N	Check Sum

Prefix: received two 0XFF, ready to run the OCS instruction.

ID:each servo has its own ID number. ID range 0 to 253, 0X00~0XFD(Hexadecimal).

Broadcast ID:ID 254 is the broadcast ID, if the ID number in one instruction is 254(0XFE), all servos receive the order, but no response.

Data length: Please refer to the specific length of each instruction explanation.

Parameter: Supplementary control information except instruction.

Summary： The calculation method is as following：

Check Sum= ~（ID+Length+Instruction+Parameter1+……+ParameterN）

If the Sum in the brackets in this calculation is over 255, then take the lowest of a byte, "~" means inversion.

2.3 Response packet

The response packet is the response of controller to some Instructions：

Prefix	ID	Data length	Instruction	Parameter	Check Sum
0XFF 0XFF	ID	Length	Instruction	Parameter1 ...Parameter N	Check Sum

The returned response packet contains the current state ERROR of the servo. If the current working status of the servo is not normal, it will be

reflected by the byte. The information of each bit is as follows：

BIT	Title	Details
BIT7	0	---
BIT6	0	---
BIT5	Overload	The position output torque is less than the load setting
BIT4	0	---
BIT3	Current error	The current exceeds the specified range
BIT2	Temperature error	The temperature exceeds the specified range
BIT1	Angle error	The angle exceeds the specified range
BIT0	Voltage error	The voltage exceeds the specified range

If error is 0，there is no error.

If the instruction is the instruction(read) READ DATA, then Parameter1 ...

Parameter N is the information.

2.4 OCS Instruction type

OCS Instructions：

Instructions	Function	Value	Data length
PING	Query the working status	0X01	0X02
READ	Query control table of characters	0X02	0X04
WRITE	Write characters to the control table	0X03	N+3
REG WRITE	Similar to WRITE DATA, but not immediately after the control characters written, until the ACTION command to reach	0X04	N+3
ACTION	Trigger REG WRITE Write action	0X05	0
SYNC WRITE	For simultaneously controlling a plurality of servos	0X83	(L + 1) * N + 4
RESET	Reset the control table to the factory values	0X06	0

2.4.1 Stats query Instruction PING

Function Used to read the work state of the servo

Length 0X02

Instruction 0X01

Parameter no

Example: Read Servo 1's working state

Instruction packet：0XFF 0XFF 0X01 0X02 0X01 0XFB

Prefix	ID	Data Length	Instruction	Check Sum
0XFF 0XFF	0X01	0X02	0X01	0XFB

Response packet： 0XFF 0XFF 0X01 0X02 0X00 0XFC

Prefix	ID	Data length	Instruction	Check Sum
0XFF 0XFF	0X01	0X02	0X00	0XFC

2.4.2 Instruction Read

Function Used to read the data inside the servo.

Length 0X04

Instruction 0X02

Parameter 1 Read the address

Parameter 2 Read the length of parameter

Example: Read Servo 1's internal temperature

It is known from the memory control table that the address 0X3F (parameter 1) is the address of the temperature and then needs to read one byte (0X01).

Instruction packet：0XFF 0XFF 0X01 0X04 0X02 0X3F 0X01 0XB8

Prefix	ID	Data Length	Instruction	Parameters	Check Sum
0XFF 0XFF	0X01	0X04	0X02	0X3F 0X01	0XB8

Response packet： 0XFF 0XFF 0X01 0X03 0X00 0X20 0XDD

Prefix	ID	Data length	Instruction	Parameter	Check Sum
0XFF 0XFF	0X01	0X03	0X00	0X1E	0XDD

Read out the data is 0X1E, 0X1E converted to decimal is 30, indicating that the current temperature is 30 ℃.

