ICS3.5 Software Manual

Command Refarence

©KONDO KAGAKU CO.,LTD　Aug, 2015　1st Edition

About ICS3.5

ICS(Interactive Communication System) is a bi-directional data communication standard between module control boards. It enables tasks such as communication for controlling servos with the control board, and changing servo motor settings using a PC, etc.

ICS3.5 is a higher level standard than the conventional ICS2.0 and ICS3.0 standards for controlling robot servo motors that includes additional and extended functions. It enables changes to various parameters within the servo motor by serial communication, and the following points have also been extended.

<Switching between "Serial" and "PWM">
ICS3.5 enables the same serial control function provided by ICS3.0 using the PWM signal.

■Main Features of ICS3.5 Serial Communication

・Max. 1.25Mbps high speed communication
・In addition to "speed" and "stretch", various servo motor characteristics such as "temperature limit" and "current limit" can be changed as desired during operation.
・Max. 32 multi-drop connections can be made to the connecting module (maximum differs depending on module and control board performance)

■Transmission Command Loop Back

ICS3.5 uses a single data line. Therefore command from the TX of the micro computer / PC is echoed (loop back) to the RX. The RX will first receive the same command from the TX before receiving the data from the servo motor.

■Multidrop Connection

The ICS utilises a multidrop connection that allows all devices to be serially linked using 3 wires. Each device (Servo Motor) share the same signal, power and ground line. The receiving devices however, only responds to their own ID.

■ICS3.5 Compatible Servos

【HV servo (Power supply voltage: 9 - 12V)】
・KRS-6003RHV ICS
・KRS-4034HV ICS
・KRS-4033HV ICS
・KRS-4032HV ICS
・KRS-4031HV ICS
・KRS-2572HV ICS
・KRS-2552RHV ICS
・KRS-2542HV ICS

【6V servo (power supply voltage: 6 - 8.4V)】
・KRS-3304 ICS
・KRS-3204 ICS

The voltage specification of both types of signal line are the same.

■Various Functions

*Default setting values of each parameter differ according to the servo. See the KRS Servo Series Instruction Manual for details of default values.

【ID】

Sets the servo ID number.

[Setting Range] 0-31

【Signal Speed】

Sets the signal speed with the servo motor.

[Setting Range] 115200bps, 625000bps, 1.25Mbps

【Stretch】

Changes the retention properties of the servo motor.
Reducing this value reduces the retention power of the motor, making it softer like a spring.

[Setting Range] (Soft) 1 - 127 (Hard)

Stretch (SET1) (SET2) (SET3) are values used by characteristic change in Heart To Heart 3. The parameter range is the same as above.

【Speed】

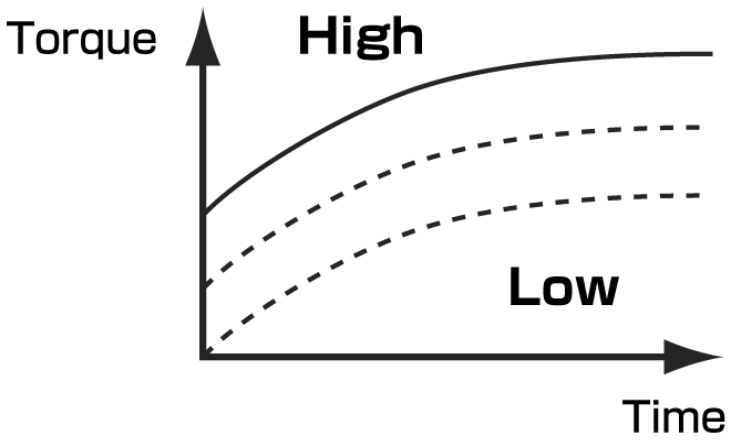
Sets the maximum rotational speed of the servo motor.

[Setting Range] (Slow) 1 - 127 (Fast)

【Punch】

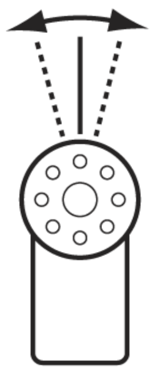
Sets the torque offset for the initial motion of the servo motor.
Increasing this value increases the amount of power output when the servo motor starts operating.

[Setting Range] (Low) 0 - 10 (High)



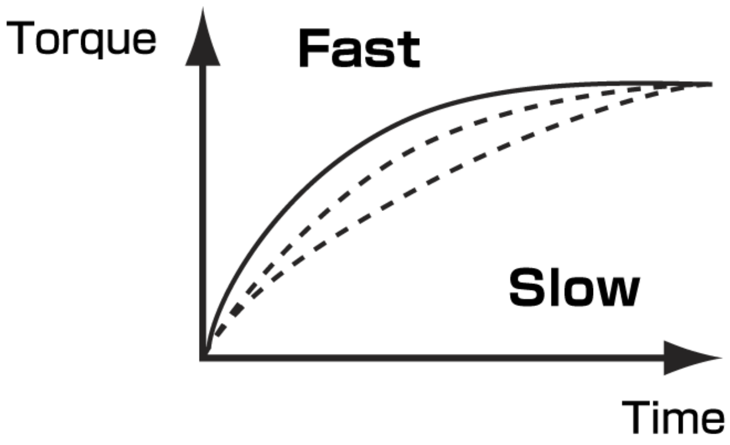
【Dead Band】
Sets the range of the neutral band (dead band) for the servo motor. Increasing this value increases the size of the range, and makes the neutral position vague. It is possible to stop servo jitter by expanding the neutral band range.

[Setting Range] (Min) 0 - 10 (Max)



【Response】
Sets the starting characteristics of the output shaft when it starts operating.
Decreasing this value makes the initial movement smoother.

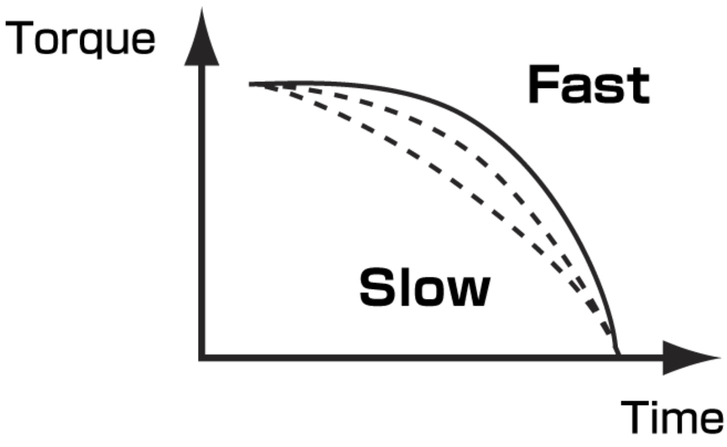
[Setting Range] (Slow) 1 - 5 (Fast)



【Damping】

Sets the braking characteristics of the output shaft when it stops operating.
Decreasing this value makes the brakes operate earlier, making the movement before stopping smoother.

[Setting Range] (Slow) 1 - 255 (Fast)



【Protection】

Sets the time until the protection operation starts.
The protection function operates when the servo is locked. It automatically reduces the servo power by 50% after startup
It is automatically activated when the lock is released after the servo returns to the home position.
<u>However, the protection function is only enabled when the servo speed parameter is set to 127.</u>

[Setting Range] (Short) 10 - 255 (Long)

Time for 1 Parameter
Approx. 0.056sec

*Time before the start of operation may differ depending on the operating conditions and model.

【Limiter】

Sets the maximum operating angle range of the servo motor.

[Setting Range]	Forward	(Min) 8000 - 11500 (Max)
	Reverse	(Min) 3500 - 7500 (Max)

【Temperature Restriction】

Sets the temperature threshold of the servo motor unit.
If the sensor installed in the servo board outputs a value higher than the set temperature value, the servo operates at a decreased power level.
It recovers when the threshold value is exceeded.

[Setting Range] (High) 1 - 127 (Low)

Temp.	Value
100℃	30
90℃	47
80℃	60
70℃	75
60℃	87

*These are approximate values. Actual operation may differ according to the circumstances.

【Current Restriction】

Sets the current threshold value.
If the sensor installed in the servo board detects a current higher than the set value, the servo operates at a decreased
It recovers when the current drops lower than the threshold value.

[Setting Range]	Forward	(Low) 0 - 63 (High)
	Reverse	(Low) 64 - 127 (High)

Current Value	Setting Value
0A	0
0.1A	1
0.5A	5
1.0A	10
1.5A	15
2.0A	20

*These are approximate values. Actual operation may differ according to the circumstances.

【Flag】

Selects whether to use reverse, serial signal (PWMINH), slave or rotation mode.

Name	Function
Reverse	Operates in the opposite direction to the set direction.
Serial Signal (PWMINH)	Prohibits operation in the PWM mode.
Slave	Does not return a reply command to the transmission comm
Rotation mode	Output axis rotates infinitely.

▼Double Servos

"Double Servos" indicates a state in which 2 servos are combined back-to-back. Attaching both axes to each servo achieves approximately twice the torque. Double servos can be used when "Flag" is set to reverse or slave.

Connect both servos and set to the same ID, and the servos receive the same command for operation.
However, as the servos transmit a reply command when a command is received, interference occurs because both servos transmit the reply command simultaneously. By setting one servo as the "slave" to prohibit the reply command, interference can be prevented even when the same command is received.
Furthermore, when servos are combined back-to-back they operate in opposite directions when the same command is received. By setting the slave servo to "reverse", the slave servo operates in the same direction as the master servo, enabling connection to a servo arm, etc.

Example Double Servo
Servo: KRS-6000 Series

【User Offset】

The default position of the output axis can be set as desired by the user.

[Setting Range] (Reverse) -127 - 127 (Forward)

Before Starting Communication

■Transmitting from a PC

Servos can be controlled directly from a PC by connecting with the USB adapter.

Items to Prepare

●USB adapter

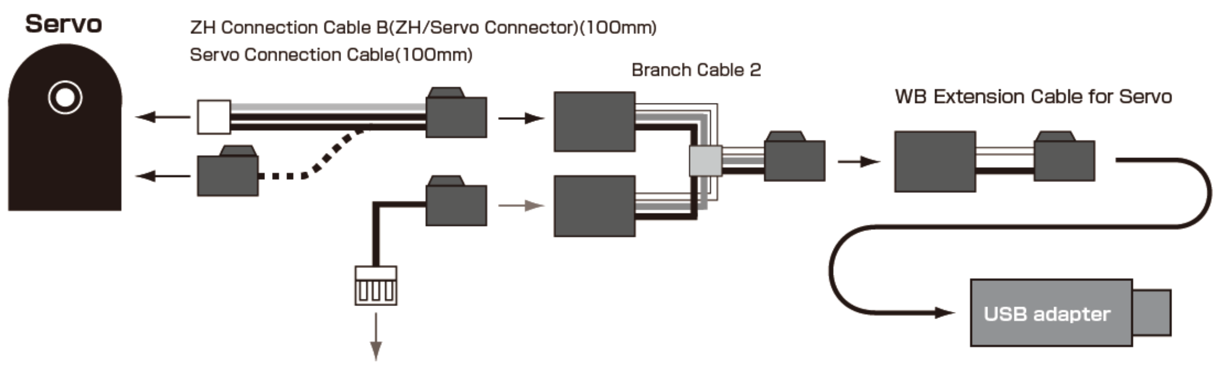
・Dual USB adapter HS (No.02116)
・ICS USB adapter HS (No.02042)
・ICS USB adapter (No.01106)

＊ "HS" in the product name is the abbreviation of "High Speed". The maximum communication speed is
＊ The ICS USB adapter (No.01106) is not compatible with high speed communication, and cannot be used at
＊ All required cables are included in the Dual USB Adapter HS set.