2.4.3 OCS Write Instruction

Function	This command is used to write parameters to the servo memory control table
Length	N+3 (N is the number of parameter written)
Instruction	0X03
Parameter1	First part of data address
Parameter 2	The first data
Parameter 3	The second data
Parameter N+1	The Nth data

Example: Change a servo's ID to ID1。

The address of the saving ID is 0X05, so you can write 1 in the address 0X05. We use broadcast ID254 (0XFE) to send instructions. If the EEPROM is not unlocked, the data won't be saved when the power is cut off .

Instruction: 0XFF 0XFF 0XFE 0X04 0X03 0X05 0X01 0XF4

Prefix	ID	Data Length	Instruction	Parameters	Check Sum
0XFF 0XFF	0XFE	0X04	0X03	0X05 0X01	0XF4

2.4.4 Instruction REG Write

REG Write is similar with Write instruction, the only difference is the execution time. When the REG WRITE packet is received, the servo will store the received data in the buffer and set the address 0X40 to 1. When the ACTION instruction is received, the stored instruction executes.

Function	This command is used to write parameters to the servo memory control table
Length	N+3 (N is the number of parameter written)

Instruction	0X03
Parameter1	First part of data address
Parameter 2	The first data
Parameter 3	The second data
Parameter N+1	The Nth data

2.4.5 Instruction ACTION

Function	Used to activate the instruction written by REG WRITE instruction.
Length	0X02
Instruction	0X05
Parameter	no

The **ACTION** instruction is useful for controlling multiple servos at the same time.

The ACTION instruction allows the first and last servos to perform their respective actions simultaneously without any delay in the control of the servos with different IDs.

When the ACTION instruction is sent to multiple servos on the series, the broadcast ID254 (0XFE) is used. Therefore, there is no data frame return when this instruction is sent.

Example : Let the servo0 to 0 ° position, and servo1 to turn to 360 ° position, the two servo need to move at the same time

Analysis: As the need for two movements at the same time, we can use

the above 2.4.4 asynchronous write REG_WRITE directive and ACTION instructions to achieve their simultaneous action, so the following steps were written , and at last all instruction will be activated by ACTION instruction. As the servo 0-300 ° corresponds to the value 0-1023, so 0 ° position is 0X0000, neutral point 512 is 0X0200.

ID=0; Instruction = REG_WRITE; Address = 0X2A; Parameter = 0X02, 0X00

ID=1; Instruction = REG_WRITE; Address = 0X2A; Parameter = 0X00, 0X00

ID=0XFE; Instruction = ACTION

Instruction packet of Servo 0: 0XFF 0XFF 0X00 0X05 0X04 0X2A 0X00 0X00 0XCC

Prefix	ID	Data Length	Instruction	Parameters	Check Sum
0XFF 0XFF	0X00	0X05	0X04	0X2A 0X00 0X02	0XCA

Response packet of Servo 0: 0XFF 0XFF 0X00 0X02 0X01 0XFC

Prefix	ID	Data Length	State	Parameters	Check Sum
0XFF 0XFF	0X00	0X02	0X00		0XFD

Instruction packet of Servo 1: 0XFF 0XFF 0X01 0X05 0X04 0X2A 0XFF 0X0F 0XBD

Prefix	ID	Data Length	Instruction	Parameters	Check Sum
0XFF 0XFF	0X01	0X05	0X04	0X2A 0X00 0X00	0XCB

Response packet of Servo 1: 0XFF 0XFF 0X01 0X02 0X00 0XFC

Prefix	ID	Data Length	State	Parameters	Check Sum
0XFF 0XFF	0X01	0X02	0X00		0XFC

ACTION Instruction packet: 0XFF 0XFF 0XFE 0X02 0X05 0XFA

Prefix	ID	Data Length	Instruction	Parameters	Check Sum
0XFF 0XFF	0XFE	0X02	0X05		0XFA

PS: The ACTION instruction packet is sent by the ID254 broadcast instruction, so no response packet data is returned.