●Power Supply

【For HV Servo】	【For 6V Servo】
ROBO Power Cell F3-850 Type (Li-Fe)	ROBO Power Cell F2-850 Type (Li-Fe)
ROBO Power Cell F3-1450 Type (Li-Fe)	ROBO Power Cell F2-1450 Type (Li-Fe)
ROBO Power Cell F3-2100 Type (Li-Fe)	
ROBO Power Cell HV C Type 9N-300mAh Ni-MH	
ROBO Power Cell HV D Type 9N-800mAh Ni-MH	

Connection Method


Servo
ZH Connection Cable B(ZH/Servo Connector)(100mm)
Servo Connection Cable(100mm)
Branch Cable 2
WB Extension Cable for Servo
USB adapter

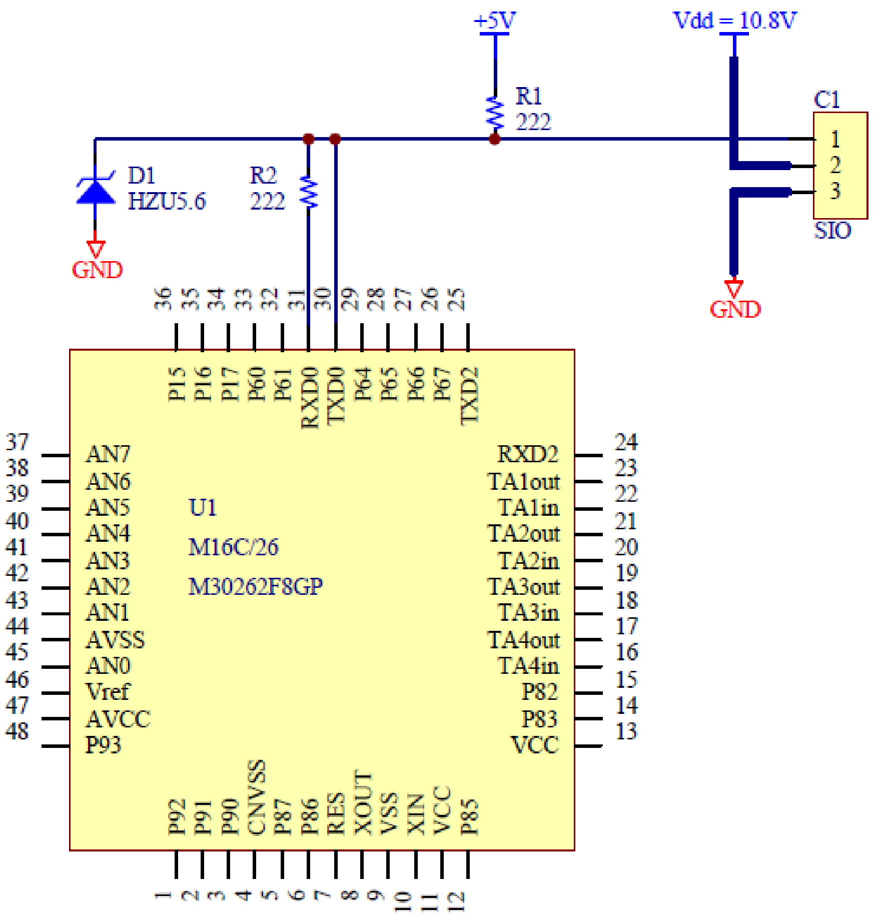
＊ See the KO Driver manual for details on how to install the driver.

▼【KONDO Website】Customer Center → Support Information → Software → KO Driver2015
http://kondo-robot.com/faq/ko-driver-2015

■Using Your Own Board for UART Communication

Prepare the following circuits to operate the KRS Servo from a microcomputer that has a UART terminal.

The circuit diagram below is written only for the serial servo motor drive section of a serial servo motor like the Kondo Kagaku KCB-1 microcomputer board. SIO connector number 1 represents the signal line, 2 represent the power line and 3 represents the ground. Serial servo motors for Kondo Kagaku robots operate on 10.8V power supply voltage using half duplex serial communication. The serial communication (signal) line uses CMOS level (identified as HIGH over approx. 3.3V) negative logic.



To achieve half-duplex communication, the serial communication terminal TXD (transmission terminal) and RXD (reception terminal) are connected together through the SIO connector signal line (no.1) on the microcomputer side. To create a negative logic circuit, a 2.2kΩ pull-up (R1) is used so that the signal voltage of the signal line is 5V. RXD is an input terminal, and has a resistor (R2) connected in series because the CPU may be damaged by noise or static. Furthermore, KCB-1 is connected to a 5.6V breakdown voltage Zener diode. As shown in the circuit diagram, the diode is facing the signal line side. Under normal conditions electricity does not flow from the signal line to the cloud side. However, if the voltage exceeds 5.6V, electricity flows to the ground protecting the CPU from high voltage.
No. 2 terminal is connected to the power supply voltage (servo compatible voltage) and the No.3 terminal is connected to the cloud. This completes preparation or the electrical circuit that operates the servo.

Serial Communication Settings

Signal Speed	115200bps, 625000bps, 1.25Mbps
Bit Length	8bit
Start	1bit
Stop	1bit
Flow Control	non
Parity	EVEN
Polarity	No Reversal
Signal Level	5V TTL

Serial Operation Settings

Serial communication with the servo is possible when the H level is maintained in the signal line for 500ms when the power is turned on. However, we recommend setting the "PWMINH" flag to 1 when operating the robot to prevent switching to PWM mode when a power flicker occurs. (For details of flags, see the EEPROM section.)

Data Structure

No. of Bytes	1(CMD)	2(SC)	3-N-1(DATA)
Details	Command header (main command) + ID number	Sub command	Data

■ Command Header (CMD)
The command header (CMD) section connects the 4 types of main command (position, read, write, ID setting) with the ID number set in the servo motor. Numbers in the main command list below that have "0b" indicate binary numbers. Subsequent numbers that have "0x" indicate hexadecimal numbers.

	CMD(1BYTE)							
	Command			**ID**				
	7bit	**6bit**	**5bit**	**4bit**	**3bit**	**2bit**	**1bit**	**0bit**
Position (decides the rotation angle of the servo motor)	1	0	0	x	x	x	x	x
Read (Reads the parameter. Type is decided by the sub	1	0	1	x	x	x	x	x
Write (Writes the parameter. Type is decided by the sub	1	1	0	x	x	x	x	x
ID (Reads or writes the ID number)	1	1	1	x	x	x	x	x

XXXX represents the 5-bit ID number. For example, when setting an ID=12 (01100 in 5-bit binary) servo in position, the 1st byte of the ICS command (command header, CMD) is "0b10001100".

■Sub Command (SC)
The sub command (SC) is an optional setting of the main command, and includes the speed, stretch and power value of the servo motor. There is no sub command for the position command. The * marks in the following list indicate sub commands for ICS3.5 and later.

EEPROM	0x00	Direct access to the servo motor ROM data.
STRC	0x01	Handles stretch data.
SPD	0x02	Handles speed data.
CUR *	0x03	Reads the current value, or writes the current restriction value.
TMP *	0x04	Reads the servo motor temperature value, or writes the temperature restr

Refer to p. 21 for details on the EEPROM contents.

■ Data(DATA)
Do not specify when reading the data (DATA). When writing data, specify the data to be written to the servo motor. There is a special data structure only for the data section of the position command. See "Position Settings" for details on how to create the following data structure.

POS_H	Upper 7 bits of position data
POS_L	Lower 7 bits of position data

*Structures other than the command header have an MSB of 0, but ID writing is the only exception.

Command List

The table described below has the following structure.

TX or RX	BYTE1	BYTE2	BYTE3	
	Details	Details	Details	…

TX ＝ Send command
RX ＝ Receive command

Position Setting

Function

○ Position Setting Command
Servo can be operated by specifying the angle.

A value from 3500 to 11500 can be specified as the position data for the servo motor. 7500 is the neutral position.
A special operation in which the servo motor is "free" is performed only when 0 is entered as the position data.

Structure

TX	1	2	3
	CMD	POS_H	POS_L

CMD Position Setting Command
POS_H / POSL Servo Setting Turning Angle

RX	1	2	3	4	5	6
	Transmit command loop back			R_CMD	TCH_H	TCH_L

TCH_H / TCH_L Current servo angle

Explanation

MSB			**CMD**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
1	0	0	x	x	x	x	x
Position Setting Command #100xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

MSB			**R_CMD**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	0	0	x	x	x	x	x
Position Reply Command #000xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

*MSB of CMD is masked for return as the servo reply so that it is not mistaken as a command from the host.
*When the signal speed for ID0 is 115.2K, MSB for RX is 1 to ensure compatibility with the conventional ICS2.0. However, this is only for the position setting command.

MSB			**POS_H**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	x	x	x	x	x	x	x
0 Fixed	Setting Turning Angle (lower 7 bits)						

MSB			**POS_L**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	x	x	x	x	x	x	x
0 Fixed	Setting Turning Angle (upper 7 bits)						

MSB			**TCH_H**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	x	x	x	x	x	x	x
0 Fixed	Current Angle (lower 7 bits)						

MSB			**TCH_L**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	x	x	x	x	x	x	x
0 Fixed	Current Angle (upper 7 bits)						

The servo turning angle range is 0 to 16383. However, the setting range for a 270° turning angle servo is 3500 to 115000, with the center at 7500.
MSB other than the command header must be 0, therefore only the lower 14 bits of data are used for 2 byte (16-bit) data. The removed 14 bits are halved, with the upper bits as POS_H and lower bits as POS_L.
For example, if the neutral position is 7500,

7500＝0b00011101_01001100(0b[00][011101_0][1001100]), therefore
POS_H＝0b000111010＝0x3A, POS_L＝0b01001100＝0x4C.

The returned value is divided into 7 bits each in the same way as the transmission command of the current position data. Therefore, the 7 bits need to be combined in order to return the data by the program.

Set the servo angle to 0x00 to set the servo as free.

Serial mode does not have a dedicated position capture command as in the conventional PWM signal. The current position (angle) of the axis is returned as the return value when the operating position is specified. If the current position is not set, the starting operation can be performed safely by specifying the operating position as "0" (Free), then acquiring the position before moving to the desired position.

Example

Transmission command to set the ID=1 servo motor position to 7500

TX	1	2	3
	CMD 0x81	POS_H 0x3A	POS_L 0x4C

Lower 14 bits of the 16 bit data are split into 7 bits each,
with the upper assigned to POS_H and lower to POS_L

RX	1	2	3	4	5	6
	Transmit command loop back			R_CMD 0x01	TCH_H 0x3A	TCH_L 0x4C

Data from 5 and 6 returned to current position

* MSB of CMD is masked for return to prevent the servo reply being mistaken as a command from the host
(When ID is 0 and signal speed is within 115.2Kbps)

Here, "transmission command loop back" indicates when the 3-byte data transmitted by the servo is returned unchanged. This is because the transmission and reception lines are the same lines in the ICS standards, meaning the data is received as it is sent.
As a normal microcomputer cannot receive when transmitting (when there is no flow control), there is no problem in ignoring the loop back and receiving just 3 bytes. However, when an OS is installed in a PC, etc., receive the reply from the servo motor as 6 bytes because the loop back is automatically stored in the buffer.
However, the MSB of the CMD returned as the 4th byte from the servo motor is 0. In this example, R_CMD=01 is returned from the transmission CMD＝0x81.
However, MSB of CMD is returned without a mask only when ID=0 and the signal speed is 115.2kbps. This specification is only for the position command.