2.4.6 Instruction SYNC WRITE

Unlike the REG WRITE + ACTION instruction, the SYNC WRITE has a higher real-time performance. A SYNC WRITE instruction can modify the contents of multiple servos memory control tables at once, while the REG WRITE + ACTION instruction is a step-by-step procedure. When using the SYNC WRITE instruction, the length of the data to be written and the address of the data to be saved must be the same, ie the same action must be performed. Simply can not control a servo at the same time, the other a servo for temperature. Can only control a few servos to move at the same time, or inquire about the temperature of a few servo at the same time, and so on.

Function Used to control several servos to do the same action.**(This order of instruction is H to L)**

ID 0XFE

Length (L + 1) * N + 4 (L: length of each parameter received by servo, N: Number of servo)

Instruction 0X83

Parameter1 Write into first part of data address

Parameter 2 Write the data length(L)

Parameter 3 Write first servo ‘s ID

Parameter 4 Write first date of servo1

Parameter 5 Write second date of servo2

...

Parameter L+3 Write Lth date of servo1

Parameter L+4 Write second servo ‘s ID

Parameter L+5 Write first date of servo2

Parameter L+6 Write second date of servo2

…

Parameter 2L+4 Write Lth date of servo2

Example: Use the SYNC WRITE Instruction to simultaneously control the Servo 0 to the 511 position (0X01FF) in 1000 ms (0X03E8). The servo 1 turns to the 511 position (0X00FF)in 3000 ms(0X0BB8) and the 4th Servo turns to the 255 position(0X00FF) in 4000 ms(0X0BB8).

Analysis: Control several servos, we use the broadcast ID254 (0XFE). Data length is (L + 1) * N + 4. In this example, the data length is 4 and the number of servos is 3. Therefore, the instruction data length is (4 + 1) * 3 + 4 = 19, and 0X13 (Hexadecimal). First address of servo position is 0X2A, data length is 0X04. So the following content (high byte first, low byte in the post):

First address、data length：0X2A 0X04

ID0：goal position：0X01FF；operating time：0X03E8

ID1：goal position：0X07FF；operating time：0X0BB8

ID4：goal position：0X0000；operating time：0X0FA0

Instruction: 0XFF 0XFF 0XFE 0X13 0X83 0X2A 0X04 0X00 0X01 0XFF 0X03 0XE8 0X01 0X01 0XFF 0X07 0XD0 0X04 0X00 0XFF 0X0B 0XB8 0XB4

Prefix	ID	Data Length	Instruction	Parameters	Check Sum
0XFF 0XFF	0XFE	0X13	0X83	0X2A 0X04 0X00 0X01 0XFF 0X03 0XE8 0X01 0X01 0XFF 0X07 0XD0 0X04 0X00 0XFF 0X0B 0XB8	0XB4

PS: The data write order of this instruction is high byte first, low byte after.

2.4.7 Instruction RESET

Function Reset to the factory default value

Length 0X02

Instruction 0X06

Parameter no

Example: Reset Servo 1 to factory default value

Instruction packet: 0XFF 0XFF 0X01 0X02 0X06 0XF6`

Prefix	ID	Data Length	Instruction	Parameters	Check Sum
0XFF 0XFF	0X01	0X02	0X06		0XF6

Response packet: 0XFF 0XFF 0X01 0X02 0X00 0XFC

Prefix	ID	Data Length	Instruction	Parameters	Check Sum
0XFF 0XFF	0X01	0X02	0X00		0XFC

2.5 OCS mode memory control table

The information and control parameters of the robot servo itself form a table that is stored in the RAM and EEPROM areas of its control chip. By changing the contents, you can control the servo constantly. This is called the memory control table.

2.5.1 Descriptions

2.5.1.1 **EEPROM and RAM**

Data in EEPROM area do not change even the power is off, while data in RAM area will be reset each time re-powered, the data won't be saved.

2.5.4.1.2 Byte L and H

High and Low Byte is generated when we need a 16-bits data. Such as: our servo can be controlled in 360 °, through these examples, we know that 0-360° corresponds AD value 0-4095.