Reading Parameters

○ Parameter Read Command

Various setting values can be read. Data that can be read are as follows: Speed, stretch, current temperature value and current values, and EEPROM data.

○ Reading the Current Value

The current value read command reads the current value and direction.
The current value is read as a value from 0 to 63 when in the forward direction, and from 64 to 127 in the reverse direction. This is because the 6-bit value is 1 in the reverse direction.

○ Reading the Temperature Value

The temperature value read command reads the current temperature value.
The temperature parameter is a value from 0 to 127, with a smaller value indicating a higher temperature. As a guide, a parameter value of 60 indicates a temperature of 80°C, and a parameter value of 30 indicates a temperature of 100°C.

Structure

TX	1	2
	CMD	SC

EEPROM

RX	1	2	3	4	5-68
	Transmit command loop back		R_CMD	SC	EEPROM 64bytes

Stretch

RX	1	2	3	4	5
	Transmit command loop back		R_CMD	SC	STRC

Speed

RX	1	2	3	4	5
	Transmit command loop back		R_CMD	SC	SPD

Current

RX	1	2	3	4	5
	Transmit command loop back		R_CMD	SC	CUR

Temperature

RX	1	2	3	4	5
	Transmit command loop back		R_CMD	SC	TMP

Explanation

MSB			CMD				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
1	0	1	x	x	x	x	x
Parameter read command #101xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

MSB			R_CMD				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	0	1	x	x	x	x	x
Read reply command #001xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

*MSB of CMD is masked for return as the servo reply so that it is not mistaken as a command from the host.

SC	
EEPROM	0x00
Stretch	0x01
Speed	0x02
Current	0x03
Temperature	0x04

DATA	
EEPROM	EEPROM Data Reference
STRC	The value in the parentheses of stretch data 1(2) - 127(254) is the EEPROM setting
SPD	The value in the parentheses of speed data 1(1) - 127(127) is the EEPROM setting
CUR	Current value: 0-63 for forward motion, 64-127 for reverse motion
TMP	Temperature value: 1 - 127

Example

Transmission Command for Reading Stretch Data from a Servo Motor whose ID=1

TX	1	2
	CMD 0xA1	SC 0x01

RX	1	2	3	4	5
	Transmit command loop back		R_CMD 0x21	SC 0x01	STRC 0x1E

*MSB of CMD is masked for return as the servo reply so that it is not mistaken as a command from the host. In this case, 0xA1 becomes 0x21.
*This information is for situations in which stretch is 30(0x1E)

Writing Parameters

Functions

○Parameter Write Command
Various setting values can be overwritten (changed). Data that can be written is as follows: Speed, stretch, current restriction value, temperature restriction value and EEPROM data.

○Stretch, Speed, Current Restriction, Temperature Restriction and Signal Speed Settings
When a write command is used, this also changes the speed, stretch, current restriction and temperature restriction information saved to the EEPROM.
When writing other information, such as the signal speed, perform a batch overwrite using the dedicated EEPROM write command.
There is a dedicated command for the ID. See the "ID Command" chapter for details.

*Caution when Using the ICS USB Adapter (No. 01106)
Take care when overwriting signal speed parameters. Once they have been overwritten, subsequent communication will need to take place at the new speed. If a high signal speed (625kbps or 1.25Mbps) is set, it will not be possible to communicate with devices that do not support high-speed communication (such as ICS USB Adapter (No.01106)).

Structure

EEPROM

TX	1	2	3-66
	CMD	SC	EEPROM 64bytes

RX	1	･･･	66	67	68
	Transmit command loop back			R_CMD	SC

Stretch

TX	1	2	3
	CMD	SC	STRC

RX	1	2	3	4	5	6
	Transmit command loop back			R_CMD	SC	STRC

Speed

TX	1	2	3
	CMD	SC	SPD

RX	1	2	3	4	5	6
	Transmit command loop back			R_CMD	SC	SPD

Current Restriction

TX	1	2	3
	CMD	SC	CURLIM

RX	1	2	3	4	5	6
	Transmit command loop back			R_CMD	SC	CURLIM

Temperature Restriction

TX	1	2	3
	CMD	SC	TMPLIM

RX	1	2	3	4	5	6
	Transmit command loop back			R_CMD	SC	TMPLIM

Explanation

MSB			**CMD**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
1	1	0	x	x	x	x	x
Parameter write command #110xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

MSB			**R_CMD**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	1	0	x	x	x	x	x
Write reply command #010xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

*MSB of CMD is masked for return as the servo reply so that it is not mistaken as a command from the host.

SC	
EEPROM	0x00
Stretch	0x01
Speed	0x02
Current	0x03
Temperature	0x04

DATA	
EEPROM	EEPROM Data Reference
STRC	The value in the parentheses of stretch data 1(2) - 127(254) is the EEPROM setting
SPD	The value in the parentheses of speed data 1(1) - 127(127) is the EEPROM setting
CUR	The value in the parentheses of current restriction values 1(1) - 63(63) is the EEPROM setting
TMP	The value in the parentheses of temperature restriction values 1(1) - 127(127) is the EEPROM setting

Example

Transmission Command for Writing 100 as the Speed for a Servo Motor whose ID=10

To give an example, here is the method by which an ICS command is created to change the speed of a servo motor whose ID number is 10. According to the reference manual, a speed value from 0 to 127 can be set (with lower values indicating lower speeds). In this example, 100 is set.
A write command (0b110XXXXX) is used to set a speed for the servo motor.
The ID number is 10 (0b00001010) in binary).
The speed setting subcommand is 2.
As the main command is "0b110XXXXX" and the ID number is "0b00001010", the CMD is 11001010＝202(0xCA).

TX	1	2	3
	CMD 0xCA	SC 0x02	SPD 0x64

The servo motor sends the following reply when a write command such as a speed change is executed.

RX	1	2	3	4	5	6
	Transmit command loop back			R_CMD 0x4A	SC 0x02	SPD 0x64

*MSB of CMD is masked for return as the servo reply so that it is not mistaken as a command from the host. In this case, 0xC2 becomes 0x42.

ID Command

Function

○The ID of the serial servo can be read and written.

Structure

TX	1	2	3	4
	CMD	SC	SC	SC

RX	1	2	3	4	5
	Transmit command loop back				R_CMD

Explanation

MSB				**CMD**			LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
1	1	1	x	x	x	x	x
ID setting command #111xxxxxb			Read ID = #11111b				
			Write ID = Set an ID to be written: 0x00(0) - 0x1F(31)				

(xxxxx is the ID number)

MSB				**R_CMD**			LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
1	1	1	x	x	x	x	x
ID setting command #111xxxxxb			Read ID = The currently configured ID is set.				
			Write ID = The ID used for transmission is set.				

(xxxxx is the ID number)

MSB is not masked, even in replies from the servo that consist only of the ID command.

SC	
Read	0x00
Write	0x01

Be sure to connect one serial servo for each device on the transmission side when using the ID command!
When an ID command is sent to a device with a multidrop connection, all of the devices reply to the command, resulting in mixed signals and invalid data. It also results in written IDs being applied to all devices.

Example

Command for reading the ID of a servo motor connected in a 1:1 configuration

The command for reading ID numbers has a different structure from other commands. Set CMD=0xFF when reading the ID from the servo (to enable IDs to be read even from servo motors whose ID is unknown.) SC is fixed as 0, and the operation is repeated 3 times.

TX	1	2	3	4
	CMD 0xFF	SC 0x00	SC 0x00	SC 0x00

The following data is returned from the servo motor (5 bytes are returned when the reply is received by a computer; otherwise, only 1 byte is returned.)

RX	1	2	3	4	5
	Transmit command loop back				R_CMD 0xF4

R_CMD is a 1-byte data item combining the read ID command 0b111XXXXX and the servo motor ID number 0b000XXXXX. It is returned with MSB as 0. For example, if 25 (0b000110011) is read as the servo motor ID, R_CMD=0b011110011(243＝0xF3) is returned.

Transmission command to set 20(0x14) as the ID of a servo motor connected in a 1:1 configuration

TX	1	2	3	4
	CMD 0xF4	SC 0x01	SC 0x01	SC 0x01

RX	1	2	3	4	5
	Transmit command loop back				R_CMD 0xF4

MSB is not masked, even in replies from the servo that consist only of the ID command.

KONDO KAGAKU CO.,LTD.

EEPROM Data

機能

Byte	Setting range	Example of factory default value	
1	Fixed as 0x5A	0x5A	Backup character: Top 4 bits - This must not be overwritten
2			Backup character: Bottom 4 bits - This must not be overwritten
3	2,4…254 2-step sequence (consisting only of even	60	Stretch gain: Top 4 bits
4	(0x02-0x100)		Stretch gain: Bottom 4 bits
5	1,2,3…127	127	Speed: Top 4 bits
6	(0x01-0x7F)		Bottom: Top 4 bits
7	0,1,2,3…10	1	Punch: Top 4 bits
8	(0x00-0x0A)		Punch: Bottom 4 bits
9	0,1,2,3,4,5	2	Dead band: Top 4 bits
10	(0x00-0x05)		Dead band: Bottom 4 bits
11	1,2…255	40	Damping: Top 4 bits
12	(0x01-0xFF)		Damping: Bottom 4 bits
13	10,11…255	250	Safe timer: Top 4 bits
14	(0x01-0xFF)		Safe timer: Bottom 4 bits
15	See "Flag Details"	0	Flag: Top 4 bits See "Flag Details"
16			Flag: Bottom 4 bits See "Flag Details"
17			Maximum pulse limit: Top byte, top 4 bits
18	3500…11500	11500	Maximum pulse limit: Top byte, bottom 4 bits
19	(0xDAC-0x2CEC)		Maximum pulse limit: Bottom byte, top 4 bits
20			Maximum pulse limit: Bottom byte, bottom 4 bits
21			Minimum pulse limit: Top byte, top 4 bits
22	3500…11500	3500	Minimum pulse limit: Top byte, bottom 4 bits
23	(0xDAC-0x2CEC)		Minimum pulse limit: Bottom byte, top 4 bits
24			Minimum pulse limit: Bottom byte, bottom 4 bits
25	Must not be changed	Must not be	Fixed correction data is written at the factory.
26		changed	Read data must be written as it is.
27	0,1,10	10	Signal speed: Top 4 bits: 10 = 115200bps, 1 = 625000bps, 0 = 1.25Mbps
28	(0x00 / 0x01 / 0x10)		Signal speed: Bottom 4 bits
29	1,2…127	80	Temperature limit: Top 4 bits
30	(0x01-0x7F)		Temperature limit: Bottom 4 bits
31	1,2…63	63	Current limit: Top 4 bits
32	(0x01-0x3F)		Current limit: Bottom 4 bits
33	Must not be changed	Must not be	Fixed correction data is written at the factory.
34		changed	Read data must be written as it is.
35	Must not be changed	Must not be	Fixed correction data is written at the factory.
36		changed	Read data must be written as it is.
37	Must not be changed	Must not be	Fixed correction data is written at the factory.
38		changed	Read data must be written as it is.
39	Must not be changed	Must not be	Fixed correction data is written at the factory.
40		changed	Read data must be written as it is.
41	Must not be changed	Must not be	Fixed correction data is written at the factory.
42		changed	Read data must be written as it is.
43	Must not be changed	Must not be	Fixed correction data is written at the factory.
44		changed	Read data must be written as it is.
45	Must not be changed	Must not be	Fixed correction data is written at the factory.
46		changed	Read data must be written as it is.
47	Must not be changed	Must not be	Fixed correction data is written at the factory.
48		changed	Read data must be written as it is.
49	Must not be changed	Must not be	Fixed correction data is written at the factory.
50		changed	Read data must be written as it is.
51	1,2,3,4,5	3	Response: Top 4 bits - Higher response value = sharper rise.
52	(0x01-0x05)		Response: Bottom 4 bits
53	0±127	0	User offset: Top byte, top 4 bits: 0 at center.
54	(-0x7F-0x7F)		User offset: Top byte, bottom 4 bits
55	Must not be changed	Must not be	Fixed correction data is written at the factory.
56		changed	Read data must be written as it is.
57	0…31	0	ID: Top 4 bits
58	(0x00-0x31)		ID: Bottom 4 bits
59	2,4…254 2-step sequence	120	Characteristic change stretch 1
60	(0x02-0x100)		
61	2,4…254 2-step sequence	60	Characteristic change stretch 2
62	(0x02-0x100)		
63	2,4…254 2-step sequence	254	Characteristic change stretch 3
64	(0x02-0x100)		