4095 convert to hexadecimal 0XFFF that is 0000 1111 1111 1111, red part is high byte H, blue part is low byte L, and we know that low byte L comes first, then the high byte H. So be sure the order in 2.4.6 and 2.4.7.

2.5.2 OCS mode memory control table:

Address	Instructions	Read/Write	Defaut Value	Storage area
0(0X00)	MODEL(L)	Read	--	EEPROM
1(0X01)	MODEL (H)	Read	--	
2(0X02)	--	--	--	
3(0X03)	Firmware version(L)	Read	--	
4(0X04)	Firmware version (H)	Read	--	
5(0X05)	Servo ID	R/W	1(0X01)	
6(0X06)	Baud Rate	R/W	0(0X00)	
7(0X07)	Response delay	R/W	0(0X00)	
8(0X08)	Response level	R/W	1(0X02)	
9(0X09)	Min angel limit (H)	R/W	0(0X00)	
10(0X0A)	Min angel limit (L)	R/W	0(0X00)	
11(0X0B)	Max angel limit (H)	R/W	03(0X03)	
12(0X0C)	Max angel limit (L)	R/W	255 (0XFF)	
13(0X0D)	Max Temperature limit	R/W	80(0X50)	
14(0X0E)	Max input voltage	R/W	130(0X82)	
15(0X0F)	Min input voltage	R/W	70(0X46)	
16(0X10)	Max torque (H)	R/W	255(0XFF)	
17(0X11)	Max torque (L)	R/W	3(0X03)	
18(0X12)	PWM Phase mode	R/W	0(0X00)	
19(0X13)	Uninstall condition	R/W	37(0X25)	
20(0X14)	LED Alarm condition	R/W	37(0X25)	

21(0X15)	PID: P Gain	R/W	15(0X0F)	
22(0X16)	PID: D Gain	R/W	00(0X00)	
23(0X17)	PID: I Gain	R/W	00(0X00)	
24(0X18)	Start Power (H)	R/W	0(0X00)	
25(0X19)	Start Power (L)	R/W	0(0X00)	
26(0X1A)	CW dead band width	R/W	1(0X02)	
27(0X1B)	CCW dead band width	R/W	1(0X02)	
28(0X1C)	Max integral limit (L)	R/W	0(0X00)	
29(0X1D)	Max integral limit (H)	R/W	0(0X00)	
30(0X1E)	Differential sampling factor	R/W	0(0X00)	
31(0X1E)	Torque step	R/W	0(0X00)	
32(0X20)	Position step	R/W	0(0X00)	
33(0X21)	Output shaft neutral point correction (L)	R/W	0(0X00)	
34(0X22)	Output shaft neutral point correction (H)	R/W	0(0X00)	
35(0X23)	Running mode	R/W	0(0X00)	
36(0X24)	Angle feedback mode	R/W	0(0X00)	
37-39	--	R/W	--	
40(0X28)	Torque switch	R/W	0(0X00)	
41(0X29)	--	R/W	--	
42(0X2A)	goal position (H)	R/W	--	
43(0X2B)	goal location (L)	R/W	--	
44(0X2C)	operation time (L)	R/W	0(0X00)	
45(0X2D)	operation time (H)	R/W	0(0X00)	
46(0X2E)	operation speed (L)	R/W	208(0XD0)	
47(0X2F)	Operation speed (H)	R/W	7(0X07)	
48(0X30)	Lock sign	R/W	1(0X01)	
49(0X31)	Number of turns (L)	R/W	0(0X00)	
50(0X32)	Number of turns (H)	R/W	0(0X00)	RAM
51(0X33)	Relative movement sign	R/W	0(0X00)	
52-55	--	--	--	
56(0X38)	current position (H)	Read	?	
57(0X39)	current position (L)	Read	?	
58(0X3A)	Current speed (H)	Read	?	
59(0X3B)	Current speed (L)	Read	?	
60(0X3C)	Current lead (H)	Read	?	
61(0X3D)	Current lead (L)	Read	?	
62(0X3E)	Current voltage	Read	?	