*When directly writing the stretch gain of EEPROM numbers 3, 4 and 59-64, enter an even number between 2 and 254

Flag Details

MSB				Flag_ H 4bit			LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	0	0	0	x	0	0	x
No Data				Slave	No Data		Rotation Mode

MSB				Flag_ L 4bit			LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	0	0	0	x	1	x	x
No Data				PWMINH	Fixed as 1	FREE	Reverse

Function	**Flag**
Reverse	0 OFF / 1 ON
FREE	0 OFF / 1 ON (Only available as a reference for reading)
Fixed as 1	Values other than 1 cannot be entered
PWMINH	0 OFF / 1 ON On when using as serial
Rotation mode	0 OFF / 1 ON
Slave mode	0 OFF / 1 ON

* See 【Flag】 in "Various Functions" (p.7) for details on each function.

User Offset Settings

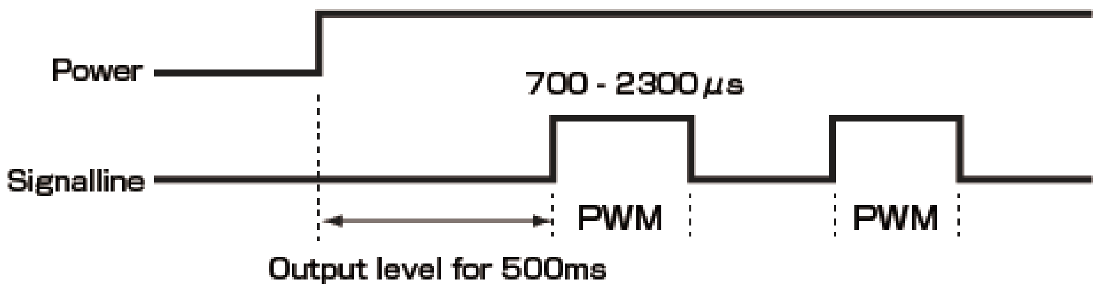
0 is the center point. 1, 2, 3…127 can be set in the forward direction and 255, 254, 253…128 can be set as negative values. For example, set 1 to move +1 or 127 to move +127. Likewise, set 255 to move -1 or 129 to move -127.

*User offset is a function for finely adjusting the output axis of entered control values. These usually do not need to be changed, as they are adjusted at the factory.

Using with PWM

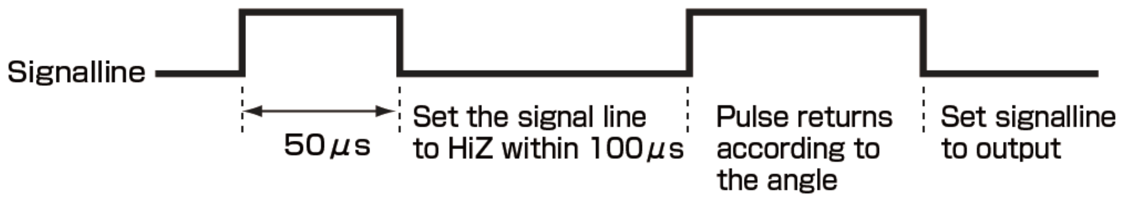
PWM Control Method

To operate the robot using PWM, set the PWMINH flag to 0 and set the signal line at L level for 500ms when turning
The PWM range is 700us-2300us and the servo operation angle is 270 degrees.
The neutral position is 1500us.



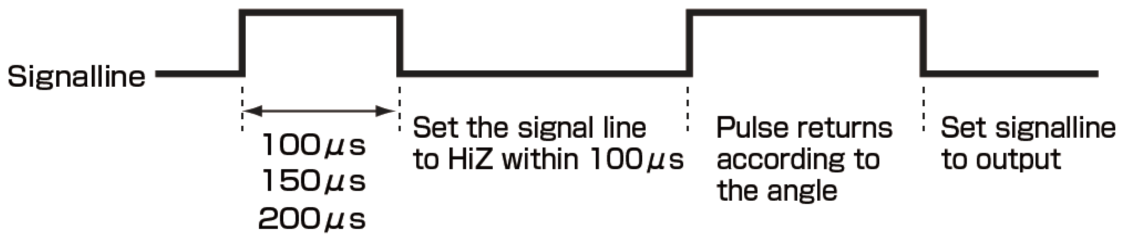
PWM Teaching Function

When a pulse with a width of 50us±5us is input, the servo power decreases and the current output angle is converted to
To acquire the returned pulse, set the signal line to high impedance within 100us after outputting the pulse for 50us.
After acquiring the returned pulse, return the signal line to output.


Signalline
50μs
Set the signal line to HiZ within 100μs
Pulse returns according to the angle
Set signalline to output

Characteristic Changes Using PWM

When a pulse with a width of 100us, 150us or 200us is sent, the servo replaces the respective stretch data of STR1, STR2 and STR3 with the current stretch data. The current output angle is also converted to a pulse width for return.
The imported stretch values are applied to operations but not written to the EEPROM, and the stretch values of the EEPROM are therefore initialized when the power is turned on again.


Signalline
100μs
150μs
200μs
Set the signal line to HiZ within 100μs
Pulse returns according to the angle
Set signalline to output

This function allows you to select one of three types of stretches in real time in PWM operation.
This function is similar to ICS2.0 but the speed is fixed.

General Commands (Other than Servo Motor)

General commands are used to make devices other than the serial servo motor compatible with ICS3.5. The input/output data of the device is mapped in the virtual memory area.

General Read Commands

Functions

Reading Data from a Device

Structure

TX	1	2	3	4
	CMD	SC	ADDR	BYTE

RX	1	2	3	4	5	6	7	8
	Transmit command loop back				R_CMD	SC	ADDR	BYTE

9	10	11	12		8+(2N-1)	8+2N
DAT1_H	DAT1_L	DAT2_H	DAT2_L	･･･	DAT(N)_H	DAT(N)_L

Explanation

MSB			**CMD**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
1	0	1	x	x	x	x	x
Parameter read command #101xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

MSB			**R_CMD**				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	0	1	x	x	x	x	x
Read reply command #001xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

*MSB of CMD is masked for return as the device reply so that it is not mistaken as a command from the host.

SC	
Write Virtual Memory Map	0x7F(Fixed)

ADDR	
Virtual Memory Map Address	0x00(0)-0x7F(127)

BYTE	
Received Data Size	0x01(1)-0x7F(127)

* The data size indicated in "BYTE" is the same as the data size of the virtual memory map on the next page, but each byte of data is divided into a top level and a bottom level when actually sending and receiving data, with the result that the actual data size is twice as large as the displayed size.

DAT1_H-DATA(N)_L
Received data (number of bytes indicated in "BYTE", maximum 127 bytes)
H and L indicate the top 4 bits and bottom 4 bits of the data respectively.

Example

10bit, 4ch analog device (ID=1)

Virtual Memory Map

ADDR	0	1	2	3	4	5	6	7
0	ch1: Top 2 bit	ch1: Bottom 8	ch2: Top 2 bit	ch2: Bottom 8	ch3: Top 2 bit	ch3: Bottom 8	ch4: Top 2 bit	ch4: Bottom 8
1	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-

Example 1: Reading all data (ID=1)

TX	1	2	3	4
	CMD	SC	ADDR	BYTE
	0xA1	0x7F	0x00	0x08

RX	1	2	3	4	5	6	7	8
	Transmit command loop back				R_CMD	SC	ADDR	BYTE
					0x21	0x7F	0x00	0x08

9	10	11	12	13	14	15	16
ch1: Top 2 bits		ch1: Bottom 8 bits		ch2: Top 2 bits		ch2: Bottom 8 bits	
Top 4 bits	Bottom 4 bits	Top 4 bits	Bottom 4 bits	Top 4 bits	Bottom 4 bits	Top 4 bits	Bottom 4 bits

17	18	19	20	21	22	23	24
ch3: Top 2 bits		ch3: Bottom 8 bits		ch4: Top 2 bits		ch4: Bottom 8 bits	
Top 4 bits	Bottom 4 bits	Top 4 bits	Bottom 4 bits	Top 4 bits	Bottom 4 bits	Top 4 bits	Bottom 4 bits

When the device sends data, each byte of the data in the memory map is divided into a top 4 bits and a bottom 4 bits. Single-byte data is created from the divided 4-bit data, with the top 4 bits indicating 0 and the bottom 4 bits serving as the data.

Example 2: Read CH3 data only (when ID = 1)

TX	1	2	3	4
	CMD	SC	ADDR	BYTE
	0xA1	0x7F	0x04	0x02

RX	1	2	3	4	5	6	7	8
	Transmit command loop back				R_CMD	SC	ADDR	BYTE
					0x21	0x7F	0x04	0x02

9	10	11	12
ch3: Top 2 bits		ch3: Bottom 8 bits	
Top 4 bits	Bottom 4 bits	Top 4 bits	Bottom 4 bits

General Write Commands

Functions

Writing Device Data

Structure

TX	1	2	3	4
	CMD	SC	ADDR	BYTE

5	6	7	8		4+(2N-1)	4+2N
DAT1_H	DAT1_L	DAT2_H	DAT2_L	･･･	DAT(N)_H	DAT(N)_L

RX	1	2	3	4	5	6	7	8
	Transmit command loop back				R_CMD	SC	ADDR	BYTE

Explanation

MSB			CMD				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
1	0	1	x	x	x	x	x
Parameter write command #110xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

MSB			R_CMD				LSB
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
0	0	1	x	x	x	x	x
Write reply command #010xxxxxb			Servo ID 0x00(0) -0x1F(31)				

(xxxxx is the ID number)

*MSB of CMD is masked for return as the device reply so that it is not mistaken as a command from the host.

SC	
Write virtual memory map	0x7F(Fixed)

ADDR	
Virtual memory map address	0x00(0)-0x7F(127)

BYTE	
Received data size	0x01(1)-0x7F(127)

* The data size indicated in "BYTE" is the same as the data size of the virtual memory map on the p.26, but each byte of data is divided into a top level and a bottom level when actually sending and receiving data, with the result that the actual data size is twice as large as the displayed size.

DAT1_H-DATA(N)_L
Received data (number of bytes indicated in "BYTE", maximum 127 bytes)
H and L indicate the top 4 bits and bottom 4 bits of the data respectively.