63(0X3F)	Current temperature	Read	?	
64(0X40)	REG WRITE sign	Read	0(0X00)	
65(0X41)	ERROR	Read	?	
66(0X42)	Actuator operating signs	Read	?	
67(0X43)	The current goal location (L)	Read	?	
68(0X44)	The current goal location (H)	Read	?	
69(0X45)	Current current (L)	Read	Not support	
70(0X46)	Current current (H)	Read	Not support	
71(0X47)	The current number of turns (L)	Read	?	
72(0X48)	The current number of turns (H)	Read	?	

2.5.2 Details of the list:

"-" that can not be modified in the Memory control table

Address: 0X05

This address is used for storage of servo ID, able to read/write, default value is 1(0X01)

Address: 0X06

This address is used for storage of Baud rate, able to read/write, default value is 0(0X00), Baud rate is 1M.

Address value	Actual baud rate	Baud rate Set	Deviation
0	1M	1M	0.0%
1	500000	500000	0.0%
2	250000	250000	0.0%
3	128000	128000	0.0%
4	115107.9	115200	0.079%
5	76923	76800	-0.16%
6	57553.9	57600	0.008%
7	38461.5	38400	-0.16%

Address:0X07

This address is used for storage of Response delay，able to read/write, default value is 0(0X00)

When the servo received an Instruction to be answered，the delay time can be set as you like. Time range: parameter(0~255)*2μs， if parameter is 100，the response is 200μs. Default parameter is 0，it means it response in a shortest time，since the servo requires a minimum response time of about 8μs，the practical minimum response time is 8μs.

Address: 0X08

This address is used for setting response level，able to read/write, default value is 2，servo turns the Instructions back.

Address value 0X10	Respond level
0	Only the Read command and the Ping command are answered
1	All instructions return the reply packet (except broadcast)

Address：0X09~0X0C

This address is used for setting angel range， able to read/write. Min angle limit and max angle have effect to goal position. The minimum angle limit must be less than the maximum angle limit.

Address：0X0D

This address is used for setting max temperature of servo，able to read/write，max temp is set to 80℃.

Address：0X0E～0X0F

This address is used for setting upper limit and lower limit of voltage，able to read/write.

Address：0X10～0X11

This address is used for setting max output torque，able to read/write，1000 is the maximum output.

Address：0X13-0X14

The address is used to set the unloading conditions of the servo, able be read or write

BIT	Function
BIT7	--
BIT6	--
BIT5	If set to 1, the torque is unloaded when overload condition occurs. Then Led alarm.
BIT4	--
BIT3	If set to 1, the torque is unloaded when an over current condition occurs. Then Led alarm.
BIT2	If set to 1, the torque is unloaded when an over heat condition occurs. Then Led alarm.
BIT1	If set to 1, the torque is unloaded when angle sensor error condition occurs. Then Led alarm.
BIT0	If set to 1, the torque is unloaded when it's out of range of voltage. Then Led alarm.

If the above occurs at the same time, follow the logic [OR] . LED alarm condition (0X14) Set to 0 to turn off the LED, otherwise turn on the LED.

Address：0X15~0X17

This address is used for parameter P，I，D，able to read/write.

Brief description of PID， please see website link below：

http://en.wikipedia.org/wiki/PID_controller

For reference，PID control principle is not limited to the motor (engine) control, however, the theory can be applied to a variety of common control.

Address：0X18~0X19

This address is used for controlling the servo motor starting effort, this use a coreless motor, its response speed and the starting current is relatively small, so this parameter can be set to 0.

Address：0X1A~0X1B

This address is used for setting the dead zone area of CW and CCW

Address：0X1C~0X1D

This address is used for setting upper limit for PID control integral value

Address：0X21~0X22

This address is used for setting neutral point of servo output.

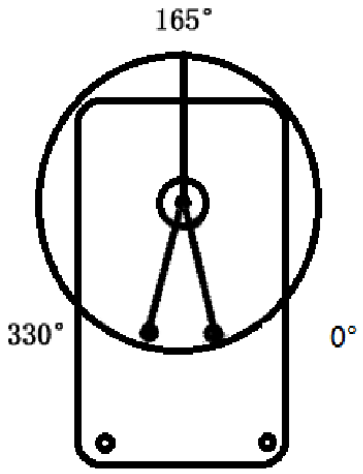
Address：0X28

This address is used for turn on or off the output torque.

Address：0X2A~0X2B

Used to set the address of the servo goal position, you want the servo to

run to a location, you want to write the two locations in the corresponding location. 0 to 1023 (0X03FF) are available.



Address：0X2C~0X2D

It is used to set the address of the time parameter that the servo is running to the goal position. can be used in units of 1 millisecond.

If it is set to 0, which means that the servo rotate at the maximum speed.

For example, it is set to 3000, and the servo reaches the goal position in 3sec.

Address：0X2E~0X2F

This is used to set the speed of the servo, can be read and write. 0-1023 can be used, if the parameters exceed the motor speed limit, will be the fastest speed.

The range and value of this parameter vary according to the following

operating modes.

Servo mode

0-1023(0X03FF)

For example, if it is set to 1023, the servos operate at maximum speed.

Motor mode

For details, refer to 2.5.5 Operation instructions

Address：0X30

Used for locking data

Data	Function
0(0X00)	Data in EEPROM can be modified
1(0X01)	Data in EEPROM can’ t be modified

Address：0X38~0X3F

This address is used for giving feedback; include position, speed, overload, voltage, and temperature, only read.

Address：0X40

If there is REG Write to be activated, it presents 1, when the REG Write is over, it display 0.

2.5.5 MOTOR mode in OCS mode

This servo can also be switched to motor mode in OCS control mode. It can be used for continuous rotating actuator such as wheel and track.

The minimum angle limit and the maximum angle limit (0x09 ~ 0x0C) are set to 0, the time address (0x2C ~ 0x2D) to write a speed value, the servo motor speed mode to turn up. Speed, size and direction of the

control, as shown in the following table:

BIT	11~15	10	9	8	7	6	5	4	3	2	1	0
VALUE	0	0/1	SPEED VALUE									

Address 0x2C ~ 0x2D in the tenth byte used to control the direction (the red position below), the opposite direction as long as 1024 (0X0400)

Note: is written in 0X2C, not 0X2E

Here's an example:

0000 0010 0000 0000 Clockwise 512 speed, converted into hexadecimal 0X0200, in accordance with the H-L write order is 02 00

0000 0110 0000 0000 Counterclockwise 512 speed is converted to hexadecimal 0X0600 (0X0200 + 0X0400), in accordance with the H-L write order is 06 00

Example: Let the servo1 turn clockwise at the speed of 3000.

Switching the operating mode: FF FF 01 07 03 09 00 00 00 00 EB

Operating speed and direction: FF FF 01 05 03 2C 06 00 C4

So the input is FF FF 01 07 03 09 00 00 00 00 EB FF FF 01 05 03 2C 06 00 C4

It will enable the 1st servo to rotate clockwise.

Remember to stop the servo: FF FF 01 05 03 2C 00 00 CA

By the way, if you want to return to the servo mode, we should change the operating mode back.

Caution:

1. This product is a high-precision product, do not artificially rotate the arm vigorously , so as not to damage the inside of the product
2. This product is a high-torque servos, exercise caution when using, to prevent accidental injury
3. Remember not to increase servos when the servos on connection is working
4. This product is similar electronic products, so as not to overload, reasonable running torque ≈1/3stall torque

5. Do not use excess pressure, otherwise easily lead to damage to the product